A Weight-Scaling Algorithm for $f$-Factors of Multigraphs

Harold N. Gabow

Abstract

The challenge for graph matching algorithms is to extend known time bounds for bipartite graphs to general graphs. We discuss combinatorial algorithms for finding a maximum weight $f$-factor on an arbitrary multigraph, for given integral weights of magnitude at most $W$. (An $f$-factor is a subgraph whose degree function is the given function $f : V \to \mathbb{N}$.) For simple bipartite graphs the best-known time bound for combinatorial algorithms is $O(n^{2/3} m \log n W)$ [Gabow and Tarjan, SIAM J Comput 18(5):1013–1036, 1989; $n$ and $m$ are respectively the number of vertices and edges.]

A recent algorithm of Duan et al. [in: Proc. of the 47th International Colloquium on Automata, Languages, and Programming (ICALP 2020), 2020] for $f$-factors of simple general graphs comes within logarithmic factors of this bound, $\tilde{O}(n^{2/3} m \log W)$. The best-known bound for bipartite multigraphs is $O(\sqrt{\Phi} m \log \Phi W)$ ($\Phi \leq m$ is the size of the $f$-factor, $\Phi = \sum_{v \in V} f(v)/2$). This bound is more general than the restriction to simple graphs, and is even superior on "small" simple graphs, i.e., $\Phi = o(n^{4/3})$. We present an algorithm that comes within a $\sqrt{\log \Phi}$ factor of this bound, i.e., $O(\sqrt{\Phi} \log \Phi m \log \Phi W)$. The algorithm is a direct generalization of the algorithm of Gabow and Tarjan [J ACM 38(4):815–853, 1991] for the special case of ordinary matching ($f \equiv 1$). We present that algorithm first. Our analysis is a simplified and more concrete version of Gabow and Tarjan [J ACM 38(4):815–853, 1991] and has independent interest. Furthermore the algorithm and analysis are both incorporated, without modification, into the $f$-factor algorithm. To extend these ideas to $f$-factors, the first step is "expanding" edges (i.e., replacing an edge by a length 3 alternating path). Duan et al. [in: Proc. of the 47th International Colloquium on Automata, Languages, and Programming (ICALP 2020), 2020] uses a one-time expansion of the entire graph. In contrast, our algorithm keeps the graph small by only expanding selected edges (edges incident to blossoms, in “$I(B)$ sets”). Expanded edges get “compressed” back to their source when no longer needed. Expansion necessitates using an alternate graph model for blossoms (we call them “e-blossoms”). Compression requires coordinating e-blossoms with standard blossoms.
Keywords  Matching · $f$-Factor · $f$-Matching · Graph · Multigraph · Scaling

1 Introduction

A guiding principle for developing matching algorithms is that any asymptotic time bound achieved for bipartite graphs can be achieved for general graphs—in spite of the complexity introduced by blossoms. (Successful examples include the Hopcroft–Karp cardinality matching algorithm [17] extended to general graphs by Micali and Vazirani [23]; the Hungarian algorithm for weighted matching [20] as implemented by Fredman and Tarjan [7], extended to general graphs by Edmonds [5] with implementations of different components in [10, 16, 26].) We discuss the problem of finding a maximum weight $f$-factor on an arbitrary multigraph, for given integral weights of magnitude at most $W$.

By way of definition recall that for a function $f : V \to \mathbb{N}$, an $f$-factor is a subgraph whose degree function is identically $f$. An $f$-matching allows vertices of degree $< f(v)$. Most weighted degree-constrained subgraph problems, e.g., finding a maximum weight $f$-matching, easily reduce to maximum weight $f$-factor. We refer to ordinary matching as 1-matching, i.e., the special case where $f \equiv 1$.

For combinatorial algorithms the best approach for small integral edge weights is given by scaling algorithms. To state their time bounds $n$ and $m$ denote the number of vertices and edges, respectively, and $\Phi$ is the size of the desired $f$-factor, i.e., $\Phi = \sum_{v \in V} f(v)/2$. (Clearly $n \leq \Phi \leq m$.) The best-known time bounds for the problem on bipartite graphs, given by Gabow and Tarjan [14], are

$$
\begin{align*}
O(n^{2/3} \ m \log nW) & \quad \text{G a simple graph} \\
O(\sqrt{\Phi} \ m \log \Phi W) & \quad \text{G a multigraph.}
\end{align*}
$$

(These bounds are improved by interior point algorithms. Most recently a randomized algorithm running in time $\tilde{O}(m + n^{1.5})$ has been given by Brand et al. [1].)

The above combinatorial bounds are within a logarithmic factor of the best bounds for the unweighted problem (due to Even and Tarjan [6] and Karzanov [19]), thus achieving the goal of scaling algorithms. Note the multigraph bound is more general than the restriction to simple graphs, and is even superior on “small” simple graphs, $\Phi = o(n^{4/3})$.

A recent algorithm of Duan et al. [3] for simple (general) graphs comes within logarithmic factors of the bipartite bound, $\tilde{O}(n^{2/3} \ m \log W)$. We present an algorithm that comes within a $\sqrt{\log \Phi}$ factor of the multigraph bound, i.e., $O(\sqrt{\Phi} \log \Phi \ m \log \Phi W)$.

Both our algorithm and Duan et al. [3] are based on algorithms for the special case of weighted matching ($f \equiv 1$). Duan et al. [3] extends the matching algorithm of Duan et al. [4]. This matching algorithm has time bound $O(\sqrt{n} \log(nW))$, the same as the best-known bound for bipartite graphs (Gabow and Tarjan [14]). The algorithm is based on several distinct ideas (“large blossoms”, the strong polynomial algorithm of [10], and the heavy path algorithm of Gabow [8]) and Duan et al. [4] poses the open problem of a simpler optimum algorithm.
Our \( f \)-factor algorithm is a direct generalization of the matching algorithm of Gabow and Tarjan (Gabow and Tarjan [15], the “GT” algorithm). This algorithm runs in time \( O(\sqrt{n \log n} \, m \, \log(n W)) \). (The bound in [15] has an extra \( \sqrt{\alpha(m, n)} \) factor, which is automatically removed by using the data structure of Thorup [26] for list splitting.) Note the extra factor \( \sqrt{\log n} \) compared to Duan et al. [4], although the simpler approach offers potential regarding the open problem of Duan et al. [4].

The current paper has two halves, the first offering a new view of the GT algorithm and the second extending it to \( f \)-factors. Our presentation of the GT algorithm incorporates an analysis that is fundamentally equivalent to Gabow and Tarjan [15] but has the following advantage. Gabow and Tarjan [15] presents the basic ideas at an abstract level (especially the “crossing function” \( \gamma \), p. 834). We give an explicit description of the graph structures that measure progress of the algorithm, namely the “objective reducers” of Sect. 3.1. These mechanisms are more involved than the usual approach of using only the dual objective function. Continuing, we derive the algorithm’s time bound using a credit system that gives a concrete view of the involved recursive timing argument. Our algorithm for \( f \)-factors uses exactly the same objective reducers and exactly the same credit system. Presenting these ideas first in the context of ordinary matching helps isolate the difficulties introduced by \( f \)-factors. It also offers a supplement to Gabow and Tarjan [15] for readers interested only in matching (as well as a possible jumping off point for improving the algorithm’s extra \( \sqrt{\log n} \) factor).

We extend the GT algorithm to find a maximum weight \( f \)-factor of a multigraph. Several difficulties must be overcome. Unlike ordinary matching, blossoms for \( f \)-factors have pendant edges called “\( I \)-edges” (for “incident” in [10], see Appendix C; these are called \( F \)-edges in [24, Ch. 32–33]). \( I \)-edges invalidate the main mechanism of the GT algorithm (the “unit translation”, for removing “inherited blossoms”, introduced in Sect. 2). We solve this problem by replacing each \( I \)-edge by its “expansion”, i.e., a length 3 alternating path through 2 artificial vertices (Figs. 13, 16). This transformation is often used in matching (e.g., [24, Ch. 32–33], [3]). Duan et al. [3] use it to overcome problems of \( I \)-edges. Specifically [3] expands every original edge (the “blowup graph”). We cannot use this graph to achieve our time bound, since it increases the size of an \( f \)-factor (and its blossoms) up to \( \Theta(1) \) rather than \( \Theta(\Phi) \).

Using edge expansions creates several complications:

Basic properties of blossoms change. On the plus side, blossoms become odd sets, just like 1-matching blossoms, when edges are expanded (Lemma 5.1). This property is crucial for Proposition 3.1, the basis of our entire charging scheme! In contrast, standard \( f \)-factor blossoms do not have fixed parity [10]. On the minus side, blossoms lose their cyclic structure when edges get expanded. We call these expanded blossoms “\( e \)-blossoms” [defined after (4.3)] and we derive and apply their fundamental properties.

Our use of scaling requires an additional property for blossoms: “Base edges” must always be “eligible”. This need not hold, even for standard blossoms (see the discussion preceding Lemma C.1). We give an efficient algorithm to achieve this desirable property (paragraph “Tightening base edges”, Lemma 4.1. The algorithm works for both standard blossoms and \( e \)-blossoms).
Expanded edges must be “compressed”, i.e., replaced by their original source edge, to keep the graph small. Compression may turn an e-blossom into a “baseless” traditional blossom. We give a simple procedure to eliminate baseless blossoms (paragraph “Baseless blossoms”, after (4.8) in Sect. 4.2.2). This procedure can also be applied to algorithms for maximum weight \( f \)-matching, which can have baseless blossoms.

The paper is organized as follows. Section 2 presents the GT matching algorithm. Various low-level details are omitted but nothing of significance for the analysis. Section 3 gives the analysis. It starts with the objective reducers and the mechanisms that exploit them. Then it presents the credit system that proves the Gabow–Tarjan time bound for 1-matching (Theorem 3.7). The proof assumes the basic inequality of scaling, (3.4), which is derived in the last part of the section.

Sections 4–5 present our \( f \)-factor algorithm. It is the GT algorithm, extended to incorporate edge expansion. Section 4 gives the algorithms for edge expansion and compression. The remaining details of the \( f \)-factor algorithm are in Sect. 2. Section 5 gives the analysis. It is the same as Sect. 3 plus modifications for edge expansions, so only the latter is covered. The main result for \( f \)-factors is stated as Theorem 4.2, which gives an outline of the entire discussion of \( f \)-factors.\(^1\)

Appendix A gives a complete statement of Edmonds’ matching algorithm, supporting Sect. 2. Appendix B gives the data structures for efficient implementation of both the GT algorithm and our \( f \)-factor algorithm. Appendix C reviews the \( f \)-factor algorithm of Gabow [10].

**Terminology** \( \log n \) denotes logarithm to the base two. We often omit set braces from singleton sets, denoting \( \{v\} \) as \( v \). So \( S – v \) denotes \( S – \{v\} \). We abbreviate expressions \( \{v\} \cup S \) to \( v + S \).

We use a common summing convention: if \( f \) is a function on elements and \( S \) is a set of elements then \( f(S) \) denotes \( \sum_{s \in S} f(s) \). We extend this to functions \( f \) with two arguments as follows. Let \( S \) and \( T \) be sets of elements. If both arguments to \( f \) are elements then \( f(S, T) = \sum_{s \in S} f(s, t) \). But if the second argument to \( f \) is a set then \( f(S, T) = \sum_{s \in S} f(s, T) \).

In a graph \( G = (V, E) \) for \( S \subseteq V \) and \( M \subseteq E \), \( \delta_M(S) (\gamma_M(S)) \) denotes the set of edges of \( M \) with exactly one (respectively two) endpoints in \( S \). It is sometimes more convenient to make \( M \) an argument rather than a subscript, e.g., \( \delta(S, M_0) \). We omit \( M \) entirely (writing \( \delta(S) \) or \( \gamma(S) \)) when \( M = E \). A loop at \( v \in S \) belongs to \( \gamma(S) – \delta(S) \). If edge \( e \in \delta(S) \) then \( e \) crosses \( S \). \( M \) crosses \( S \) if at least one of its edges crosses \( S \). \( S \) is uncrossed by \( M \) if none of its edges crosses \( S \).

We use the term “matching” as a shorthand for \( f \)-matching. (This is not to be confused with 1-matching.) A vertex on \( < f(v) \) matched edges is free.

\(^1\) Corollary 4.6 shows the first scale of our algorithm finds an \( f \)-factor of a given multigraph in time \( O(\sqrt{\Phi} \log \Phi m) \). Huang and Pettie [18] achieve time \( O(\sqrt{\Phi} m a(m, n)) \) for maximum cardinality \( f \)-matching, a more general result.
2 The Matching Algorithm

We presume familiarity with Edmonds’ weighted matching algorithm and blossoms [5] (see also [2, 21, 22, 24]) as well as previous scaling versions of the algorithm [15] or [4]. Here we give an overview that contains the details needed for the derivation of our result.

The various versions of weighted matching are essentially equivalent from an algorithmic viewpoint. Our discussion focuses on maximum weight perfect matching, which we shorten to maximum weight matching.

2.1 Edmonds’ Algorithm and its Variants

We start with Edmonds’ formulation of maximum weight matching as a linear program. The primal LP defines all possible perfect matchings. The dual LP has variables $y(v), v \in V$ and $z(B), B$ a blossom. $B$ denotes the family of nontrivial blossoms. The dual LP requires $z$ to be nonnegative, and every edge $e$ must satisfy

$$\hat{y}z(e) \geq w(e), \quad (2.1)$$

where $w$ is the given weight function and we define

$$\hat{y}z(e) = y(e) + z\{B : e \subseteq B \in B\}.$$

Here our summing convention implies that for $e = vw$, $y(e)$ denotes $y(v) + y(w)$, and $z\{B : e \subseteq B \in B\}$ denotes $\sum_{e \subseteq B \in B} z(B)$. The unique complementary slackness condition is that equality holds in (2.1) for every edge that is matched.

For convenience our algorithm treats the vertex set $V$ as a blossom, but it differs from all other blossoms. $|V|$ is even (we assume $G$ has a perfect matching) but every other blossom is odd. Our algorithm will assign nonpositive values to $z(V)$, but every other $z$ value must be nonnegative. There is no harm in values $z(V) < 0$, since they are easily eliminated: Add $z(V)/2$ to every $y$-value and change $z(V)$ to 0. This preserves the value $\hat{y}z(e)$ for every edge $e$, so the modified duals are valid.

Edmonds’ algorithm finds a maximum weight matching by repeatedly searching the graph for a maximum weight augmenting path $P$. Wlog the input graph has a perfect matching, so such a $P$ always exists. The algorithm uses $P$ to augment the current matching. Eventually every vertex gets matched and the algorithm halts (Fig. 1).

The algorithm is a primal-dual scheme. It maintains LP duals $y, z$ that are always feasible, i.e., (2.1) holds for every $e$, with equality for every $e$ that is matched or in a blossom subgraph. We say $e$ is dominated when (2.1) holds, and tight when (2.1) holds with equality.

The algorithm assigns nonzero $z$ values to subsets called blossoms. The blossoms form a laminar family that is represented by a blossom tree $T$, defined as follows. The nodes of $T$ are the vertex set $V$, the blossoms of $G$, and the vertices of $V$ that occur in

---

2 Each variant of weighted matching, be it perfect matching, maximum cardinality matching, etc., has a variant of the primal LP.
blossoms. The root of $T$ is $V$. Its children are the maximal blossoms. The children of any blossom are its constituent subblossoms and singleton vertices. (Note the leaves of $T$ are not blossoms.) Each blossom $B$ has a corresponding subgraph. Its vertex set $V(B)$ consist of the vertex descendants of $B$ in $T$. Its edge set $E(B)$ consists of tight edges that span $V(B)$. For convenience we often refer to $B$ as the vertex set $V(B)$.

Recall the detailed structure of a blossom as an odd cycle in a contracted graph [24, p. 416].

Edmonds’ algorithm maintains a *structured matching* which consists of
- a matching,
- LP duals $y$, $z$, that dominate every edge, and are tight on every edge that is matched or in a blossom subgraph,
- a blossom tree.

When the matching is perfect the structure is an *optimum structured matching*. The corresponding matching has maximum weight, and $y$, $z$ are optimum LP duals. So our ultimate goal is to find an optimum structured matching.

Edmonds’ algorithm finds augmenting paths by building a *search structure*. This structure is a forest in a graph formed by contracting various blossoms. The edges of this forest are all tight. Similarly the edges forming the contracted blossoms are tight. These conditions are preserved as the algorithm modifies dual variables and enlarges the search structure. We next give a high-level sketch of Edmonds’ original algorithm. For completeness Appendix A gives detailed pseudocode for the algorithm.

Edmonds’ algorithm has five types of steps: *grow*, *blossom*, *expand*, *augment* and *dual adjustment*. The search structure is constructed by repeatedly executing grow, blossom and expand steps. An augmenting path $P$ (of tight edges) may be discovered in a blossom step. In that case the algorithm proceeds to the augment step. It enlarges the matching by rematching $P$. Then an entirely new search structure is constructed.
Repeat the following procedure until the matching becomes perfect (in the augmenting step).

- rematch a maximal set of disjoint augmenting paths of eligible edges
- construct a maximal search structure of eligible edges in $S$
- adjust dual variables in the search structure

**Fig. 2** Pseudocode for Edmonds’ algorithm with near-optimum duals

and the procedure repeats. The algorithm halts when an augment makes the matching perfect. (We assume the input graph has a perfect matching.)

If the search structure becomes maximal—i.e., no grow, blossom, or expand step can be executed, and no augmenting path has been found—then a dual adjustment step is executed. It modifies the dual variables to guarantee the search structure can change (via grow, blossom or augment steps). Eventually (after some number of dual adjustments) the desired augmenting path of tight edges is found. (For further details of Edmonds’ algorithm see e.g., [2, 4, 5, 15, 21, 24].)

**Near optimality** Edmonds’ original algorithm is easily modified to use a variant of LP dual variables: Define near-optimum duals by

$$
\begin{align*}
\hat{y}z(e) &\geq w(e) - 2 & & e \in E \\
\hat{y}z(e) &\leq w(e) & & e \in M \cup \bigcup_{B \in \mathcal{B}} E(B)
\end{align*}
$$

near domination

near tightness.

(2.2)

Here $M$ denotes the current matching and $E(B)$ denotes the set of edges in the odd cycle that defines blossom $B$.

Consider a graph that satisfies the conditions for an optimum structured matching except that the duals are near-optimum rather than optimum. The perfect matching $M$ weighs $\geq W^* - n$, for $W^*$ the maximum weight of a perfect matching of $G$ ([15, Lemma 2.1], [4, Lemma 2.2]). Thus if the given weight function $w$ is replaced by an integer multiple $aw$, $a > n$, the matching given by the blossom tree has maximum weight. The duals $y, z$ need not be optimum LP duals. Such duals are required in some applications. They can be easily derived from the near-optimum duals, see [15, Theorem 10.4]. Thus we can take our goal to be finding an optimum structured matching with near-optimum duals.

The advantage of near-optimum duals is that augmenting paths can be found in batches rather than individually. Specifically, when Edmonds’ algorithm uses LP duals a search is guaranteed to find one augmenting path, if such exists. Near-optimum duals provide a stronger guarantee: After a given search, rematching a maximal collection of disjoint augmenting paths guarantees that duals must change to get another augmenting path. Details of a depth-first search that finds the maximal collection are given in [15] and also [12]. Figure 2 gives high level pseudocode for this batching version of Edmonds’ algorithm.

The advantage of batching augmenting paths dissipates in the later searches of the algorithm: As time goes on the “batches” have smaller size, eventually just a few augmenting paths. In keeping with this our algorithm operates in “phases”. In Phase
1 batching is advantageous. In Phase 2 batches have shrunk to small size. Phase 3 actually has no augmenting paths.

In order to modify Edmonds’ algorithm to use near-optimum duals, we no longer use tightness as the criterion for being in the search structure. Call an edge eligible if it can be in the search structure. The condition determining eligibility depends on the phase. In Phase 1 an edge \( e \) is eligible if it essentially satisfies (2.2) with equality, specifically

\[
\hat{y}z(e) = \begin{cases} 
    w(e) & \text{if } e \text{ is matched} \\
    w(e) - 2 & \text{if } e \text{ is unmatched.}
\end{cases}
\]  

(2.3)

Edges that form contracted blossoms of the search structure satisfy

\[
\hat{y}z(e) \in \{w(e), w(e) - 2\}.
\]

(2.4)

This modification is needed since blossom edges enter the search structure as eligible edges [satisfying (2.3)] but can get rematched in augments.

In Phases 2 and 3 the definition is less restrictive: \( e \) is eligible if it satisfies (2.4). Again the reason for the change is due to blossoms.

Modulo these changes, the search structure is constructed as in Edmonds’ original algorithm. Consider now Fig. 2. Note the rematch step is given first. This handles the possibility that augmenting paths exist for the starting dual variables. This will be the case in our algorithm.

For succinctness Fig. 2 omits obvious logic to increase efficiency: When no augmenting path is found, the current search structure can be extended rather than constructed starting from scratch. Figures 4, 5, 6 and 7 present our algorithm following the same philosophy, omitting obvious optimizations in favor of conceptual simplicity.

### 2.2 Overall Algorithm: The Scaling Loop

Our overall algorithm is a scaling loop. Here we follow the description in [4]. Take \( W \) so that the given weight function, denoted \( \hat{w} \), takes on integral values in \([0, W]\). (Negative weights are easily eliminated since every perfect matching of \( G \) has cardinality \( n/2 \).)

Define the weight function \( \overline{w} = n\hat{w} \). Let \( b = \lfloor \log nW \rfloor + 1 \). Every value \( \overline{w}(e) \) is an integer of at most \( b \) bits. We treat each value \( \overline{w}(e) \) as an integer of exactly \( b \) bits, by possibly adding leading 0’s. The algorithm operates in \( b \) scales.

For \( i = 1, \ldots, b \), the \( i \)th scale finds a near-optimum structured matching for the weight function

\[
w(e) = 2 \times (\text{the leading } i \text{ bits of } \overline{w}(e)).
\]

To justify this outline note that the \( b \)th scale uses the weight function \( 2\overline{w} = 2n\hat{w} \), and it gives the desired near-optimum structured matching. (We are using \( a = 2n \) in the above discussion of near-optimum matchings.)
Fig. 3 DismantlePath(Q) searching major path P(Q). Blossoms of P(Q) are circular arcs or circles. Dissolved blossoms are dashed. Shell S is atomic. Free vertices are squares. Q contains 4 smaller major paths.

decompose the blossom tree T into major paths P(Q) 
traverse the mpr’s Q in a bottom-up fashion, executing DismantlePath(Q)

Fig. 4 Dismantler algorithm for a scale, pseudocode

The i-th scale starts by scaling up:

\[
\begin{align*}
    w(e) &\leftarrow 2(w(e) + \text{the } i\text{th bit of } w(e)) \quad \forall e \in E \\
    y(v) &\leftarrow 2y(v) + 2 \quad \forall v \in V \\
    z(B) &\leftarrow 2z(B) \quad \forall \text{ blossom } B.
\end{align*}
\]

The matching of the previous scale is discarded, the new matching is empty.

The new duals nearly dominate the new weights: For any edge e, the previous scale ends with \( \hat{y}z(e) \geq w(e) - 2 \), so \( 2\hat{y}z(e) \geq 2w(e) - 4 \). Increasing both sides by 4 makes the left-hand side equal to \( \hat{y}z(e) \) in the new scale, and the right-hand side equal to \( w(e) \) or \( w(e) - 2 \) in the new scale. So the new duals nearly dominate. (The first scale is a special case: Each \( w(e) \) in the new scale is 0 or 2, so we take every new \( y \) value to be 0. Also there are no blossoms.)

2.3 Algorithm for a Scale

The Dismantler algorithm finds a near optimum matching for the new edge weights. The algorithm starts with a collection of inherited blossoms from the previous scale. Since the matching is initially empty these blossoms are no longer valid. A main goal of Dismantler is to discard them. An inherited blossom dissolves when its \( z \) value has decreased to 0, and it is no longer relevant.

We consider \( V \) to be an inherited blossom. It never dissolves: The initialization makes \( z(V) = 0 \), after which \( z(V) \) becomes negative and never increases in value.

As Dismantler progresses it creates new, current blossoms. These are valid for the current matching. As in [4] \( \Omega \) denotes the set of all current blossoms, \( \Omega^- \) denotes the set of all inherited blossoms. \( (V \in \Omega^-) \). Let \( T \) denote the blossom tree of the inherited blossoms. Our strategy for dissolving blossoms is to use the heavy path decomposition of \( T \) [25], defined as follows.
Phase 1: Repeat the following procedure until it ends Phase 1.
rematch a maximal set of disjoint augmenting paths of eligible edges
in every atomic shell of $P(Q)$
/* each pass starts here */
if the atomic shells contain $\leq \pi$ free vertices then end Phase 1
/* $\pi$ is set to $\Theta(\sqrt{|Q| \log |Q|})$ in (3.14), balancing Phases 1 and 2 */
let $\mathcal{A}$ be the set of atomic shells containing a free vertex
for shell $S \in \mathcal{A}$ with size $|S|$ nonincreasing
   if neither boundary of $S$ has dissolved then
      ShellSearch($S$)

Algorithm ShellSearch($S$):
construct a maximal search structure of eligible edges in $S$
translate the boundaries $S^+ , S^-$ of $S$ by 1
adjust dual variables in the search structure by 1

Fig. 5 Pseudocode for Phase 1 of DismantlePath($Q$)

Phase 2: Repeat the following procedure until the atomic shells of $P(Q)$ contain $\leq 1$
free vertex.
let $S$ be the atomic shell of $P(Q)$ that contains a free vertex and has $S^+$ maximal
ShellSearch($S$)
while $\exists$ augmenting path $P$ of eligible edges in an atomic shell
   /* the shell is $S$ or the current atomic shell containing $S$ */
augment $P$

Fig. 6 Pseudocode for Phase 2 of DismantlePath($Q$)

Phase 3: Repeat the following procedure until every blossom of $P(Q)$ has dissolved.
$B_\omega \leftarrow$ the minimal undissolved blossom
ShellSearch($(B_\omega, \emptyset)$)

Fig. 7 Pseudocode for Phase 3 of DismantlePath($Q$)

Let a blossom $B \in \Omega^- - V$ have height $> 1$ in $T$, i.e., $B$ has at least one subblossom
that is inherited. The major child of $B$ is the subblossom $C$ of $B$ with the greatest size
$|V(C)|$. A subblossom with $|V(C)| > |V(B)|/2$ is obviously the unique major child.
If such $C$ does not exist there may be a tie for major child, in which case the major
child is chosen arbitrarily from all candidates. Since $|V(B)|$ is odd a tie can occur
only when every child $C$ has size $|V(C)| < |V(B)|/2$. A blossom $Q \in \Omega^-$ that is
not a major child has a major path $P(Q)$, formed by starting at $Q$ and repeatedly
descending to the major child. $Q$ is a major path root; we abbreviate major path root to
mpr. (See Fig. 3. Note $P(Q)$ consists entirely of blossoms, it does not include leaves
of $T$.)
In addition we take $V$ to be a major path root, with major path simply $V$. The major paths form a partition of $\Omega^-$. 

The DISMANTLER dissolves blossoms using a routine DISMANTLEPATH($Q$), where $Q$ is an mpr. This routine works on the blossoms of $P(Q)$, eventually eliminating them all. DISMANTLEPATH($V$) is a special case: It eliminates all free vertices, eventually constructing a perfect matching of $G$. Figure 4 gives a high-level statement of DISMANTLER.

On entry to DISMANTLEPATH($Q$) every blossom $B \in P(Q)$ has $z(B)$ equal to its scaled up value, and all other blossoms descending from $Q$ are now valid, i.e., either $z(B) = 0$ or $B$ is a current blossom. Clearly DISMANTLEPATH($V$) is executed with every inherited blossom besides $V$ dissolved.

### 2.3.1 Algorithm DISMANTLEPATH($Q$)

The algorithm works on shells of major paths: A shell is a subgraph of $G$ induced by a vertex set of the form $C - D$, where $C$ is a blossom of $T$ and $D \subseteq C$ is either a descendant of $C$ or $\emptyset$. (In the first case $D$ may be either a blossom or a vertex.) The shell has outer boundary $C$ and inner boundary $D$. The shell is even or odd depending on the parity of $|C - D|$. For $Q \neq V$ a shell is odd iff $D = \emptyset$ (since any blossom is odd, as is a single vertex). $P(V)$ has the unique shell $V - \emptyset$, an even shell. Shells having a vertex as their inner boundary are only used in the analysis of the algorithm, not the algorithm itself.

We denote the above shell as $(C, D)$. Alternatively for a shell $S$ we may use the notation $S^-$, $S^+$ to denote the inner and outer boundaries of $S$, respectively. Thus $S^- \subseteq S^+$ and $S = S^+ - S^-$. A blossom $B$ with $S^- \subseteq B \subseteq S^+$ is an interior blossom of $S$. ($B$ is interior to the $T$-path joining $S^-$ and $S^+$..) DISMANTLEPATH($Q$) works on shells $S$ of $P(Q)$, specifically, $S^+$ is a blossom of $P(Q)$, $S^-$ is either a blossom of $P(Q)$ or $\emptyset$.

The main operation to dismantle blossoms is a unit translation. To translate a blossom $B$ by 1 means to decrease $z(B)$ by 2 and increase every $y(v), v \in B$ by 1. Clearly this preserves the value of $\hat{y}z(e)$ for every $e \in \gamma(B)$ and maintains near domination for every $e \in \delta(B)$.

As discussed previously the details of DISMANTLEPATH($Q$) depend on the phase. Phase 1 is the workhorse of the algorithm, and we start by discussing it in detail. Phases 2–3 require less elaboration, being variations of Phase 1.

**Phase 1: algorithm** Phase 1 is presented in Fig. 5. It starts with a rematch step that processes augmenting paths that may exist from previous executions of DISMANTLEPATH or the previous scale. The rest of the execution of Phase 1 is divided into “passes”, where each pass processes every relevant shell of $P(Q)$. In Fig. 5 a comment line marks the demarcation point between passes. Thus a pass begins by constructing $A$, continues by processing the shells of $A$, and ends with the global rematching step (in the next iteration of Pass 1).

At the start of a pass a shell of $P(Q)$ is atomic if every interior blossom has dissolved but neither boundary has dissolved. (A boundary $S^- = \emptyset$ never dissolves.) The atomic shells partition the vertices of the maximal undissolved blossom of $P(Q)$.
The algorithm sorts $A$ on nonincreasing order of $|S|$, the number of vertices in shells $S$.

At any point in DismantlePath define

$$B_{\omega} = \text{the currently minimal undissolved blossom of } P(Q).$$

$A$ always contains the shell $S = (B_{\omega}, \emptyset)$. $S$ is odd (even) for $Q \neq V$ ($Q = V$), respectively. As mentioned $S^- = \emptyset$ never dissolves and it never gets translated [in SHELLSEARCH($S$)].

During the pass SHELLSEARCH is executed for every atomic shell $S \in A$ whose boundaries are still intact when $S$ is examined. The dual adjustment made in SHELLSEARCH($S$) is valid because of the rematch step in the preceding pass. (In detail, the duals can be adjusted in a search structure that contains no augmenting path. This is guaranteed by the rematch step. This also explains why SHELLSEARCH($S$) does not do the rematch: A boundary of $S$ may dissolve in later executions of SHELLSEARCH during the current pass. The new shell containing $S$ may contain augmenting paths that leave $S$.)

The dual adjustment in SHELLSEARCH($S$) may allow the search structure to be modified by new grow, blossom, and expand steps. These steps are not executed in SHELLSEARCH($S$) for two reasons. The first is given above, i.e., the search structure needs to be calculated for the atomic shell that contains $S$ at the end of the pass. Second, the current search structure in $S$ may become invalid due to expand steps: As mentioned above, the edges that replace an expanded blossom $B$ may not be eligible for Phase 1, due to rematching of $B$. Thus the rematching step begins by computing the entire search structure for each shell, and then finds a maximal set of augmenting paths.

Each pass ends with the rematch step. Note that an atomic shell $S$ in that step may consist of a number of shells that were atomic at the start of the pass and got merged because of blossoms dissolving. ($S$ will be atomic at the start of the next pass.)

At the start of a pass let $S$ be the atomic shell having $S^+$ as the maximal undisolved blossom of $P(Q)$. ($S$ exists since $A \neq \emptyset$.) Suppose $S \in A$. SHELLSEARCH($S$) is executed, and the unit translation dissolves $S^+$. (Each of these conditions may fail to occur.) After SHELLSEARCH($S$) returns, the vertices of $S$ are “inactive”—they will never be examined again in the current execution of DismantlePath($Q$). (They are processed when the mpr containing $Q$ gets dismantled.) The pass may go on to deactivate more shells in this manner. Furthermore if $S^-$ dissolves along with $S^+$ then other vertices of $P(Q)$ are deactivated, specifically the vertices of atomic shell $T$ with $T^+ = S^-$. If all remaining vertices of $P(Q)$ are deactivated then DismantlePath terminates. (We shall see there is no Phase 2 or 3.) We say a shell $T$ becomes inactive whenever a unit translation deactivates the vertices of $T$ in any of the scenarios described above. (The rightmost shell of Fig. 3 is inactive.)

We say a shell $S \in A$ is preempted if SHELLSEARCH($S$) is not executed because one of its boundaries has dissolved. A deactivated shell may or may not be preempted. Other scenarios are possible for a blossom $B$ to dissolve without causing a preemption: $B$ may be the boundary of a shell of $A$ that was already processed in the pass, or the boundary of a shell not added to $A$ because it had no free vertex. A given pass may
end with a (new) atomic shell that contains any number of such dissolved blossoms as well as preemption blossoms.

To illustrate the algorithm recall $B_\omega$ defined above, the innermost undissolved blossom of $P(Q)$. Suppose $Q \neq V$, so $B_\omega$ contains an odd number of free vertices. If there is exactly one such free vertex we call it $\omega$. $\omega$ evolves in Phase 1 as follows. A vertex $v$ may at some point become $\omega$ (when augments in the innermost shell leave only one free vertex). Later $v$ may cease being $\omega$ (when blossom $B_\omega$ dissolves, creating a new $B_\omega$ with more free vertices). Later still $v$ or a different vertex may become the next $\omega$ (due to more augments).

**Phase 1: correctness** The correctness of SHELLSEARCH hinges on the fact that it is essentially simulating Edmonds’ algorithm (modified for near optimality) on shell $S$. An iteration of this simulation begins with the first step of Phase 1 (enlarging the matching on $S$), the subsequent construction of the search structure, and following dual variable adjustment. The search structure does not contain an augmenting path. This follows from the preceding enlargement of the matching. If we were to continue Edmonds’ algorithm in $S$, it would adjust duals by some positive integral quantity $\delta$. SHELLSEARCH adjusts the duals by $1 \leq \delta$. We will show this is safe, i.e., the adjustment by 1 preserves near-optimality on every edge $e$ of $G$.

In proof, Edmonds’ algorithm is guaranteed to preserve near-optimality if duals are adjusted by any value $\leq \delta$. So our adjustment is safe if $e$ has both ends in $S$. Suppose only one end $v$ is in $S$. $e$ is not matched, so we need to show near-domination, $\hat{y}(e) \geq w(e) - 2$, after the dual adjustment. The unit translations of SHELLSEARCH increase $y(v)$ by 1 and the following dual adjustment decreases it by $\leq 1$. (In detail, $v$ is either outer, inner, or not in the search structure, and $y(v)$ decreases by 1, $-1$, or 0, respectively.) So near-domination continues to hold.

A last detail of the simulation of Edmonds’ algorithm is that it requires all free vertices to have $y$-values of the same parity. (This is needed to preserve integrality in blossom steps.) A scale starts out with every $y$-value being even. DISMANTLEPATH maintains the property that every free vertex $f$ has $y(f)$ even. In proof $y$ values are only changed in SHELLSEARCH($S$). For $f \in S$ the unit translation of $S^+$ increases $y(f)$ by 1 and the subsequent dual adjustment decreases it by 1 ($f$ is necessarily outer). For $f \in S^-$ the two unit translations increase $y(f)$ by 2.

Other details of correctness are exactly the same as [15] or [4]. We conclude that the simulation of Edmonds’ algorithm is correct.

**Phase 1: efficiency** Let $F$ denote the set of vertices in $Q$ that are currently free and still active (i.e., still in an atomic shell of $P(Q)$). Any vertex $v \in F$ is processed in SHELLSEARCH unless its shell gets preempted by a shell with at least as many vertices. So a preemption doubles the size of its new atomic shell. Thus a free vertex $v$ can be preempted $\leq \log |Q|$ times. This will lead to the crucial inequality $|F|(#passes - \log n) \leq n \log n$. (This inequality is restated as (3.13).) Each pass runs in linear time $O(m)$—data structure details used to achieve this time bound are given in Appendix B.

**Phase 2: algorithm** The algorithm is presented in Fig. 6. The major change from Phase 1 is switching to the less restrictive eligibility condition of (2.4). As previously mentioned this is done since batching is no longer effective. The switch makes the
algorithm behave like Edmonds’ original algorithm, in that rematching an augmenting path may create a new augmenting path. The augment step in Phase 2 allows for this by repeatedly augmenting as many times as possible.

The second change from Phase 1 is to work on just one shell $S$ rather than pass over all the shells. This simplifies low-level details of the algorithm, as discussed below.

The algorithm works on the chosen shell $S$ until it gets completely matched or a boundary of $S$ dissolves. (If the latter, $S$ either gets merged with an adjacent shell, or $S$ is deactivated in \textsc{DismantlePath}($Q$).) Then the algorithm repeats the process on the next shell, etc. If $Q = V$ Phase 2 ends with a perfect matching (the scale is now complete). If $Q \neq V$ Phase 2 ends when either every shell of $P(Q)$ dissolves (\textsc{DismantlePath} is done) or there is a unique free vertex $\omega$ (Phase 3 is executed).

\textbf{Phase 2: efficiency} Fig. 6 is a high-level version of the Phase 2 algorithm. Adjusting duals by only 1 (as done in \textsc{ShellSearch}) is inefficient, since larger adjustments are needed to make progress. An efficient implementation of Phase 2 adjusts duals by the same quantity $\delta$ computed in Edmonds’ algorithm. It is important to use (2.4) as the criterion for eligibility, so that inner blossoms can be expanded without invalidating the search forest.

The low-level implementation of Phase 2 tracks events in Edmonds’ search algorithm using a bucket-based priority queue. In addition to Edmonds events the queue tracks the $z$-values of inherited blossoms so they get dissolved at the appropriate time. The simple details of the priority queue are given in Appendix B.

The high-level analysis of Phase 2 must ensure that the number of buckets in the priority queue is acceptable—this is done in Lemma 3.6. Assuming that result it is clear that Phase 2 uses $O(m)$ time for each augment.

The algorithm of Fig. 6 works on one shell $S$ at a time in order to simplify the priority queue. It is possible to organize Phase 2 like Phase 1, where each pass searches all relevant shells. But this complicates the details of the bucket-based queue, because preemptions of shell searches necessitate the rescheduling of many events.

Choosing $S^+$ maximal also simplifies the low-level algorithm. This choice guarantees that once $S^+$ is chosen as the blossom $B_{\omega}$ it remains as $B_{\omega}$ for the duration of Phase 2.

\textbf{Phase 3: algorithm} Phase 3 is only executed when $Q \neq V$, i.e., $Q$ is a blossom. It is essentially one unsuccessful Edmonds search. Every blossom eventually dissolves, in preparation for the execution of \textsc{DismantlePath} on the mpr containing $Q$. Phase 3 continues to use the eligibility criteria (2.4). Figure 7 gives the detailed statement.

It is worth explaining how blossoms dissolve in this phase. Consider an arbitrary point in Phase 3 when $B_{\omega}$ is the blossom $B \in \Omega^-$. It is possible that some execution of \textsc{ShellSearch}($B$) makes $B$ a blossom (of $\Omega$). Once that occurs, each subsequent execution of \textsc{ShellSearch}($B$) decreases $z_0(B)$ ($B$’s dual as an $\Omega^-$ blossom) by 2 (in a unit translation) and then adjusts duals by increasing $z(B)$ (the dual for $B$ as an $\Omega$ blossom) by 2. In other words the value of $z_0(B)$ gets transferred to $z(B)$, at which point $B$ dissolves. A more direct approach could be used at the expense of a more involved statement of Phase 3.

\textbf{Phase 3: efficiency} For efficiency Phase 3 is not implemented using \textsc{ShellSearch} but rather a simple implementation of Edmonds’ algorithm, e.g. the $O(m \log n)$ algorithm.
Fig. 8 Difficulties and basic notions. a Unit translation of a blossom $B$ increases the dual objective function by 1. b Dual adjustment with near optimum duals need not decrease the dual objective. c Dual objective reducers: $d(v)$, dual adjustments of free vertices; $\kappa_0(INT(S))$, crossings of interior blossoms by matched edges; $\kappa_1(SUB(S))$, double crossings of contained blossoms by matched edges.

of [16]. The search runs on the current minimal shell $(B_\omega, \emptyset)$. If $B_\omega$ becomes a blossom a dual adjustment of $\delta = z(B_\omega)/2$ dissolves $B_\omega$. The time for Phase 3 is strictly dominated by the rest of our algorithm, so we ignore it. A potential issue in the analysis is the fact that Phase 3 can perform many unit translations, far more than Phases 1 and 2. We shall see this does not present a problem.

The overall algorithm The input graph is assumed to be perfectly matchable. The algorithm is initialized by setting every weight $w(e)$ to 0, every $y$ value to $-1$, and taking $z$ identically 0 with no blossoms. These are near optimum duals for any perfect matching $M$. The algorithm starts by scaling the initial $y$ and $z$ values up for the first scale, as described above. Note that the matching $M$ is not needed to start the first scale, and there are no inherited blossoms.

The output of the algorithm is an optimum structured matching (with near-optimum duals). This is accomplished using an outer loop that does scaling. Each scale executes the Dismantler algorithm of Fig. 4. Eventually DismantlePath($V$) ends in Phase 2 with a perfect matching. The last scale ends with a perfect matching of maximum weight (for the given weights). Although the duals at the end of Dismantler are near-optimum, they are easily converted to optimum duals in linear time [15]. We ignore the time for this conversion in the rest of the analysis.

3 The Analysis

We turn to the analysis of the algorithm. Similar to the analysis of cardinality matching algorithms, the goal is to prove a tradeoff between the number of passes in DismantlePath and the number of free vertices. This is ultimately achieved as the inequality (3.13). It follows from two main precursors, (3.25) and (3.4).

We begin by providing motivation, specifically discussing the difficulties of the analysis and the basic notions that we use. Figure 8 illustrates these. It also introduces our notation for these fundamental concepts. [The precise definitions are below, (3.5) and (3.5).]

Previous scaling algorithms make progress by decreasing the dual objective function or the dual objective $y$ values (e.g., [3]). The fundamental difficulty in nonbipartite
matching is that translating a blossom increases the dual objective by 1, Fig. 8a. As in that figure \( \tau(B) \) denotes the number of unit translations of an inherited blossom \( B \). A further difficulty introduced by near optimum duals is illustrated in Fig. 8b which illustrates a dual adjustment of 1. When optimum duals are used the dual objective for the two edges decreases by 1, but for near optimum duals the objective can stay the same. Specifically in both cases the free vertex \( v \) decreases its \( y(v) \) by 1. The inner vertex \( u \) increases \( y(u) \) by 1. For optimum duals matched edges are tight, \( \hat{y}z(e) = w(e) \). So \( y(u') \) decreases by 1, and dual objective decreases by a total of 1. But near optimum duals may have \( \hat{y}z(uu') < w(uu') \). In that case \( y(u') \) does not change and the dual objective remains the same. If there are \( s \) similar edges \( uu' \) the dual objective increases by \( s - 1 \).

Our analysis uses 3 structures that contribute to decreases in the dual objective. As illustrated in Fig. 8c, the first is the aforementioned decrease in \( y \) values of free vertices in a dual adjustment. We track this using the quantity \( d(v) \), the number of dual adjustments made for free vertex \( v \). In fact we also use two variants of \( d(v) \), to track the decrease in dual adjustments made at specific times in the algorithm. The variant \( d(F, C) \) tracks decreases made in searches of a “core” \( C \). The variant \( d(F, Q) \) tracks decreases made in the execution of DismantlePath(\( Q \)) for an mpr \( Q \). \( [d(v), d(F, C), \text{and } d(F, Q) \text{ are defined in (3.5), (3.1), and (3.2) respectively.} \]

The two other structures involve inherited blossoms that are crossed by the current matching \( M \) of the algorithm. (See Fig. 8c and also Fig. 11). The term \( \kappa_0(INT(S)) \) (“crossed interior blossoms”) counts the number of crossings by \( M \) of interior blossoms \( B \). Since the shell \( (B, S) \) is even, each such crossing decreases the objective of the previous scale by \( z_0(B)/2 \), for \( z_0 \) the \( z \) values of the previous scale. (We prove this below. As an example, in Fig. 11 the two crossings of \( B_1 \) replace the possibility of one matched edge in \( B_1 \), thus decreasing the objective by \( z_0(B_1) \).) The term \( \kappa_1(SUB(S)) \) (“crossed subset blossoms”) counts the number of crossings by \( M \), beyond the first, of inherited blossoms \( B \) contained in a shell. Since a blossom \( B \) is odd, each crossing beyond the first decreases the objective of the previous scale by \( z_0(B)/2 \). (Again see Fig. 11. The \( \kappa \) functions are defined right before Lemma 3.2.)

The three objective reducers occur naturally in the fundamental “scaling inequality” (3.4), obtained by comparing the current matching with the matching of the previous scale.

The difficulty of working with these dual objective function reducers is their ephemeral nature: As the matching gets augmented, free vertices as well as blossom crossings may disappear, in essentially arbitrary fashion.

Outline of the analysis section Sect. 3.1 presents our mechanisms for applying objective reducers. Section 3.2 states the scaling inequality (3.4). Section 3.3 presents the main theorem, the bound on the number of phases. It assumes the scaling inequality. Finally Sect. 3.4 completes the analysis by proving the scaling inequality. It derives the bound (3.4) on the size of reducers, by comparing the current scale with the previous one. Readers interested only in 1-matching can proceed directly to Sect. 3.1.

For readers interested in \( f \)-factors please note that most of the analysis in this section applies verbatim to \( f \)-factors. Specifically the scaling inequality for 1-matching, (3.4), has a direct analog for \( f \)-factors, (5.3). That inequality is derived in Sect. 5. Aside from
proving the scaling inequality, the analysis of Sect. 3 is valid for $f$-factors. To allow the reader to verify Sect. 3 for $f$-factors, we introduce a simple oracle that describes the operation of the DISMANTLER. The reader can easily verify that $f$-factor searches, using the algorithm of Gabow [10], are consistent with this oracle.

The oracle for $f$-factors The oracle returns the results of each Edmonds search in the DISMANTLER that obey the following restrictions but are otherwise arbitrary.

The search decreases each value $y(v)$, $v$ free, by 1.

The search does an arbitrary number of augments, possibly 0, along arbitrary augmenting trails (within the current shell).

Inequality (3.4) holds after every search. (5.3) has the term $d(F)$ instead of $d(F)$, and that term is interpreted in the natural way: If a vertex $v \in V$ is free, i.e., it is on $|\delta(v, M)| < f(v)$ matched edges, then it contributes $d(v) \times (f(v) - |\delta(v, M)|)$ to the quantity $d(F)$ in (3.4).

Duals of nonfree vertices may change arbitrarily. So the only numerical information available to the efficiency analysis (as it applies the objective reducers) are the $\tau$ values and the quantities $d(v)$, $v$ free. The oracle does not report any details of blossoms in the search graph. So the credit system cannot use any information about $\Omega_1$ blossoms.

The edges forming an $f$-matching can be viewed in two equally valid ways. In the first we view the edge set as a 1-matching on the graph where each vertex $v$ has been expanded to $f(v)$ copies, each either free or matched to the appropriate other vertex. For example the vertex $\omega$ is one copy of some given $f$-factor vertex. An expression $|A|$, for $A$ a vertex set like a shell or an mpr, etc., is interpreted as $\sum_{v \in A} f(v)$. An inherited blossom is called an e-blossom in Sect. 4, and Lemma 5.1 shows an e-blossom $B$ has odd size $f(B)$. Thus a blossom in this section has odd size $|B|$, a shell $S$ with two blossom boundaries has even size $|S|$, etc.

The second way to view this section, for readers familiar with Sects. 4–5.3, is to simply make the obvious changes in notation and terminology. For instance “matching” is interpreted as “$f$-matching”, the size $|S|$ of a shell $S$ is interpreted as $f(S)$, etc.

3.1 Objective Reducers

We use two mechanisms that apply objective reducers, both based on the following simple structural property.

Proposition 3.1 Consider a shell $S$ with two blossom boundaries. Let $M$ be a matching that crosses exactly one of the boundaries, say blossom $B \in \{S^+, S^\}$). Either $S$ contains a free vertex of $M$ or $|\delta_M(B)| \geq 2$.

Proof For an arbitrary shell $S$, its complement $\overline{S}$ is the disjoint union of $S^-$ and $\overline{S}^+$. Thus $\delta_M(S)$ is the set of $M$-edges crossing exactly one of the boundaries $S^+, S^-$ (this holds even if $M$ is not a matching). Applied to the proposition we get $\delta_M(S) = \delta_M(B)$. Suppose $S$ contains no free vertex. Since $S$ is an even shell, $|\delta_M(S)|$ is even. $B$ crossed implies $|\delta_M(B)| \geq 1$. Thus $|\delta_M(B)| = |\delta_M(S)| \geq 2$. ☐
The proposition will allow us to charge a unit translation to some free vertex or some doubly crossed blossom. For motivation start with the simplest possible case: an execution of DismantlePath(Q) where the shells of \( \mathcal{A} \) never change in Phase 1. Every unit translation can be charged to some \( d(f) \) quantity, i.e., a dual adjustment of a free vertex. We will generalize these unchanging chargeable shells \( \mathcal{A} \) to shells that we call “cores”. As shown in the next lemma, cores charge translations to \( d(f) \) quantities, as above, or to the \( \kappa \) quantities previously introduced. Figure 9 illustrates the following definition.

Let \( S \) be an arbitrary shell of \( P(Q) \). (\( S \) may be even or odd.) Consider the last execution of ShellSearch in Phase 1 or 2 of DismantlePath(Q) whose argument is a subshell of \( S \), if such exists. Let ShellSearch(R) be that execution. Let \( M \) be the matching on \( R \) at that point. A core \( C \) of \( S \) is a shell that is contained in \( R \), contains a free vertex of \( M \), has no boundary crossed by \( M \), and is minimal wrt these conditions.

We make several comments on this definition:

- The core \( C \) of \( S \) exists as long as some subshell of \( S \) gets searched in Phase 1 or 2. This follows since \( R \) is a candidate for the core.
- As an extreme example the shell \((Q, \emptyset)\) of \( P(Q) \) may have no core: If \( Q \)'s only free vertex is \( \omega \), DismantlePath(Q) goes directly to Phase 3. Note cores are not defined in Phase 3, since we shall see their analysis is trivial.
- There may be various choices for \( C \). In that case the choice is arbitrary.
- It is possible that \( S = R \), or \( C = R \), or \( S = R = C \).
- The minimality of \( C \) implies every interior blossom of \( C \) is crossed when ShellSearch(R) is executed.

Note that the notion of an “uncrossed shell” is more permissive than what we are using: A shell with a matched edge from \( D \) to \( V - C \) is uncrossed. But the proof of Lemma 3.2 fails for such a shell. Unlike Gabow and Tarjan [15] we avoid the term “uncrossed shell” since it seems ambiguous and misleading. Our notions of “crossing” are standard and reviewed in the Terminology section.
We introduce more notation associated with a core \(C\) of \(S\). \(Q\) and \(R\) retain their meaning from the definition of core. In the following definitions \(\mathcal{B}\) denotes a set of inherited blossoms and \(\epsilon\) is 0 or 1.

Define

\[
M = \text{the matching on } C \text{ when SHELLSEARCH}(R) \text{ is executed}
\]

\[
F = \text{the set of vertices that are free in } M
\]

\[
INT(S) = \{B : B \text{ an interior blossom of } S\}
\]

\[
\kappa_\epsilon(B) = \sum \{(|\delta_M(B)| - \epsilon) \ z_0(B) / 2 : B \in \mathcal{B}, \ |\delta_M(B)| > \epsilon\}
\]

\[
d(F, C) = \text{the total number of dual adjustments made on vertices } f \in F
\]

from the start of DismantlePath\((Q)\) to the end of SHELLSEARCH\((R)\).

(3.1)

Note \(F \subseteq C\), and matched edges not in \(\gamma(C)\) are irrelevant. Figure 9 illustrates the following lemma.

**Lemma 3.2** Let \(C\) be a core of \(S\). In phases 1 and 2 of DismantlePath\((Q)\), the total number of unit translations in executions SHELLSEARCH\((T)\), where \(T \subseteq S\) and \(T \cap C \neq \emptyset\), is at most \(d(F, C) + \kappa_0(INT(C))\). If we exclude translations of blossoms interior to \(C\) (i.e., \(T^- \in INT(C)\) is not counted, and similarly for \(T^+\)) the number is at most \(d(F, C) + \kappa_1(INT(C))\).

**Proof** It suffices to prove the second bound, where translations of blossoms \(B\) interior to \(C\) are not counted. In proof, every interior blossom \(B\) is crossed, by the minimality of core \(C\). Thus \(B\) contributes exactly \((|\delta_M(B)| - 1)z_0(B)/2\) to the second upper bound, and an additional \(z_0(B)/2\) more to the first upper bound. Since \(B\) is translated at most \(z_0(B)/2\) times, the additional contribution to the first upper bound covers counting the translations of \(B\).

For the second bound we will account for every unit translation of a noninterior \(T\) boundary by exhibiting a corresponding contribution to a \(d\) or \(\kappa_1\) quantity. Assume \(C^- \neq \emptyset\). The opposite case is degenerate will be treated after the main argument. (In fact we will not use this case in the final argument.)

There are two symmetric cases depending on which boundary of \(T\) is noninterior. Remembering that \(T \cap C \neq \emptyset\) the cases are (a) \(T^+\) is not interior, implying

\[
T^- \subseteq C^+ \subseteq T^+,
\]

and (a') \(T^-\) is not interior, implying

\[
T^- \subseteq C^- \subseteq T^+.
\]

(Both possibilities hold when \(T^- \subseteq C^- \subseteq C^+ \subseteq T^+\).)

\(C\) is an even shell with no boundary crossed by \(M\). So \(C\) contains \(\geq 2\) free vertices, say \(f, f' (f, f' \in F)\). Choose \(f\) “closest to” \(C^+\), i.e., for any blossom \(B \subseteq C^+\) interior to \(Q\), \(f \in B\) implies the shell \((C^+, B)\) has no free vertex. Associate \(f\) with
case (a) above. Treat case (a’) symmetrically, i.e., associate it with a free vertex \( f’ \in C \) that is closest to \( C^- \). Clearly this \( f’ \) can be chosen distinct from \( f \). We now analyze case (a). Here the goal is to pay for every translation of \( T^+ \). The analysis of case (a’) is symmetric. For a shell \( T \) satisfying both cases we need only account for translations of \( T^+ \), leaving \( T^- \) to case (a’).

**Case \( f \in T^- \):** Case (a) shows \( f \in T^- \subseteq C^+ \). This makes \( T^- \) interior to \( C \). The choice of \( f \) implies \((C^+, T^-)\) has no free vertex. Since \( C^+ \) is uncrossed and \( T^- \) is crossed, \(|\delta_M(T^-)| \geq 2 \) (Proposition 3.1). Thus \( T^- \) contributes \( \geq z_0(T^-)/2 \) to \( \kappa_1(INT(C)) \). We assign this contribution to the various blossoms \( T^+ \) that were translated in a \( SHELLSEARCH \) with the fixed blossom \( T^- \). There are \( \leq z_0(T^-)/2 \) such translations so they are all accounted for.

**Case \( f \notin T^- \):** Case (a) shows this places \( f \) in the shell \( T \). \( f \) is free when \( SHELLSEARCH(T) \) is executed. (In proof observe the definition of \( R \) along with \( T \subseteq S \) makes \( T \subseteq R \). Furthermore \( f \) is free in \( SHELLSEARCH(R) \).) So the translation of the various blossoms \( T^+ \) are counted in \( d(f, C) \) (which contributes to \( d(F, C) \)).

Note a degenerate possibility for case (a’) is that \( T^- = \emptyset \), which has no translation for \( T^- \). This completes the argument for \( C^- \) a blossom.

We turn to the case \( C^- = \emptyset \). \( C \) has at least one free vertex, so vertex \( f \) of the previous case exists. Thus the argument for case (a), \( C^+ \subseteq T^+ \), is unchanged. The other possibility is \( T^+ \subseteq C^+ \), i.e., \( T^+ \) interior to \( C \). This case is vacuous. \( \square \)

The lemma allows us to account for every unit translation made in phase 1 or 2 of \( DISMANTLEPATH(Q) \), as follows: We use a recursive procedure that accounts for all unit translations in phase 1 or 2 within a given shell \( S \) or \( T \). Start by applying the lemma to a core \( C \) of \( S \). It accounts for all translations made in executions \( SHELLSEARCH(T), T \subseteq S \), unless \( T \) is disjoint from \( C \). Account for these by recursively applying the procedure to the two shells \((S^+, C^+), (C^-, S^-)\). We call this the core decomposition of \( S \).

The cores in this decomposition may be defined by matchings that occur at different moments in the execution of the Dismantler. For example the matching defining the core of \( S \) may have no free vertices in \((S^+, C^+)\). That shell’s cores are defined by a matching occurring earlier in the execution of \( DISMANTLEPATH(Q) \). In fact our final bound on the translations made in an execution \( DISMANTLEPATH(X) \) will use matchings occurring at various different times in various different executions of \( DISMANTLEPATH(Q), Q \) ranging over the mprs contained in \( X \).

This transition necessitates use of a second mechanism for applying objective reducers. (In detail, one could account for every unit translation ever made by the Dismantler by using the core decomposition of shell \( (Q, \emptyset) \) for every mpr \( Q \). This choice leads to a poor upper bound, because of recursive applications of the (recursive) core decomposition!) The second mechanism is actually a simple variation of Lemma 3.2, which we now present.

To set the stage consider a moment in time in the execution of \( DISMANTLEPATH(X) \) (“X” for “executing”). We will be required to account for certain unit translations made previously, in executions \( DISMANTLEPATH(Q), Q \subseteq X \). Specifically the required translations are those made for blossoms \( B \in P(Q) \) that are uncrossed by
the current matching on $X$. (These translations correspond to the term $\tau(U)$ in Lemma 3.4 below.) For an intuitive explanation recall the opening discussion of objective reducers: Every crossing of a blossom reduces the dual objective. So only uncrossed blossoms are problematic.)

To describe the situation in detail consider an mpr $Q \subset X$. Let $M$ be the set of edges that are matched at the chosen point in DismantlePath($X$), and have one or both vertices in $Q$. (Edges of $M$ may cross $Q$.) In what follows all references to “crossing” and “uncrossing” refer to $M$. Let $F$ be the set of vertices of $Q$ unmatched by $M$. Clearly $F \subseteq Q$ is a subset of the vertices that are free at the end of DismantlePath($Q$). Let $B_\omega$ and $B_\alpha$ be the minimal and maximal uncrossed blossoms of $P(Q)$, respectively. Decompose $P(Q)$ into the minimal shells with uncrossed boundaries, specifically $(B_\omega, \emptyset)$;

minimal shells $S$ with two blossom boundaries $S^-, S^+$ that are both uncrossed (i.e., every interior blossom of $S$ is crossed);

$(Q, B_\alpha)$ if $Q$ is crossed.

It is possible that no uncrossed blossom exists. In the opposite case $B_\omega$ and $B_\alpha$ exist, although we may have $B_\omega = B_\alpha$.

Say that a shell $S$ of $P(Q)$ with two uncrossed blossom boundaries is void if among all such shells, $S$ is maximal wrt the condition $S \cap F = \emptyset$. Observe that a void shell is perfectly matched. This holds since an edge crossing $S$ must cross either $S^+$ or $S^-$, so it cannot be matched. Also note that an interior blossom of $S$ may be crossed or uncrossed.

Define

$$d(F, Q) = \text{the total number of dual adjustments made on vertices } f \in F$$

in the execution of DismantlePath($Q$). \hfill (3.2)

Although the notation $d(F, Q)$ has the potential to conflict with $d(F, C)$ from (3.1) this will never be the case: Context will always make clear whether the second argument is an mpr or a core. Also whenever possible we use $Q$ or $C$ as the second argument (if not a close relative). Finally when there is any danger of confusion we explicitly indicate which function is being used.

The following lemma bounds all relevant unit translations except those contained in void shells. It is illustrated in Fig. 10a.

**Lemma 3.3** The total number of unit translations of blossoms $B \in P(Q)$ uncrossed by $M$, made in invocations SHELLSEARCH($T$) for $T$ not contained in a void shell, is at most $d(F, Q) + \kappa_1(|\{B : B \in P(Q) \text{ not interior to a void shell}\}|)$.

**Proof** Consider an execution of SHELLSEARCH($T$) in DismantlePath($Q$). The lemma is vacuous if both boundaries $T^-, T^+$ are crossed. So the following three cases exhaust all possibilities:

*Case $T$ has exactly one uncrossed blossom boundary, and $T \cap F \neq \emptyset$: Take any $f \in T \cap F$. $f$ is free in DismantlePath($X$) so it is free in DismantlePath($Q$). So the unit translation of SHELLSEARCH($T$) decreases $y(f)$, and is counted in $d(F, Q)$.*
Note this case allows $T$ to have two blossom boundaries or only one. The second possibility, i.e., $T$ an odd shell $(T^+, \emptyset)$, is always covered by this case. In proof $T^+$ uncrossed and the minimality of $B_\omega$ implies $B_\omega \subseteq T^+$. Thus $\omega \in T^+$.

**Case $T^-$** has exactly one uncrossed blossom boundary, and $T \cap F = \emptyset$: Let $T^+$ be uncrossed. (The case $T^-$ uncrossed is symmetric.) $T$ has two blossom boundaries (as noted in the previous case). $T^-$ is crossed and $T$ has no free vertex. Thus $|\delta_M(T^-)| \geq 2$ (Proposition 3.1). If $T^-$ is interior to a void shell $V$ then $T \subseteq V$ (by the maximality of $V$). If $T^-$ is not interior it contributes $\geq z_0(T^-)/2$ to the lemma’s $\kappa_1$ term. This accounts for the translation of $T^+$. ($T^-$ was translated in SHELLSEARCH($T$) and there are $\leq z_0(T^-)/2$ such translations. Thus $T^-$ is not overcharged.)

An example of this case is the shell $(Q, B_\alpha)$ when it contains no free vertex. Figure 10a gives an example.

**Case $T^-$ and $T^+$** are both uncrossed blossoms: $T$ is an even shell. So if $T$ contains a free vertex, it has at least two, say $f$, $f' \in F$. Both vertices are free in SHELLSEARCH($T$). So $y(f)$ and $y(f')$ get decremented. This contributes 2 to $d(F, Q)$, paying for the 2 unit translations of SHELLSEARCH($T$). If $T$ has no free vertex then $T$ is contained in a void shell.

The first shell $(B_\omega, \emptyset)$ in the above decomposition of $P(Q)$ is a special case—it is the only odd shell. Vertex $\omega$ is in this shell, and the validity of the lemma requires that $\omega \in F$ (in the first case of the proof, $d(\omega)$ can be charged for a translation with $T^- = \emptyset$). We will explicitly verify that $\omega \in F$ in our applications of the lemma. This verification is done in the proofs of (3.3) below and Lemma 3.4.

The last two lemmas combine to give a complete accounting for the unit translations of uncrossed blossoms in DISMANTLEPATH($Q$). (We exclude Phase 3.) To give a precise statement we expand previous notation to include the point in time that is being referenced, e.g., $M_X$ denotes the current matching in DISMANTLEPATH($X$). Similarly the notation $\kappa_e(\cdot, M)$ makes the matching in the $\delta_M$ term of $\kappa_e$ explicit.

Consider an arbitrary point in the execution of DISMANTLEPATH($X$). Let $M_X$ denote the current matching on $Q \cup \delta(Q)$, with $F_X$ the subset of free vertices in $Q$. Let $C$ denote the collection of all sets in the core decompositions of the various void shells of $P(Q)$. For $C \in \mathcal{C}$, $M_C$ denotes the matching on $C$ at the time in DISMANTLEPATH($Q$).
that defines $C$ as a core. $F_C \subseteq C$ denotes its set of free vertices. The total number of unit translations of uncrossed blossoms $B \in P(Q)$ is at most

$$d(F_X, Q) + \kappa_1(B : B \in P(Q) \text{ not interior to a void shell of } Q, M_X) + \sum_{C \in C}(d(F_C, C) + \kappa_0(INT(C), M_C)). \quad (3.3)$$

Note the uncrossed blossoms and the void shells referenced here are determined by the chosen point in DISMANTLEPATH($X$), i.e., by $M_X$.

To prove (3.3), the first line constitutes the bound of Lemma 3.3, which is computed during the execution of DISMANTLEPATH($X$). In particular note that the definition of $F_X$ includes the free vertex $\omega$, as required for Lemma 3.3. (Recall shell $(B_\omega, \emptyset)$.)

The second line is the bound of Lemma 3.2 for the void shells of $P(Q)$. The cores of $C$ are defined at various points in time in the execution of DISMANTLEPATH($Q$). Note that a void shell of $Q$ may not have a core decomposition, but only when Lemma 3.3 shows there are no unit translations to pay for.

3.2 The Scaling Inequality

As usual the framework of the analysis is provided by an inequality that shows each scale begins with a close-to-optimal structure. This section states that scaling inequality. Section 3.3 shows how the inequality is used to prove the main bound on $d(F)$. Section 3.4 completes the analysis by proving the scaling inequality.

We begin by reiterating the setting: The recursive accounting procedure of Sect. 3.1 requires analysis of an arbitrary shell (or core) $S$ of a major path $P(Q)$, at any point in the execution of the DISMANTLER. Define $X$ (mnemonic for “executing”) to be the mpr where DISMANTLEPATH($X$) is currently executing. There are two qualitatively different cases, $X = Q$ and $X \supset Q$ (i.e., DISMANTLEPATH($Q$) is either currently executing or has returned). At various points in the derivation we will comment about differences between the two cases.

To state the inequality let $S$ be an even shell of a major path $P(Q)$. Consider an arbitrary point in time during the execution of DISMANTLEPATH($X$) for $X \supset Q$. Assume the current matching on $X$ does not cross a boundary of $S$; let $M \subseteq \gamma(S)$ be the current matching restricted to $S$. Let $F \subseteq S$ be the set of free vertices of $M$. The scaling inequality is

$$d(F) + \kappa_0(INT(S)) + \kappa_1(SUB(S)) \leq cn + \tau(U), \quad (3.4)$$

where the various terms are defined as follows: Let $v$ be a free vertex. Let $B \subseteq Q$ be an arbitrary inherited blossom ($B$ may be contained in an mpr $Q' \subset Q$). Define $INT$ and $\kappa_\epsilon$ as in (3.1) and define

$$d(v) = \text{the number of Phase 1 or 2 dual adjustments made for } v,$$

up to and including the chosen point in DISMANTLEPATH($X$)

$$\tau(B) = \text{the number of Phase 1 or 2 unit translations made on } B,$$

up to and including the chosen point in DISMANTLEPATH($X$)
\[ SUB(S) = \{ B : B \subset S \} \]
\[ c = 4 \]
\[ n = |S| \]
\[ U = \{ B : B \subset S \text{ not crossed by } M \}. \] (3.5)

Here \( M \) is used to define the two crossed blossom functions \( \kappa_e \), and \( n \) is the number of vertices of shell \( S \).

We comment on several aspects of this definition. Fundamentally note that the inequality is consistent with the intuitive discussion of objective reducers at the start of the section. In particular the left-hand side of (3.4) corresponds to the current reduction in the value of the dual objective function.

Equation (3.4) requires \( S \) to be an even shell. We account for odd shells by taking \( S^+ \) to be a vertex \( \omega \). The assumption that \( M \) does not cross \( S^+ \) requires \( \omega \) to be free. Such a free vertex exists since \(|S^+|\) is odd. Note that by definition \( \omega \notin F \) in (3.4). So applications of (3.3) require that the case of odd shells be avoided.

The set \( F \) will be empty when \(|S^+ - V(M)| = 1\). \( F \) may also be empty when \( S^- \) is a blossom.

Recall that our summing convention (Terminology section) defines \( d(F) \) as \( \sum_{f \in F} d(v) \). Note also the quantities \( d \) and \( \tau \) ignore Phase 3. We prove Phase 3 contributions are irrelevant below. (Intuitively, a unit translation in Phase 3 does not increase the dual objective function, since it is undone by the immediately following dual adjustment of the unique free vertex.)

### 3.3 The Dismantler’s Time Bound

This section proves Theorem 3.7, the time bound for the entire algorithm, assuming the scaling inequality. The crucial analysis is accomplished in Lemmas 3.4 and 3.5, which prove the \( \sqrt{n \log n} \) bound on the number of passes made in DismantlePath. We start with an overview of our strategy for these two lemmas. Then we give their formal proof. We conclude the section with the theorem.

We shall see the number of passes can be bounded by applying (3.3), so we concentrate on that. The upper bound on passes must be a function of \( n \) alone. So the bound of (3.3) must be converted to a function of \( n = |Q| \). This is automatic for the first line of (3.3): The goal of upper-bounding the unit translations of uncrossed blossoms comes from a previous application of the scaling inequality (3.4), specifically from its \( \tau(U) \) term. That inequality also upper-bounds the terms of the first line of (3.3) by \( cn \).

We use the same strategy for the second line: Apply the scaling inequality (3.4) to each core \( C \in \mathcal{C} \). The left-hand side gives a bound on the terms of the second line. Consider the right-hand side. Its \( cn \) term (now \( n = |C| \)) contributes to the required function of \( n \). Its \( \tau(U) \) term gives a new set of uncrossed blossoms that must be upper-bounded! More precisely let \( \mathcal{U}_C \) denote the set of blossoms uncrossed by \( M_C \). The \( \tau(U_C) \) term in (3.4) can be bounded by applying the entire strategy recursively, i.e., each core \( C \) now plays the role of \( X \). The formal analysis will use a system of credits and debits to help track the recursive calls.
To present the formal analysis we start with notation. For any mpr \( Q \) define
\[
\hat{Q} = \{ Q' : Q' \text{ an mpr properly contained in } Q \}.
\]

Note the expression \( Q + \hat{Q} \) simply adds \( Q \) to the set of mprs. (The expression is interpreted using our notational convention for singleton sets.) The set of all inherited blossoms contained in mpr \( Q \) is \( P(Q + \hat{Q}) \).

We also extend the \( d \) function, as follows. For any mpr \( Q \) and any vertex \( v \), \( d(v, Q) \) denotes the contribution of \( v \) to the quantity \( d(F, Q) \) in (3.2), i.e., the number of dual adjustments made on \( v \) during DismantlePath(\( Q \)) while \( v \) is free. We allow the possibility \( v \notin Q \), in which case obviously \( d(v, Q) = 0 \). Our summing conventions (Terminology section) imply \( d(F, Q + \hat{Q}) = \sum \{ d(v, Q') : v \in F, \text{ mpr } Q' \subseteq Q \} \) and also
\[
d(F, Q + \hat{Q}) = d(F, Q) + d(F, \hat{Q}).
\]

We begin the analysis. First Lemma 3.4 discusses the case \( Q \subseteq X \). Then Lemma 3.5 discusses \( Q = X \).

To state the upper bound for \( Q \subseteq X \) consider an mpr \( Q \). Let \( M \) be the algorithm’s matching at a chosen point in time in the execution of DismantlePath(\( X \)), \( X \supseteq Q \). \( M \) may cross blossoms of \( Q \) arbitrarily. Define
\[
\mathcal{U} = \{ B : B \subseteq Q \text{ is inherited and uncrossed by } M \}.
\]

Let \( F \) be the set of free vertices of \( M \).

Let \( Q \) have height \( h \) in \( Q + \hat{Q} \), i.e., the longest chain of mprs contained in \( Q \) has \( h \) mprs. So \( h = 1 \) means \( Q \) is a minimal mpr, i.e., every inherited blossom \( B \subseteq Q \) belongs to \( P(Q) \).

**Lemma 3.4**
\[
\tau(\mathcal{U}) \leq d(F, Q + \hat{Q}) + \kappa_1(P(Q + \hat{Q})) + c|Q|h.
\]

*Here the matching \( M \) on \( X \) is used to define \( \mathcal{U}, F, \) and \( \kappa_1 \).*

**Proof.** The proof is by induction on \( h \). Let \( R \) range over the maximal mprs properly contained in \( Q \). A blossom of \( \mathcal{U} \) belongs to either \( P(Q) \) or some set \( P(R + \hat{R}) \).

View the quantity \( \tau(\mathcal{U}) \) as a debt and the upper bound of the lemma as a collection of credits. One credit can pay for one unit translation of an uncrossed blossom. We will show the lemma’s credits can pay for every unit translation of an uncrossed inherited blossom.

Distribute the credits of the upper bound as follows. Distribute the credits for each free vertex \( f \) according to the identity \( d(F, Q + \hat{Q}) = d(F, Q) + \sum_{f,R} d(f, R + \hat{R}) \), i.e., the occurrence of \( f \) in \( Q \) (respectively \( R \)) gets \( d(f, Q)(d(f, R + \hat{R})) \) credits. For each blossom \( B \) contributing to \( \kappa_1(P(Q + \hat{Q})) \) distribute the corresponding credits, numbering \( (|\delta_M(B)| - 1)z_0(B)/2 \), to either \( Q \) if \( B \in P(Q) \) or to the mpr \( R \) where \( B \subseteq R \). Say the term \( c|Q|h \) gives *vertex credits*. Keep \( c|Q| \) vertex credits on \( Q \) and give \( c|R|(h - 1) \) vertex credits to each \( R \). To help keep track of credits in this proof note that the mprs \( R \) will be treated in two different ways, depending on whether or not \( R \cap F = \emptyset \).
Each mpr $R$ has height $\leq h - 1$. The inductive hypothesis shows

$$
\tau(U \cap P(R + \hat{R})) \leq d(F, R + \hat{R}) + \kappa_1(P(R + \hat{R})) + c|R|(h - 1) \quad (3.6)
$$

where $M$ is used to define $U$, $F$, and $\kappa_1$. We pay for the right-hand side using the credits on $R$ (these credits exactly match the debt on the right-hand side). Figure 10b gives an example. Note that we can assume $R \cap F \neq \emptyset$. For the opposite implies every blossom $B \subseteq R$ is crossed (since a blossom is an odd set). That implies $R$ does not contain any set of $U$, so $R$ does not contribute to the left-hand side of the lemma.

It remains only to pay for the uncrossed blossoms of $P(Q)$. We do this using (3.3). The distributed credits pay for the terms of the first line of (3.3) (with unused credits for crossed blossoms $B$ that are interior to a void shell). In particular the set $F$ of the lemma contains vertex $\omega$, as required in the first line of (3.3).

For the second line of (3.3) consider a core $C \in \mathcal{C}$. Apply the scaling inequality (3.4) to $C$ right after the dual adjustment (made in SHELLSEARCH) that defines $C$:

$$
d(F_C) + \kappa_0(INT(C), M_C) + \kappa_1(SUB(B(C), M_C)) \leq c|C| + \tau(\mathcal{U}_C). \quad (3.7)
$$

Here $\mathcal{U}_C$ is the set of inherited blossoms properly contained in $C$ and not crossed by $M_C$. For the $d$ term recall $F_C$ is the set of free vertices in $M_C$, and $d(F_C)$ counts every dual adjustment up to the time $C$ is defined. Thus

$$
d(F_C) = d(F_C, C) + d(F_C, \hat{Q}), \quad (3.8)
$$

where the second term counts the dual adjustments made before the execution of DISMANTLEPATH($Q$) that defines $C$.

View the left-hand side of (3.7) as new credits, and the right-hand side as a new debt to pay for these credits. Using (3.8) write the left-hand side as

$$
\left(d(F_C, C) + \kappa_0(INT(C), M_C)\right) + \left(d(F_C, \hat{Q}) + \kappa_1(SUB(B(C), M_C))\right). \quad (3.9)
$$

The first line exactly pays for $C$’s term in (the second line of) (3.3). It only remains to pay for the new-debt terms of (3.7). The term $c|C|$ summed over all cores is at most $c|Q|$. The vertex credits on $Q$ pay this debt. Consider the new debt $\tau(\mathcal{U}_C)$. The blossoms of $\mathcal{U}_C$ are contained in the various maximal mprs $R \subseteq C$. Recall from (3.3) that the cores are contained in void shells. Thus each such $R$ has $R \cap F = \emptyset$. This ensures that all the credits on $R$ are still available. It also implies that $F_C$ in (3.7) does not exclude any vertex of $C$ (i.e., $C$ is not an odd shell).

Now consider two cases.

**Case $h = 1$:** (This is the base case of the induction.) $Q$ is a minimal mpr so $SUB(C)$ is empty, as is $\mathcal{U}_C$. So there is no new debt.

**Case $h > 1$:** From the second line in (3.9) give $R d(F_C, R + \hat{R})$ credits from free vertices and $(|\delta_{M_C}(B)| - 1)z_0(B)/2$ credits for blossoms $B \subseteq R$. Apply the inductive
hypothesis to $R$. This is valid since as noted above, $F_C$ does not exclude any free vertex of $C$. The application gives (3.6), assuming we change “$\mathcal{U}$” on the left-hand side to “$\mathcal{U}_C$” and use $M_C$ to define $\mathcal{U}_C$, $F$, and $\kappa_1$. The debt on the right-hand side is paid for by these new credits, plus the $c|R|(h - 1)$ vertex credits originally given to $R$. Figure 10c gives an example. □

We turn to the $Q = X$ case. Consider the execution of DISMANTLEPATH($X$) at an arbitrary point in Phase 1 or 2. $X$ is an arbitrary mpr, possibly $X = V$. Let $M_X$ be the current matching on $X$. Let $F_X$ be its set of free vertices where, if $X \neq V$, we exclude $\omega$, a free vertex in the smallest possible blossom $B_\omega \in P(X)$. (This contrasts with the treatment of $\omega$ in Lemma 3.4.) Note that $d(F_X, X)$ counts all dual adjustments on free vertices in DISMANTLEPATH($X$) up to the chosen moment.

**Lemma 3.5** $d(F_X, X) \leq c|X|\log |X|$.

**Proof** Let $X$ have height $h$ in the set of major path roots ($h < \log |X|$). Let $S$ be an even shell of $P(X)$ that contains a vertex of $F_X$ and has no boundary crossed by $M_X$. $S$ may even be a currently dissolved shell of $P(X)$. Also $S$ may be $(B_\omega, \omega)$ if it contains a vertex of $F_X$. Let $F$ denote $F_X \cap S$. Equation (3.4) gives

$$d(F) + \kappa_0(INT(S)) + \kappa_1(SUB(B(S)) \leq c|S| + \tau(\mathcal{U}). \quad (3.10)$$

$M_X$ is used to define all matching-dependent terms, i.e., $F$, the $\kappa_\epsilon$, and $\mathcal{U}$.

Apply Lemma 3.4 to each maximal mpr $Q \subset S$ and sum the inequalities to get

$$\tau(\mathcal{U}) \leq d(F, \hat{X}) + \kappa_1(SUB(B(S)) + c|S|(h - 1). \quad (3.11)$$

$M_X$ again defines all matching-dependent terms. Combining (3.10) with (3.11) and recalling $d(F) = d(F, X) + d(F, \hat{X}))$ gives

$$d(F, X) + \kappa_0(INT(S)) \leq c|S| + c|S|(h - 1) = c|S|h. \quad (3.12)$$

Adding (3.12) for all shells $S$ (and ignoring interior blossoms of $S$) gives $d(F_X, X) \leq c|X|h$ proving the lemma. □

**Bounding the number of passes** Let $F_p$ denote the subset of free vertices $F_X$ that are still active at the end of the $p$th pass of Phase 1. (So $F_p$ excludes free vertices in dissolved shells, as well as $\omega$.) The $p$ passes collectively perform $\geq p - \log |X|$ dual adjustments on every vertex of $F_p$. So the lemma implies

$$|F_p|(p - \log |X|) \leq c|X|\log |X|. \quad (3.13)$$

Define the parameter $\pi$ of Phase 1 (Fig. 5),

$$\pi = c\sqrt{|X|\log |X|} + 1. \quad (3.14)$$

Suppose pass $p$ is not the last pass of Phase 1. Recalling vertex $\omega$ implies $|F_p| > \pi - 1 = c\sqrt{|X|\log |X|}$. Thus (3.13) implies $p < \sqrt{|X|\log |X|} + \log |X| \leq 2\sqrt{|X|\log |X|}$. Springer
So Phase 1 uses total time $O(\sqrt{|X|} \log |X| \, m(X))$. (The notation $m(X)$ views $X$ as a subgraph so $m(X)$ is the number of its edges.)

Phase 2 begins with at most $\pi$ free vertices in active shells. So Phase 2 finds $O(\sqrt{|X|} \log |X|)$ augmenting paths. As mentioned we implement Edmonds’ algorithm using a straightforward bucket-based priority queue, say $\mathcal{P} \mathcal{Q}$, for the search step events. The algorithm steps through each time unit of $\mathcal{P} \mathcal{Q}$, invoking $\text{ShellSearch}(S)$ to execute the Edmonds steps (if any) scheduled for that time. The time for all Edmonds steps to find one augmenting path is $O(m(X))$. So these steps use total time $O(\sqrt{|X|} \log |X| \, m(X))$, as desired. It remains to bound the number of time slots in $\mathcal{P} \mathcal{Q}$. Since the $\text{ShellSearch}$ of each time unit performs 2 unit translations, this amounts to bounding the total number of unit translations in Phase 2. The next lemma shows the number of time slots is $O(|X| \lg |X|)$. So the extra time for $\mathcal{P} \mathcal{Q}$ is irrelevant. It is also simple to implement $\mathcal{P} \mathcal{Q}$ using $O(|X|) = O(n)$ space.

Let $S$ be a shell of $P(X)$. For instance $S$ can be $(X, \emptyset)$. Define

$$\tau(S) = \text{the total number of unit translations made in Phase 1-2 searches of } P(X)\text{-shells } R \subseteq S.$$ 

**Lemma 3.6** $\tau(X, \emptyset) \leq |X| \log |X|$.

**Proof** Let $S$ be an arbitrary shell of $P(X)$. Let $R$ be the last subshell of $S$ that is searched in Phase 1 or 2, and $C$ its core. Let $F$ be the set of free vertices in $C$ when $R$ is searched for the last time. Lemma 3.2 and the core decomposition of $S$ translates to

$$\tau(S) \leq \kappa_0(INT(C)) + d(F, R) + \tau(C^-, S^-) + \tau(S^+, C^+). \tag{3.15}$$

Let $h$ be the height of $X$. Equation (3.12) applied to the core $C$ gives $d(F, R) + \kappa_0(INT(C)) \leq c|C|h$. Hence we have the recurrence $\tau(S) \leq c|C|h + \tau(C^-, S^-) + \tau(S^+, C^+)$. It easily solves to $\tau(S) \leq c|S|h$ by induction on $|S|$. Taking $S = (X, \emptyset)$ gives the lemma. □

We conclude $\text{DismantlePath}(X)$ uses total time $O(\sqrt{|X|} \log |X| \, m(X))$.

The blossoms $X$ for mpr’s with $2^{l-1} < |X| \leq 2^l$ are vertex disjoint. So $\text{DismantlePath}$ for all of these blossoms uses total time $O(\sqrt{2^l} \, m)$. Summing over all $i \leq \log n$ gives total time $O(\sqrt{n} \log n \, m)$ for each scale, as desired.

**Theorem 3.7** *The scaling algorithm (using the Dismantler of Fig. 4 for each scale) finds a maximum weight perfect matching in time $O(\sqrt{n} \log n \, m \log nW)$.* □

### 3.4 Proof of the Scaling Inequality

#### 3.4.1 Preparation for the Proof

Consider an mpr $X$. (This includes $X = V$.) We will analyze an arbitrary point in the execution of Phase 1 or Phase 2 of $\text{DismantlePath}(X)$. Recall the time for Phase 3
(one search of Edmonds’ algorithm) is not an issue. But the analysis must track how Phase 3 influences Phases 1–2, if at all. Note that a blossom that dissolves before Phase 3 has \( \tau(B) = z_0(B)/2 \). A blossom that dissolves during Phase 3 has \( \tau(B) < z_0(B)/2 \). (And \( B = V \) never dissolves.)

Use all the notation of (3.4), e.g., \( mpr_Q \) contained in \( mpr_X \) at the chosen point in the execution of \( \text{DismantlePath}(X) \); even shell \( S \) of \( P(Q) \); no currently matched edge crosses a boundary \( S^- \) or \( S^+ \); current matching \( M \) on \( S \) (so \( M \subseteq \gamma(S) \)); \( F \) the set of free vertices of \( M \) (\( F \subseteq S \)). We sometimes use the notation \( S = (C, D) \) (as usual \( D \) may be a vertex). See Fig. 11 for an illustration of \( S \).

Although we assume \( M \) does not cross a boundary of \( S \), this does not prevent an \( \Omega \)-blossom \( B \) from crossing a boundary of \( S \): \( B \) can cross arbitrarily many times on unmatched edges. This can occur when \( S \) is a core of \( Q \). Similarly \( B \) may contain \( M \)-matched edges in void shells, but \( B \) crosses a void shell only on unmatched edges.

\( y_0 \) and \( z_0 \) denote the dual functions immediately after scaling up, i.e., when \( \text{Dismantler} \) begins. \( y \) and \( z \) denote the current duals in the execution of \( \text{DismantlePath}(X) \).

We begin the analysis with several preliminary properties, (3.16)–(3.18). Every free vertex \( v \) satisfies

\[
y(v) = y_0(v) - d(v) + \sum_{v \in B} \tau(B). \tag{3.16}
\]

In proof note that whenever \( v \) is the free vertex of Phase 3, \( y(v) \) does not change, since every unit translation that increments \( y(v) \) is offset by the following dual adjustment that decrements it.

Next we show that every \( B \in \Omega^- \) with \( B \cap S \neq \emptyset \) satisfies

\[
B \subseteq S \text{ or } D \subseteq B. \tag{3.17}
\]

(This includes the possibility \( D = \omega \), where the second condition is equivalent to \( \omega \in B \) by our notational convention for singletons. Note that such a blossom \( B \) may belong to some \( P(Q) \). This cannot hold when boundary \( D \) is a blossom.)

First assume \( S \neq (V, \emptyset) \) and consider the blossom tree \( T \): \( B \) is either (i) a proper ancestor of \( C \), or a descendant of \( C \) that is either (ii) a proper ancestor of \( D \), or (iii) a nondescendant of \( D \). (i) and (ii) imply \( D \subseteq B \). (iii) implies \( B \subseteq S \). We have shown that \( B \) satisfies exactly one of the two alternatives. When \( S = (V, \emptyset) \) both alternatives trivially hold although we will only use the first one.

The current dual function \( z \) can be nonzero for two types of blossoms \( B \): \( B \in \Omega \) or \( B \in \Omega^- \) either interior to \( (X, \emptyset) \) or containing \( X \). In more detail any \( B \in \Omega^- \) has

\[
z(B) = \begin{cases} 
0 & B \subseteq Q \subseteq X \text{ or } \omega \in B \subseteq Q' \text{ for some } Q' \subset Q \\
z_0(B) - \tau(B) & B \text{ interior to } X \\
z_0(B) & B \supseteq X.
\end{cases}
\]
The analysis will use the dual objective function of the shell $S$. Recall this function, denoted $\widehat{yz}[S]$, is defined as
\[
\widehat{yz}[S] = y(S) + \sum_B z(B) \lfloor |S \cap B|/2 \rfloor.
\]

When $y, z$ are standard LP duals $\widehat{yz}[S]$ is an upper bound on the weight of a perfect matching on $S$.

Our analysis will use $\widehat{yz}[S]$ (even though we use near-optimum duals). We now show for the current matching $M \subseteq \gamma(S)$ the summation term in the dual objective has
\[
\lfloor |S \cap B|/2 \rfloor = |\gamma_M(B)| + \begin{cases} 0 & B \in \Omega \\ |F \cap B|/2 & B \in \Omega^- \text{ is undissolved.} \end{cases} \tag{3.18}
\]

Note that the case \(B \in \Omega^- \text{ dissolved}\) is irrelevant: $B$ dissolves when $z(B)$ decreases to 0. At that point $B$ does not contribute to the dual objective and so is irrelevant. Later an $\Omega$-blossom $B'$ with the same vertex set as $B$ may be formed, and its dual $z(B')$ may become positive. $B'$ is treated in the first case of (3.18).

To prove (3.18), recalling $S = V(M) \cup F$ gives
\[
|S \cap B| = 2|\gamma_M(B)| + |\delta_M(B)| + |F \cap B|. \tag{3.19}
\]
(The terms on the right-hand side correspond to the edges of $M$ that contain either 2, 1 or 0 vertices of $B$.)

First suppose $B \in \Omega$. $B$ is made up of matched edges and a base vertex $b$. $b$ either is not in $S$, or is free, or is matched to a vertex in $S - B$. In all three cases (3.19) has $|\delta_M(B)| + |F \cap B| \leq 1$. This implies (3.18). (Note this sum is 0 when $D = \omega$ and $\omega \in B$.)

Now suppose $B \in \Omega^-$ and $B$ is undissolved at the chosen instant of time. (So $Q = X$.) The latter implies no matched edge crosses $B$, $|\delta_M(B)| = 0$. Also $B$ undissolved implies either $B$ is interior to $S$ or $S \subseteq B$. In both cases $|S \cap B|$ is even. This implies (3.18). (Note that $|F \cap B|$ may be arbitrarily large.)

### 3.4.2 The Analysis

We will compare the current matching $M$ to a matching $M_\omega$, that is a perfect matching of $S$ that comes from the previous scale. If $S = V$ then $M_\omega$ is the previous scale’s perfect matching of $V$. Suppose $S \subseteq V$. Recall that in general, for any blossom $B$ any vertex $v \in B$ can be chosen as the base. Furthermore $v$ is the base of every blossom $B' \subseteq B$ with $v \in B'$. Take $B$ to be $C$ and $v$ to be any vertex in $D$ that is free in $M$. ($v$ exists since $M$ does not cross $D$.) Let $M_\omega$ denote the matching of $C$ with base vertex

---

4 The square brackets help to distinguish the dual objective from a function evaluation $\widehat{yz}(T) = \sum_{e \in T} \widehat{yz}(e)$. Observe $S$ is a set of vertices and $T$ is a set of edges.
$v = \omega$. $M_\omega$ is a perfect matching of both $C - v$ and $D - v$. Thus $M_\omega$ is a perfect matching of the shell $(C, D)$.

The argument consists of 5 steps. The first four apply the two dual functions, successively using near tightness of $\hat{y_0z_0}$, near domination of $\hat{y_0z}$, near tightness of $\hat{y_0z}$, and near domination of $\hat{y_0z_0}$. The 5th step analyzes how $M$ crosses the blossoms of $\Omega^-$.

Start the first step, which concerns duals $y_0, z_0$, by observing

$$\hat{y_0z_0}(M_\omega) = \hat{y_0z_0}[S].$$

This follows since $M_\omega$ is a perfect matching on $S$, and any blossom $B \in \Omega^-$ contains exactly $\lfloor |S \cap B|/2 \rfloor$ edges of $M_\omega$. For the latter, (3.17) implies when $S \cap B \neq \emptyset$

$$|y_{M_\omega}(S \cap B)| = \begin{cases} \lfloor |B|/2 \rfloor & B \subset S \\ |S \cap B|/2 & B \text{ interior to } S \text{ or } \omega \in B \subseteq Q' \text{ for some } Q' \subset Q \\ |S|/2 & C \subseteq B. \end{cases}$$

(3.20)

All three quantities equal $\lfloor |S \cap B|/2 \rfloor$. (Note the alternative involving $\omega$ is irrelevant if $S^-$ is a blossom, since that makes $S \cap B = \emptyset$. If $S^- = \omega$ and the alternative holds then $S \cap B = B - \omega$ so $|S \cap B|$ is even as required.)

Any edge $e \in M_\omega$ was nearly tight in the previous scale, since it was a blossom edge (or a matched edge of $S = V$). Thus in the new scale $w(e) \geq \hat{y_0z_0}(e) - 4$ (recall scaling up operation $y(v) \leftarrow 2y(v) + 2$). Summing these $n/2$ inequalities gives

$$w(M_\omega) \geq \hat{y_0z_0}(M_\omega) - 4(n/2) = \hat{y_0z_0}[S] - 4(n/2).$$

Now consider the current dual objective function. Using (3.18) we can rewrite the current dual objective function:

$$\hat{y_0z}[S] = y(V(M)) + y(F) + \sum_{B \in \Omega \cup \Omega^-} z(B)|\gamma_M(B)| + \sum_{B \in \Omega^- \text{ undissolved}} z(B)|F \cap B|/2$$

$$= \hat{y_0z}(M) + y(F) + \sum_{D \subseteq B \in \Omega^-} z(B)|F \cap B|/2.$$  

Note the range $D \subseteq B \in \Omega^-$ in the summation term is justified by (3.17) and the fact that $\Omega^-$-blossoms $B \subseteq S$ have dissolved.

Since $M_\omega$ is a perfect matching of $S$, $\hat{y_0z}(M_\omega) \leq \hat{y_0z}[S]$. Using the last displayed equation for $\hat{y_0z}[S]$, plus the previous lower bound on $w(M_\omega)$, plus near domination and near tightness of $y, z$, gives

$$\hat{y_0z_0}[S] - 3n \leq w(M_\omega) - 2(n/2) \leq \hat{y_0z}(M_\omega)$$

near domination of $\hat{y_0z}$
\[
\leq \hat{y}z[S] \\
\leq y(F) + w(M) + \sum_{D \subseteq B \in \Omega^-} z(B)|F \cap B|/2 \quad \text{near tightness of } \hat{y}z.
\]

(3.21)

We continue these inequalities by upper-bounding \(y(F)\) and \(w(M)\). Sum (3.16) for every \(v \in F\) to get

\[
y(F) = y_0(F) - d(F) + \sum_B |F \cap B|\tau(B).
\]

Bound the matched edges by

\[
w(M) \leq y_0(V(M)) + \sum_B z_0(B)|\gamma_M(B)| + 2(n/2) \quad \text{near domination of } \hat{y_0}z_0.
\]

Combining the last two inequalities gives

\[
y(F) + w(M) \leq y_0(S) - d(F) + \sum_B \left( |F \cap B|\tau(B) + z_0(B)|\gamma_M(B)| \right) + n.
\]

(3.22)

3.4.3 Analysis of Blossom Crossings

We will show

\[
\beta \leq \sum_B z_0(B)|B \cap S|/2 - \sum_{D \subseteq B \in \Omega^-} |F \cap B|z(B)/2 + \Delta
\]

(3.23)

where \(\Delta\) consists of the \(\tau\) and \(\kappa_\varepsilon\) terms of (3.4),

\[
\Delta = \tau(U) - \kappa_0(INT(S)) - \kappa_1(SUB(S)).
\]

Figure 11 illustrates the \(\kappa\) terms.

The blossoms \(B\) contributing to \(\beta\) belong to \(\Omega^-\) (\(\tau(B)\) or \(z_0(B)\) is positive) with \(B \cap S \neq \emptyset\) (\(|F \cap B|\) or \(|\gamma_M(B)|\) is positive). Thus they satisfy (3.17). We consider the two corresponding possibilities, choosing the possibility \(B \subset S\) when \(S = (V, \emptyset), Q = X\). Each possibility will have two subcases, and all four of these subcases are similar. In particular they use (3.19) in the form

\[
|F \cap B| + 2|\gamma_M(B)| = |S \cap B| - |\delta_M(B)|.
\]

We always have \(z_0(B)/2 \leq z_0(B)/2 - \tau(B)\) (equality holds during Phases 1 and 2).
Case $D \subset B$: Since $S \neq V$, $Q$ is a blossom. If $D$ is a blossom then either $B \in P(Q)$ or $Q \subset B$. So $|S \cap B|$ is even. If $D = \omega$ there is the additional possibility, $\omega \in B \subseteq Q'$ for some $Q' \subset Q$. In this case $S \cap B = B - \omega$ and again $|S \cap B|$ is even. We conclude

$$|S \cap B|/2 = [|S \cap B|/2] \quad (3.24)$$

in this case.

Subcase $z(B) = 0$: Using $\tau(B) \leq z_0(B)/2$ the term for $B$ in $\beta$ is at most

$$z_0(B)(|F \cap B| + 2|\gamma_M(B)|)/2 = z_0(B)(|S \cap B| - |\delta_M(B)|)/2 \leq z_0(B)|S \cap B|/2.$$  

Using (3.24) shows the upper bound matches the term for $B$ in the first summation of (3.23). There is no contribution to the second summation when $z(B) = 0$.

Subcase $z(B) > 0$: $B$ is undissoled, so either $B \in P(Q)$ or $Q \subset B$. (In other words when $D = \omega$ the additional possibility, $\omega \in B \subseteq Q'$ for some $Q' \subset Q$, occurs only in the previous subcase.) When $B \in P(Q)$ the unit translations of DismantlePath($Q$) in Phases 1 and 2 maintain the invariant $z_0(B)/2 = \tau(B) + z(B)/2$. This holds trivially in the second case $Q \subset B$, since $\tau(B) = 0$, $z(B) = z_0(B)$.

Rearranging the invariant to $\tau(B) = z_0(B)/2 - z(B)/2$ shows $B$’s term in $\beta$ is

$$|F \cap B|\tau(B) + z_0(B)|\gamma_M(B)| = (z_0(B)/2)(|F \cap B| + 2|\gamma_M(B)|) - (z(B)/2)|F \cap B| = z_0(B)(|S \cap B| - |\delta_M(B)|)/2 - (z(B)/2)|F \cap B|.$$

Using (3.24) and rearranging terms changes the last line to

$$z_0(B)[|S \cap B|/2] - (z(B)/2)|F \cap B| - |\delta_M(B)|z_0(B)/2.$$  

The first two terms match the terms for $B$ in the two summations of (3.23). The third term, $-|\delta_M(B)|z_0(B)/2$, is nonzero only when $B \in INT(S)$. (In proof $B \notin INT(S)$

\[ Springer \]
makes $S^+ \subseteq B$, even if $B \in P(Q)$. So $S \subseteq B$ and $M \subseteq \gamma(S) \subseteq \gamma(B)$, so no edge of $M$ crosses $B$.) We include this third term in the $\kappa_0$ quantity of $\Delta$.

**Case $B \subset S$:**

**Subcase $B$ crossed:** Again using $\tau(B) \leq z_0(B)/2$, the term for $B$ in $\beta$ is at most

$$z_0(B)(|F \cap B| + 2|\gamma_M(B)|)/2 = z_0(B)(|B| - |\delta_M(B)|)/2.$$ Substituting $|B|/2 = [|B|/2] + 1/2$, the displayed quantity becomes

$$z_0(B)[|B \cap S|/2] + z_0(B)(1 - |\delta_M(B)|)/2.$$ The first term matches the term for $B$ in the first summation of (3.23). $B$ does not occur in the second summation since it is dissolved. Since $B$ is crossed we can include the second term, $-z_0(B)(|\delta_M(B)| - 1)/2$, in the $\kappa_1$ quantity of $\Delta$.

**Subcase $B$ uncrossed:** This requires $B$ to contain a free vertex. Thus $|F \cap B|\tau(B) \leq \tau(B) + (|F \cap B| - 1)z_0(B)/2$. Using this and proceeding as before, $B$’s term in $\beta$ is

$$|F \cap B|\tau(B) + z_0(B)|\gamma_M(B)| \leq \tau(B) + z_0(B)(|B| - 1)/2 = \tau(B) + z_0(B)[|B \cap S|/2].$$

The second term of this upper bound matches the term for $B$ in the first summation of (3.23). As before there is no contribution to the second summation. We include the first term $\tau(B)$ in $\tau$ quantity of $\Delta$.

Using the bound on $\beta$ in (3.22) gives

$$y(F) + w(M) \leq y_0(S) - d(F) + \sum_B z_0(B)[|B \cap S|/2] - \sum_{D \subseteq B \in \Omega^-} |F \cap B|\tau(B)/2 + \Delta + n$$

$$= \hat{y}_0z_0[S] - d(F) - \sum_{D \subseteq B \in \Omega^-} |F \cap B|\tau(B)/2 + \Delta + n.$$ Combining this with (3.21) gives

$$\hat{y}_0z_0[S] - 3n \leq \hat{y}_0z_0[S] - d(F) + \Delta + n.$$ Hence

$$d(F) \leq 4n + \Delta.$$ (3.25)

Rearranging the terms of $\Delta$ gives the scaling inequality (3.4).
The reader is assumed familiar with the $f$-factor algorithm of [10] (e.g., review Fig. 12). The cogent points are reviewed in Appendix C. Many details of the scaling algorithm are identical to, or obvious analogs of, 1-matching. We postpone discussing those details until the end of Sect. 4.2. Instead this section is organized as follows. Section 4.1 presents the difficulties that arise in adapting the 1-matching algorithm to $f$-factors. It introduces two new mechanisms we use to treat these difficulties, giving details of one (“tightening”). Section 4.2 gives the $f$-factor algorithm, with detailed discussions of three new procedures (Sects. 4.2.1–4.2.3) followed by details that are analogs of 1-matching (Sect. 4.2.4).

4.1 Difficulties for $f$-Factors

We start by reviewing new features of $f$-factor duals. Then we discuss the difficulties they present.

Near optimality and $I(B)$ sets $f$-factors introduce the possibility of matched edges that cannot be nearly dominated. Let $M$ be the current matching. The duals are near-optimum if edges $e$ not in blossom subgraphs satisfy

$$\hat{y}_{z}(e) \begin{cases} \geq w(e) - 2 & e \notin M \quad \text{near domination} \\ \leq w(e) & e \in M \quad \text{near tightness}, \end{cases} \quad (4.1)$$
and edges $e$ in blossom subgraphs ($e \in E(B)$, $B$ a blossom) satisfy

$$\widehat{yz}(e) \in \{w(e) - 2, w(e)\}.$$ 

$e$ is undervalued\(^5\) if

$$\widehat{yz}(e) < w(e) - 2.$$ 

Near domination implies an undervalued edge must be matched.

We sometimes use the following terminology, from Appendix C, regarding dual values: Edge $e$ is dominated, tight, or underrated depending on whether $\widehat{yz}(e)$ is $\geq w(e)$, $= w(e)$, or $\leq w(e)$, respectively. Strictly dominated refers to the possibility $> w(e)$. Eligibility continues to be defined as in Sect. 2 [specifically Eqs. (2.3)–(2.4)].

The second new feature is that a blossom $B$ can contribute to the dual value of an edge incident to $B$, via its set $I(B)$ ("$I$" stands for "incident"). As reviewed in Appendix C the dual value of an edge is

$$\widehat{yz}(e) = y(e) + z\{B : e \in \gamma(B) \cup I(B)\},$$

where

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

i.e., an edge of $I(B)$ is incident to $B$ and either matched but not $B$’s base edge or unmatched and equal to $B$’s base edge. Figure 12 illustrates this.

**Difficulties** Our basic approach of processing one major path $P(Q)$ at a time is complicated by $I(B)$ edges that leave $Q$. They must occur in the scaling inequality, i.e., they cannot be eliminated like $M_\omega$ eliminates the base edge in 1-matching. So the DISMANTLER must handle them.

As an example of the difficulty caused by $I(B)$ sets consider an edge $uv$ joining disjoint mprs $Q_u \ni u$ and $Q_v \ni v$. Suppose $uv \in I(Q_u) - I(Q_v)$ (e.g., in the previous scale $uv$ was unmatched and the base edge of $Q_u$ but not $Q_v$). As usual a unit translation of $Q_v$ has $\Delta(\widehat{yz}(uv)) = 1$ ($\Delta(y(u)) = 1$), but a unit translation of $Q_u$ has $\Delta(\widehat{yz}(uv)) = -1$ ($\Delta(y(u)) = 1$, $\Delta(z(Q_u)) = -2$). So executing many unit translations of $Q_v$ can force $uv$ to be unmatched, while many translations of $Q_u$ can force it to be matched. This inconsistency can lead to various contradictions. For example, suppose first DISMANTLEPATH($Q_v$) makes $uv$ unmatched and $v$ saturated (i.e., on $f(v)$ matched edges). Later DISMANTLEPATH($Q_u$) makes $uv$ matched. This violates $v$’s degree constraint. Alternatively first DISMANTLEPATH($Q_u$) makes $uv$ matched. Assume $u$ gets saturated, and subsequent searches in DISMANTLEPATH($Q_u$) do not decrease $y(u)$. Then DISMANTLEPATH($Q_v$) makes $uv$ unmatched. $u$ becomes free. But now $u$ violates the crucial inequality (3.16) for free vertices.

Another difficulty in the same situation is when there are many different $u$ vertices incident to the same vertex $v$. The various DISMANTLEPATH($Q_u$) invocations force each $uv$ edge to be matched. Vertex $v$ may get $\gg f(v)$ matched edges.

We avoid such problems in several ways.

---

\(^5\) The term “undervalued” is the analog of “strictly underrated” when optimum duals are used. See Appendix C.
Fig. 13 Eliminating $I(B)$ sets by edge expansion: (a) Edge $uv \in I(B_u)$. (b) Expansion of $uv$ for $uv \in I(B_u) \cap I(B_v)$. (c) Expansion for $uv \in I(B_u) - I(B_v)$

**Edge expansion** We eliminate $I(B)$ sets by using an alternative graph model for the dual values $\hat{y}(\cdot)$. Specifically we replace an edge $uv \in I(B)$ by a length 3 path $u, u\bar{v}, v\bar{u}, v$, for artificial vertices $u\bar{v}, v\bar{u}$, with edge $(u\bar{v}, v\bar{u})$ matched iff $uv$ is unmatched. As illustrated in Fig. 13, the artificial vertices can be placed so that every blossom $B$ has $uv \in I(B)$ iff $(u, u\bar{v}) \in \gamma(B)$. This makes the contribution of $uv$ to $\hat{y}(uv)$ correspond to the same criterion as blossoms for 1-matching, so our previous approach can work.

We call an edge $uv \in I(B_u) \cup I(B_v)$ an $I$-edge. We call its length 3 path the expansion of $uv$. We call the blossoms of the alternative graph model “e-blossoms”.

Edge expansion increases the size of the graph. To prevent it from growing too big, we compress expanded edges back to their original source $uv \in E(G)$, when expansion is no longer needed.

Expansion by length 3 paths is used in various reductions of general matching to 1-matching [24, Ch.32]. This replacement is used by Duan et al. [3] to overcome problems of $I$-edges. In more detail they use the “blowup graph”, wherein every original edge gets expanded. We cannot use this graph, since it increases the size of an $f$-factor (and its blossoms) up to $\Theta(m)$ rather than $\Theta(\Phi)$.

Edge compression can create two new difficulties: violations of standard blossom structure (“baseless blossoms”) and problematic structures (ineligible base edges). Figure 25 of Appendix C illustrates how both these configurations can be introduced by edge compression. We resolve the problem of ineligible base edges in a general fashion. It may find other applications. For that reason we present our resolution in this section before proceeding to the $f$-factor algorithm (Sect. 4.2).

**Tightening base edges** This difficulty is caused by base edges of blossoms that are ineligible. (An ineligible base edge that is undervalued and becomes matched may cause the problems illustrated above. An ineligible base edge that is unmatched can invalidate the analysis of the algorithm.) The base edge of a maximal blossom may be ineligible. (This is the only possibility—other base edges belong to a blossom subgraph, and hence are eligible.)

We use the high-level procedure of Fig. 14 to eliminate base edges that are undervalued or strictly dominated. The procedure proves the following structural fact. An $\eta$-edge is the base edge of some blossom.
Algorithmica (2023) 85:3214–3289
3251

Fig. 14 High-level algorithm to eliminate ineligible base edges. Subroutines for breaking C and tightening e are given below

\[
\begin{align*}
\text{while } \exists \text{ a cycle C of } \eta\text{-edges} \\
& \quad \text{break C} \\
\text{while } \exists \text{ an ineligible } \eta\text{-edge e} \\
& \quad \text{choose an e with minimal depth} \\
& \quad \text{tighten e}
\end{align*}
\]

Lemma 4.1 A maximum weight f-factor always has (optimum or near-optimum) dual variables wherein every \( \eta \)-edge is eligible.

(An edge is eligible for optimum duals if it is tight, \( \hat{\varphi}(\eta) = w(\eta) \).

The rest of this section gives the details of the procedure, thereby proving the lemma. We also show the procedure uses \( O(n) \) time. For completeness we present the algorithm for both optimum and near-optimum duals, even though we only use the latter. To do this define \( t(\eta) \), the “target” value for eligibility, to be \( w(\eta) - 2 \) for an unmatched edge \( \eta \) with near-optimum duals, and \( w(\eta) \) otherwise (optimum duals, or \( \eta \) matched with near-optimum duals).

We start with a high level overview of the algorithm. For any \( f \)-factor with arbitrary duals and blossoms, use the \( \eta \)-edges to define a directed pseudoforest \( P \) as follows. Contract every maximal blossom. Retain only their \( \eta \)-edges, each directed away from its base vertex. All other edges are discarded. An edge is bidirected if it is the base edge of both its end blossoms.

Every contracted blossom of \( P \) has outdegree 1 and every atom has outdegree 0. So \( P \) is a directed pseudoforest, i.e., a collection of connected components, each consisting of a root subgraph \( R \) (an atom, a bidirected edge, or a directed cycle) plus zero or more in-trees, each rooted at a vertex of \( R \).

The algorithm of Fig. 14 uses unit translations to reduce \( z \)-values and possibly dissolve blossoms. The first step, “breaking”, eliminates cycles from the pseudoforest. So every root subgraph is an atom or a bidirected edge. The second step, “tightening”, does a top-down traversal of each remaining (acyclic) component \( C \). Each ineligible \( \eta \)-edge is either made eligible, or its tail becomes atomic (so it is no longer an \( \eta \)-edge). To define “top-down”, in Fig. 14 say that an ineligible edge \( \eta \) has minimal depth if the directed path from \( \eta \) to the root subgraph of its connected component \( C \) consists entirely of eligible edges (other than \( \eta \)). Note this makes sense when the root subgraph is bidirected.

We now proceed with the details of the two steps of Fig. 14. The first step executes the following procedure on each connected component \( C \) of \( P \).

/* breaking routine */
while \( C \) contains a directed cycle of contracted blossoms \( R \)
\[ \delta \leftarrow \min\{z(B)/2 : B \text{ a maximal blossom in } R\} \]
translate every maximal blossom of \( R \) by \( \delta \)

First observe that the cycle \( R \) may change from iteration to iteration. This is because a blossom on \( R \) may dissolve and get replaced by a path of subblossoms and atoms. The processing of \( C \) is complete the first time \( R \) contains an atom. Note also that as
vertices and subblossoms leave \( R \), some trees of \( C \) may no longer be rooted in (the new) \( R \). This causes no problem.

For correctness we must show that each unit translation preserves optimality (or near optimality) of the duals. There are several cases. First consider an \( \eta \)-edge \( uv \) on \( R \). It suffices to show \( \hat{\gamma}z(uv) \) does not change. Clearly \( y(uv) \) increases by \( 1 + 1 = 2 \). So it suffices to show the \( z \) terms contributing to \( \hat{\gamma}z(uv) \) decrease by 2. Let \( A_u \) and \( A_v \) be the maximal blossoms containing \( u \) and \( v \) respectively. It suffices to show \( uv \) belongs to exactly one of the sets \( I(A_u), I(A_v) \). Observe \( uv = \eta(A_u) \neq \eta(A_v) \) (this follows since \( R \) is a cycle throughout the breaking routine, not a bidirected edge).

If \( uv \) is matched then \( uv \in I(A_v) - I(A_u) \), as desired. If \( uv \) is unmatched then \( uv \in I(A_u) - I(A_v) \), as desired. We conclude the duals remain optimum on the edges of \( R \).

Now consider an edge \( e \) that is not on \( R \) but has one or both ends in blossoms of \( R \). If \( e \in \gamma(B) \) for a blossom \( B \) of \( R \), a unit translation of \( B \) preserves the value of \( \hat{\gamma}z(uv) \). Suppose \( e \in \delta(B) \). If \( e \) is matched then \( e \in I(B) \) (since \( e \neq \eta(B) \)). So \( \hat{\gamma}z(e) \) decreases by 1, and the requirement \( \hat{\gamma}z \leq t(e) \) is preserved. (If both ends of \( e \) are in such blossoms the decrease is 2.) If \( e \) is unmatched then \( e \notin I(B) \) (since \( e \neq \eta(B) \)). So \( \hat{\gamma}z(e) \) increases by 1, and the requirement \( \hat{\gamma}z \geq t(e) \) is preserved. (If both ends of \( e \) are in such blossoms the increase is 2.)

Clearly there is no cycle of \( \eta \)-edges when the first step halts.

Since we use this algorithm (in the compression step below), let us show that it can be implemented in time \( O(n) \). Consider a directed cycle \( R_0 \) in the initial pseudoforest \( P \). As the algorithm progresses \( R_0 \) changes into various cycles \( R \), terminating when \( R \) contains an atom \( x^* \). The approach is to find \( x^* \). For any vertex \( x \) define \( \Delta(x) = \sum_{v \in \Delta} z(A) / 2 \). \( \Delta(x) \) is the number of unit translations that can be done on the blossom currently containing \( x \) until \( x \) becomes atomic. Consider a maximal blossom \( B \) of \( R_0 \). Let \( R_0 \) enter \( B \) on edge \( uv \) and leave it on \( u'v' \). Let \( T \) be the alternating trail in \( B \) that joins \( v \) to \( u' \), and alternates with both \( uv \) and \( u'v' \). Let \( \Delta = \min_{x \in T} \Delta(x) \). As the algorithm progresses, \( B \) is replaced by paths consisting entirely of subblossoms, until \( \Delta \) unit translations have been made. So \( x^* \) is the vertex that minimizes \( \Delta(x) \) from among all maximal blossoms \( B \) of \( R_0 \). The total number of unit translations made in the algorithm for a given initial cycle \( R_0 \) is \( \Delta(x^*) \).

We achieve time \( O(n) \) using standard data structures: The quantities \( \Delta(x), x \in V \), can be computed using the blossom tree. The trails \( T \) are computed as the \( P_i \) trails of Appendix C (or [10]). The unit translations of the algorithm are executed using the blossom tree. The straightforward details of this approach give total time \( O(n) \) for step one.

We turn to step two. Let \( e = uv \) be the edge of step two. We tighten \( e \) using the following procedure.

```c
/* tightening routine */
let e = uv = \eta(B) \) with \( u = \beta(B) \)
while u is not atomic and uv is ineligible
  \delta \leftarrow \min\{z(B) / 2, |\hat{\gamma}z(e) - t(e)| \} : B the maximal blossom containing u
  translate B by \delta
```

\( \square \) Springer
compress all expanded edges /* this changes the graph back to G */
        dissolving baseless blossoms and tightening \( \eta \)-edges
scale up the dual values
expand all \( I(B) \) edges /* this forms the expanded graph \( \overline{G} \) */
        matching every undervalued edge
use the \( f \)-DISMANTLER to dismantle all inherited blossoms
        and find a nearly optimum \( f \)-factor on \( \overline{G} \)

Fig. 15  Pseudocode for a scale of the \( f \)-factor algorithm

Note the root edge \( e = uv \) may be bidirected. We can choose either orientation to be processed. If \( u \) becomes atomic and \( uv \) is still ineligible, the algorithm of Fig. 14 will eventually choose \( e = vu \) as an ineligible \( \eta \)-edge.

Every unit translation decreases the distance from eligibility, \(|\hat{yz}(e) - t(e)|\). This follows since \( e \) matched implies \( \hat{yz}(e) < t(e) \) and \( e \notin I(B) \), and \( \hat{yz} \) increases by 1. Symmetrically \( e \) unmatched implies \( \hat{yz}(e) \geq t(e) \) and \( e \in I(B) \), and \( \hat{yz} \) decreases by 1. As in step one every unit translation maintains feasibility of the duals for edges of \( \delta(B) - \eta(B) \).

It is possible that the translation changes an edge \( f \) from eligible to ineligible. This can occur when \( f \) is the \( \eta \)-edge \( rs \) with \( s \in B \) (the translated blossom). So we must show that \( f \) was not previously tightened by step two. If \( f \) was previously ineligible, it would not have minimal depth, because of \( e \). Thus \( f \) would not be chosen for tightening.

We conclude that step two is correct. The time for step two in its entirety is easily seen to be \( O(n) \). This concludes the proof of Lemma 4.1.

4.2 The Algorithm and Basic Properties

Our algorithm computes a maximum weight \( f \)-factor for any given multigraph \( G \), loops allowed. Figure 15 gives the high-level algorithm for a scale. Other details of the complete algorithm (including initialization, a scaling loop, and postprocessing of the final graph \( \overline{G} \)) are simple analogs of the 1-matching algorithm. They are given in detail below. We call the generalization of the algorithm of Sect. 2 (Fig. 4) the \( f \)-DISMANTLER. It is essentially identical to the DISMANTLER (see Sect. 4.2.4). As before we treat \( V \) as a blossom (obviously \( I(V) = \emptyset \)).

Our result on \( f \)-factors is the following theorem. The theorem is proved in this section and the next. To give a broad overview we outline the proof below.

**Theorem 4.2**  The scaling algorithm (using the algorithm of Fig. 15 for each scale) finds a maximum weight \( f \)-factor on an arbitrary multigraph in time \( O(\sqrt{\Phi} \log \Phi m \log \Phi W) \).

**Proof Outline**  The rest of Sect. 4 gives the details for Fig. 15 and proves their basic properties. Excluding the \( f \)-DISMANTLER, The time for one execution of Fig. 15 is \( O(m) \).
Section 5 analyzes the $f$-Dismantler: Sect. 5.1 establishes basic properties of the expanded graph $\overline{G}$. Then Sect. 5.3 follows the analysis of the Dismantler in Sect. 3. In particular it derives the scaling inequality (5.3), the analog of (3.4). This inequality is the basis of the accounting strategy that proves the fundamental result, Lemma 3.4. Recall the details of the analysis in Sect. 3.1 are valid for $f$-factors. So inequality (5.3) completes the proof of the theorem.

The next two subsections give the details of edge expansion and compression. In a graph with a matching the $M$-type of an edge is M or U, depending on whether the edge is matched or unmatched respectively. For any edge $e$ we write $\gamma z(e) = w(e) + \Delta(e)$. When $e$ is an undervalued matched edge, $\Delta(e)$ is the value $-u(e)$ from the linear program for a maximum weight $f$-factor (given in Appendix C).

### 4.2.1 Edge Expansion

This section defines the expansion of an edge, both the new function values (e.g., edge weights, dual variables) and placement of vertices. It ends by proving the basic property that the dual variables make the edges of an expansion tight.

An edge $uv$ that belongs to at least one set $I(B)$ gets expanded. The expansion, illustrated in Figs. 16 and 17, is defined in the following discussion. We explicitly give the details for the new vertex $u\bar{v}$, and we rely on symmetry to infer the details for $v\bar{u}$. These two vertices are called the e-vertices. So an expanded edge has two $G$-vertices (its ends) and two e-vertices. The expansion consists of the two end edges $(u, u\bar{v})$ and $(v, v\bar{u})$. The algorithm uses a slightly different matching when it expands the graph for the $f$-Dismantler, see Sect. 4.2.3.

Functional values Functional values of an expanded edge are illustrated in Fig. 16. The two e-vertices have $f$-value (degree constraint) 1. Where $f$-factors are concerned we use the natural correspondence between an edge and its expansion: $uv$ has the same M-type as the two end edges $(u, u\bar{v})$ and $(v, v\bar{u})$, opposite M-type from $(u\bar{v}, v\bar{u})$. We call this the natural matching of the expanded graph. The algorithm uses a slightly different matching when it expands the graph for the $f$-Dismantler, see Sect. 4.2.3. Starting with the known values $w(uv)$ and $\Delta(uv)$ we distribute them to the two end edges via the definitions

$$w(uv) = w(u, u\bar{v}) + w(v, v\bar{u})$$
\[ \Delta(uv) = \Delta(u, u\bar{v}) + \Delta(v, v\bar{u}). \]

Note \( \Delta(uv) \) is defined in the figure. \( w(uv) \) and \( \Delta(uv) = \tilde{y}z(uv) - w(uv) \) are even because of the scaling loop. (This is exactly the same as 1-matching. The scaling operation for \( f \)-factors is given at the start of Sect. 4.2.3 and all values are even.) The exact values of the expansion edge quantities (i.e., \( w(u, u\bar{v}), \) etc.) are flexible but must be even. This is always possible since \( w(uv) \) and \( \Delta(uv) \) are both even, and an even value \( 4a + b, b \in \{0, 2\} \) can be expressed as \( 4a + b = 2a + (2a + b) \). So we will assume the magnitude of any distributed value is at most the magnitude of its source, e.g., \( \Delta(u, u\bar{v}) \leq \Delta(uv) \). The only other constraints on \( \Delta(\cdot) \) values come from undervalued edges, which must be matched. The definition of \( \Delta(\cdot) \) for these edges will be given in Fig. 19.

The \( f \)-factors on \( G \) and \( \overline{G} \) correspond 1-to-1 with edge weights preserved. Our algorithm will halt with a maximum weight \( f \)-factor on the \( \overline{G} \) graph of the last scale. The correspondence gives the desired \( f \)-factor on \( G \). The dual variables are near optimum on \( \overline{G} \). If optimum duals on \( G \) are desired they can be found using the techniques from 1-matching [15].

The \( e \)-vertices have \( y \) values defined so the end edges have dual values analogous to \( uv \). In other words we want

\[ \tilde{y}z(u, u\bar{v}) = w(u, u\bar{v}) + \Delta(u, u\bar{v}). \]  

(4.2)

This is easily achieved by defining \( y(u\bar{v}) = w(u, u\bar{v}) + \Delta(u, u\bar{v}) - (y(u) + z(B : u, u\bar{v} \in B)) \). (The blossoms \( B \) containing \( u \) and \( u\bar{v} \) are illustrated in Fig. 17 and are explicitly defined in next subsection.) As required \( y(u\bar{v}) \) is even, \( y(v\bar{u}) \) is treated symmetrically. We show below that this makes \( (u\bar{v}, v\bar{u}) \) tight, i.e., \( \tilde{y}z(u\bar{v}, v\bar{u}) = 0 \).

**Vertex placement** Recall \( B_u \) denotes the smallest blossom containing vertex \( u \). It always exists since we define \( V \) as a blossom. \( B_{uv} \) is the smallest blossom containing both \( u \) and \( v \). Again it always exists. Note that \( uv \) gets expanded exactly when it is an \( I(B) \) edge (Fig. 15), i.e., when \( uv \in I(B_u) \cup I(B_v) \).

We define the location of \( e \)-vertices in the laminar family of blossoms by specifying \( B_{u\bar{v}} \), the minimal blossom containing \( u\bar{v} \). (There is no danger of interpreting this as the minimal blossom containing \( u \) and \( \bar{v} \), since no vertex \( \bar{v} \) exists.) The following definition is illustrated in Fig. 17:

\[
B_{u\bar{v}} = \begin{cases} 
B_u & uv \in I(B_u) \\
B_{uv} & uv \notin I(B_u) 
\end{cases}
\]

The same definition holds mutatis mutandis for \( B_{v\bar{u}} \), i.e., it equals \( B_v \) when \( uv \in I(B_v) \).

Some special cases are worth noting. When no blossoms of \( P(Q) \) separate \( u \) and \( v \) (the three circular arcs disappear) and \( uv \in I(B_u) \oplus I(B_v) \), Fig. 17b, c are identical up to renaming. (This case includes the possibility \( B_{uv} = V \).) Also throughout Fig. 17 \( B_u \) or \( B_v \) may be a blossom of \( P(Q) \) (the corresponding oval disappears). Finally note the alternatives in the definition of \( B_{u\bar{v}} \) need not be distinct: In Fig. 17c assume there are only two blossoms, the oval \( B_v \) and the arc \( B_{uv} \). Then \( B_{u\bar{v}} = B_{uv} = B_u \). (We shall
Fig. 17 Expansion of an edge \( uv \in I(B_u) \cup I(B_v) \). \( uv \) may be matched or unmatched. Blossoms of \( P(Q) \), where \( B_{uv} \in P(Q) \), are shown as circular arcs; the minimal blossoms \( B_u \) and \( B_v \) are shown as ovals. a \( uv \in I(B_u) \cap I(B_v) \). b \( uv \in I(B_u) - I(B_v) \). c \( uv \in I(B_v) - I(B_u) \)

see this possibility is illustrated in Fig. 18 below, for \( u \) chosen as the vertex labelled \( x \) and \( u \bar{v} \) any of the three vertices labelled \( w \).

**Analysis of expansion duals** The rest of Sect. 4.2.1 proves the middle edge of an expansion is tight, see (4.4). We start by characterizing the e-vertices. Let \( G \) be an expanded graph immediately after its construction (i.e., right before the invocation of the \( f \)-Dismantler in Fig. 15). Figure 17 illustrates the following characterization. (In fact the characterization is easy to see using Fig. 13.)

**Lemma 4.3** Let \( uv \) be an \( I(B) \) edge, so it is expanded in \( \overline{G} \). Let \( B \) be an e-blossom of \( G \).

(i) \( uv \) crosses \( B \) iff an edge of its expansion crosses \( B \).

(ii) Suppose \( uv \) crosses \( B \), say \( u \in B \not\in v \). Then \( v \bar{u} \not\in B \). Furthermore \( u \bar{v} \in B \) iff \( uv \in I(B) \).

**Proof** The definition of \( B_{u \bar{v}} \) immediately gives

\[ (a) \quad u \bar{v} \in B_{uv} \]
\[ (b) \quad u \bar{v} \in B \implies u \in B. \]
Proof of (i): We prove the inverse. First suppose \( u, v \notin B \). Property (b) and symmetry shows \( u\bar{v}, v\bar{u} \notin B \). Next suppose \( u, v \in B \). \( B_{uv} \subseteq B \) and property (a) gives \( u\bar{v}, v\bar{u} \in B \). Proof of (ii): Property (b) again shows \( v\bar{u} \notin B \).

Next we show \( u\bar{v} \in B \) iff \( uv \in I(B_u) \). This case implies \( B_u \subseteq B \subseteq B_{uv} \). Now the definition of \( B_{u\bar{v}} \) shows \( u\bar{v} \in B \) if \( uv \in I(B_u) \) and \( u\bar{v} \notin B \) if \( uv \notin I(B_u) \).

Since \( uv \) leaves \( B \) it leaves \( B_u \). The definition of \( I(B) \) shows \( uv \in I(B_u) \) iff \( uv \in I(B) \). (We use the fact that \( uv = \eta(B_u) \) iff \( uv = \eta(B) \).) \( \square \)

As usual \( \Omega^- \) is the family of inherited blossoms, and \( \Omega \) is the family of blossoms constructed by the \( f \)-DISMANTLER. Call an \( f \)-factor blossom, as defined in [10], a standard blossom. (So a standard blossom has an \( \eta \)-edge and an \( I(B) \) set.) Define

\[
\Omega^e = \text{the family of all inherited blossoms, as modified by edge expansion.} \quad (4.3)
\]

The term \( e \)-blossom refers to a member of \( \Omega^e \).

The \( f \)-DISMANTLER algorithm dismantles \( e \)-blossoms, in the process constructing standard \( \Omega \)-blossoms in \( \overline{G} \). As illustrated in Fig. 18 \( e \)-blossoms have slightly different structure than standard blossoms: There may be \( e \)-vertices in the closed path defining

---

**Fig. 18** A heavy \( e \)-blossom \( B \), and shell \( S \) with outer boundary \( B \). Edges in \( C(B) \) and their expansions are solid. Other expanded edges are dashed. The natural matching is used. (The vertex labels \( w \) and \( w' \) are explained in Lemma 5.2)
the blossom as well as e-vertices not on that path. Like blossoms for 1-matching, e-
blossoms do not have \( I(B) \) sets. Like all blossoms \( \Omega^e \) is a laminar family. Its blossom
tree is used to determine mprs and hence the invocations \( \text{DismantlePath}(Q) \).

Our approach to analyzing dual variables in expansion and compression is based
on a “compression quantity” \( C \) and “expansion quantity” \( E \),

\[
C = \hat{y}z(uv) \\
E = \hat{y}z(u, u\bar{v}) + \hat{y}z(v, v\bar{u}) - \hat{y}z(u\bar{v}, v\bar{u})
\]

When \( uv \) gets expanded we start with the known quantity \( C \) and analyze the new
quantity \( E \); vice versa for compression. \( C \) and \( E \) are closely related. For instance any
arbitrary values of the four \( y \) terms make equal contributions to \( C \) and \( E \),

\[
y(u) + y(v) = (y(u) + y(u\bar{v})) + (y(v) + y(v\bar{u})) - (y(u\bar{v}) + y(v\bar{u})).
\]

We now show the \( z \) terms also make identical contributions on expansion. (For com-
pression the contributions needn’t be identical but they relate in the proper fashion, see below.)

**Lemma 4.4** An \( I(B) \) edge \( e = uv \) has \( C = E \).

**Proof** Let \( B \) be an arbitrary blossom of \( G \). We show \( B \) contributes the same amount
to the quantities \( C \) and \( E \) for the expanded edge \( e \).

**Case** \( uv \) does not cross \( B \): Lemma 4.3(i) shows no expansion edge crosses \( B \). If all
four vertices \( u, u\bar{v}, v\bar{u}, v \) belong to \( B \) then \( B \) contributes \( z(B) \) to \( C \) and \( z(B) + z(B) - z(B) = z(B) \) to \( E \). If none of the four vertices belong to \( B \) then \( B \) contributes 0 to both \( C \) and \( E \).

**Case** \( uv \) crosses \( B \): Wlog \( u \in B \not\ni v \). Lemma 4.3(ii) shows \( v\bar{u} \not\in B \), so edges \( (u\bar{v}, v\bar{u}) \) and \( (v\bar{u}, v) \) do not contribute to \( E \). Lemma 4.3(ii) shows \( u\bar{v} \in B \) iff \( uv \in I(B) \). When
both sides are true \( B \) contributes \( z(B) \) to both \( C \) and \( E \). When both sides are false \( B \) contributes 0 to both \( C \) and \( E \).

The lemma implies \( (u\bar{v}, v\bar{u}) \) is tight:

\[
\hat{y}z(u\bar{v}, v\bar{u}) = \hat{y}z(u, u\bar{v}) + \hat{y}z(v, v\bar{u}) - E \\
= w(uv) + \Delta(uv) - E \\
= \hat{y}z(uv) - E = C - E = 0. \quad (4.4)
\]

### 4.2.2 Edge Compression

This section gives the details for compressing an edge, i.e., replacing an edge expansion
by its original source. In addition it analyzes the dual variables of the compressed edge
[which will be used in the ensuing scaling up operation; see (4.8)].
The compression step begins with an $f$-factor that has standard blossoms on the expanded graph $\overline{G}$. It constructs an $f$-factor that has standard blossoms on the corresponding graph $G$. As stated in Fig. 15 this is accomplished in three main substeps. The first is a simple replacement of every edge expansion by its source. The two post-processing steps restore the structure of standard blossoms. We begin by presenting the replacement step and analyzing it. After that we do the same for the two post-processing steps.

We set the stage by reviewing the structure of the $f$-factor on $G$ that is returned by the $f$-Dismantler. Let $B$ be a blossom ($B \in \Omega_1$ is a standard blossom). Let $uv$ be an edge whose expansion is in $\overline{G}$. Instead of Lemma 4.3 the following holds. Consider the expanded graph immediately after the $f$-Dismantler returns.

Either all four vertices $u, u\bar{v}, v\bar{u}, v$ belong to $B$ or neither $u\bar{v}$ nor $v\bar{u}$ belongs to $B$.  

\begin{equation}
(4.5)
\end{equation}

In proof it suffices to show that when $u\bar{v}$ belongs to $B$ all four vertices belong to $B$. This follows since every vertex in a standard blossom $B$ is on at least two edges of the subgraph of $B$. (In detail, $B$’s subgraph contains the edges of the cycle $C(B)$ that forms $B$, plus the edges in the subgraphs of the contracted blossoms on $C(B)$.)

The compression step replaces each edge expansion $(u, u\bar{v}, v\bar{u}, v)$ by $uv$. The end edges $(u, u\bar{v})$ and $(v\bar{u}, v)$ have the same $M$-type as $uv$, $(u\bar{v}, v\bar{u})$ has opposite $M$-type. Since the compressed graph $G$ has standard blossoms, we must specify the $\eta$ edge and $I(B)$ edges for each blossom $B$.

Any $\eta$ or $I(B)$ edge that belongs to $E(G)$ does not change in compression. Similarly if no expansion edge crosses $B$ then no expansion edge is an $\eta$ or $I(B)$ edge, so the same holds for $uv$. In the remaining case by (4.5) assume (wlog)

\[ u \in B \neq u\bar{v}, v\bar{u}. \]

Define

\[
\begin{align*}
uv \in I(B) & \quad \text{if } v \notin B \text{ and } (u, u\bar{v}) \in I(B) \\
uv = \eta(B) & \quad \text{if } v \notin B \text{ and } (u, u\bar{v}) = \eta(B) \\
\eta(B) = \emptyset & \quad \text{if } v \in B \text{ and } (u, u\bar{v}) = \eta(B).
\end{align*}
\]

\begin{equation}
(4.6)
\end{equation}

The last case makes $B$ (and possibly other blossoms) baseless. The post-processing remedies this, as discussed below. To check the validity of the first two cases, first recall this general property: Consider an edge $e \in \delta(B)$ for some arbitrary blossom $B$.

\[ e \in I(B) \iff \begin{cases} e \neq \eta(B) & e \in M \\
e \eta(B) & e \notin M. \end{cases} \]

\begin{equation}
(4.7)
\end{equation}
Since \( uv \) and \((u, u\bar{v})\) have the same M-type, the first two cases are valid definitions. (The requirements on a base edge given in the definition of “blossom” are easily seen to hold since edge \( uv \) and its expansion have the same ends, vertices \( u \) and \( v \).)

Next we analyze how compression affects the dual variables.

**Lemma 4.5** Compressing the expansion of edge \( uv \) makes

\[
\begin{align*}
C &\leq E \text{ if } uv \text{ gets matched} \\
C &\geq E \text{ if } uv \text{ gets unmatched.}
\end{align*}
\]

In addition \( C = E \) if \( uv \) becomes a blossom subgraph edge (i.e., if the expanded edges are all contained in \( E(B_{uv}) \)).

**Proof** Let \( B \) be the blossom of interest. Clearly \( C = E = 0 \) if none of the four expansion vertices belongs to \( B \). If the opposite holds then \((4.5)\) implies \( \{u, u\bar{v}, v\bar{u}, v\} \cap B \) is either \( \{u, u\bar{v}, v\bar{u}, v\} \), \( \{u, v\} \), or \( \{u\} \)

where the last case holds wlog. We consider the corresponding three cases. We will see the first two have equal contributions to \( C \) and \( E \).

**Case** \( \{u, u\bar{v}, v\bar{u}, v\} \subseteq B \): \( B \) contributes to all three expansion edges, so the net contribution to \( E \) is \( z(B) + z(B) - z(B) = z(B) \). So \( B \) makes the same contribution to \( C \) and \( E \).

**Case** \( \{u, u\bar{v}, v\bar{u}, v\} \cap B = \{u, v\} \): \( B \) contributes to \( E \) iff \( (u, u\bar{v}) \in I(B) \). Definition \((4.6)\) shows this holds iff \( uv \in I(B) \), i.e., iff \( uv \) contributes to \( C \).

**Case** \( \{u, u\bar{v}, v\bar{u}, v\} \cap B = \{u, v\} \): \( u \in \gamma(B) \) so \( B \) contributes \( z(B) \) to \( C \).

Suppose \( (u, u\bar{v}) \) and \( (v, v\bar{u}) \) are both matched. One of these edges must belong to \( I(B) \), the other belongs to \( I(B) \) iff it is not \( \eta(B) \). \( B \) contributes \( 2z(B) \) to \( E \) in the first case, \( z(B) \) in the second. In both cases the contribution to \( E \) is \( \geq \) the contribution to \( C \), as desired.

Suppose \( (u, u\bar{v}) \) and \( (v, v\bar{u}) \) are both unmatched. If one of these edges is \( \eta(B) \), \( B \) contributes \( z(B) \) to \( E \). If neither is \( \eta(B) \) the contribution is 0. So the contribution to \( E \) is \( \leq \) the contribution to \( C \), as desired.

In this last case \( uv \) does not become a blossom edge. Since equality holds in the first two cases, \( C = E \) holds when \( uv \) becomes a blossom edge. \( \square \)

At the end of the compression step let \( M \) be the \( f \)-factor (i.e., edges \( uv \) that got compressed are matched or unmatched according to their expansions). Every edge \( e \) now belongs to \( G \), and we will show

\[
\hat{\gamma}(e) = w(e) + \Delta(e) \text{ where } \Delta(e) \begin{cases} 
\geq -4 & e \notin M \\
\leq 2 & e \in M \\
\in [-4, 2] & e \in \cup \{E(B) : B \text{ a blossom}\}.
\end{cases}
\]

(4.8)
Note in the third possibility $B$ is a standard blossom.

To prove (4.8), these inequalities are obviously weak if $e$ does not result from a compression. In the opposite case let $e = uv$ with corresponding e-vertices $u \bar{v}, v \bar{u}$. Recall $w(uv) = w(u\bar{v}) + w(v\bar{u})$ and $w(u\bar{v}, v\bar{u}) = 0$. An unmatched edge $e$ has $C \geq E$ and $(u \bar{v}, v \bar{u})$ matched, so

$$\gamma(z(uv)) \geq (w(u\bar{v}) - 2) + (w(v\bar{u}) - 2) - 0 = w(uv) - 4$$

(using near domination and near tightness). Similarly a matched edge $e$ has $C \leq E$ and $(u \bar{v}, v \bar{u})$ unmatched, so

$$\gamma(z(uv)) \leq w(u\bar{v}) + w(v\bar{u}) - (−2) = w(uv) + 2.$$ 

Finally if $e$ is a blossom subgraph edge, the above inequalities all hold, since before compression every edge $e'$ in a blossom subgraph has $\Delta(e') \in [-2, 0]$.

**Baseless blossoms** The next step eliminates blossoms that have $\eta(B) = \emptyset$ after compression. First note this possibility can actually occur in the algorithm: Fig. 25 illustrates how such a blossom $B$ can be created (in an invocation DismantlePath($Q$) for arbitrary $Q$): In Fig. 25a, assume the $u \bar{v}$ and $v \bar{u}$ are the two expansion vertices. In parts (b) and (c) $z(B) > 0$ and the compression operation makes $\eta(B) = \emptyset$.

Our algorithm can also be used to eliminate baseless blossoms that occur in algorithms for maximum weight $f$-matching.

As shown in the lemma’s proof, $\eta(B) = \emptyset$ when

$$u, v \in B \not= \ u \bar{v}, v \bar{u} \text{ and } \eta(B) = (u, u \bar{v}).$$

(The fact that $v \bar{u} \not= B$ follows since we are in the lemma’s third case.)

Consider a maximal baseless blossom $B$. $B$ is actually a maximal blossom. In proof, suppose for contradiction that $B$ is a maximal subblossom of a blossom $A \not= V$. The cycle defining $A$ contains $B$ and $\eta(B) = (u, u \bar{v})$. Thus it contains the subpath $(u, u \bar{v}), (u \bar{v}, v \bar{u}), (v \bar{u}, v) = (v \bar{u}, B)$. The base edge $\eta(A)$ can only leave the cycle from $v \bar{u}$ or $v \bar{u}$, but this is impossible.

The algorithm dissolves $B$, i.e.,

$$y(v) \leftarrow y(v) + z(B)/2 \forall v \in B$$

$$z(B) \leftarrow 0.$$ 

This change preserves inequalities (4.8). In proof consider an edge in the compressed graph with one or both vertices in $B$. There are three possibilities:

- $e \in y(B)$: $\gamma(z(e))$ is preserved.
- $e \in \delta(B) \cap M$: $e \neq \eta(B)$ since $\eta(B) = \emptyset$. So $e \in I(B)$ and $\gamma(z(e))$ decreases. $e$ may become an undervalued matched edge.
- $e \in \delta(B) - M$: $e \neq \eta(B)$ since $\eta(B) = \emptyset$. So $e \not= I(B)$ and $\gamma(z(e))$ increases. The quantity $\Delta(e)$ remains $\geq -4$.

The remaining details of the blossom structure obviously hold.
In summary the overall algorithm repeatedly chooses a maximal baseless blossom and dissolves it. It is easy to implement this algorithm in time $O(n)$ using the blossom tree. We conclude that after this step, all blossoms in $G$ are valid standard blossoms in the given graph $G$.

**Tightening $\eta$-edges** We make all $\eta$-edges eligible by executing the procedure of Lemma 4.1. (Regarding step one of that procedure, the example of Fig. 25 easily extends to show the compression step can create cycles of $\eta$-edges.) Let $ET$ be the set of all $\eta$-edges. After this step we can extend the third case of (4.8) to

$$\Delta(e) \in [-4, 2] \quad e \in \{E(B) : B a blossom\} \cup ET.$$  

### 4.2.3 Scaling Up

This section presents the procedure for scaling up in Fig. 15. It is more involved than 1-matching because of undervalued edges. The algorithms of this section guarantee that the $f$-Dismantler (in Fig. 15) begins with valid near-optimum duals on the expanded graph $\overline{G}$.

We start with the initialization of the algorithm. Every edge weight $w(e)$ and every dual value $y(v), z(B)$ is initialized to 0. This makes every edge tight, so the duals are optimum for any $f$-factor.

The first scale of the algorithm, like any other, finds an $f$-factor. So we have the following result.

**Corollary 4.6** The first scale finds an $f$-factor of an arbitrary multigraph in time $O(\sqrt{\Phi \log \Phi m})$.

The scaling up operation is executed on the compressed graph $G$. In detail the $i$th scale starts by scaling up as follows:

$$w(e) \leftarrow 2w(e) + \text{the } i \text{th leading bit of } \overline{w}(e) \quad \forall e \in E$$
$$y(v) \leftarrow 2y(v) + 4 \quad \forall v \in V$$
$$z(B) \leftarrow 2z(B) \quad \forall \text{ blossom } B.$$

As before $y_0$ and $z_0$ denote the dual functions for the new scale. Observe that as required in the definition of functional values for edge expansion (Sect. 4.2.1) $\Delta(e) = \overline{y}_0z_0(e) - w(e)$ is even.

Let $M_0$ be the matching after edge compression. ($M_0 = \emptyset$ when we are starting the first scale.) Let $e$ be any edge in the compressed graph. Comparing its new weight $w(e)$ with its weight in the previous scale, denoted $w^-(e)$,

$$2w^-(e) \leq w(e) \leq 2w^-(e) + 2. \quad (4.9)$$

Combining (4.9) with (4.8) and its extension to $ET$ gives, for any edge $e \in E(G)$,

$$\overline{y}_0z_0(e) = w(e) + \Delta_0(e), \quad \text{where} \quad \Delta_0(e) = \begin{cases} \geq -2 & e \notin M_0 \\ \leq 12 & e \in M_0 \\ \in [-2, 12] & e \in \bigcup\{E(B) : B a blossom\} \cup ET. \end{cases} \quad (4.10)$$
For every $I(B)$ edge $uv$ of $G$

if $\Delta_0(uv) < -2$ then

wlog assume $uv \in I(B)$ for some blossom $B$ of $G$

$w(u, u\bar{v}) \leftarrow w(uv)$, $\Delta(u, u\bar{v}) \leftarrow \Delta(uv)$

$w(v, v\bar{u}) \leftarrow 0$, $\Delta(v, v\bar{u}) \leftarrow 0$

match $(u, u\bar{v})$

For every edge $uv$ of $G$ that joins two atoms of a shell (i.e., $B_u = B_v = B_{uv}$)

if $\Delta_0(uv) < -2$ then match $uv$

Fig. 19 Edge expansion procedure to establish near optimality

Edge expansion: initial matching This step gives the remaining functional values of expanded edges so the $f$-DISMANTLER begins with near optimum duals. This is done by the procedure of Fig. 19.

First observe that the matching of Fig. 19 is valid, i.e., any vertex $u$ is on at most $f(u)$ matched edges. This follows since unmatched edges are dominated, so only undervalued edges create a matched edge $(u, u\bar{v})$.

To prove the goal is achieved we start by showing that a version of (4.10) holds in the expanded graph. To make this precise we translate various notions from $G$ to the expanded graph $G$. In (4.10) “$M_0$” now denotes the expanded matching (as modified by Fig. 19). A “blossom” is an $e$-blossom $B$. Its blossom subgraph $E(B)$ is the $\bar{G}$-image of $E(A)$, for $A$ the $\Omega$-blossom that expands to $B$. (For example in Fig. 18 $E(B)$ excludes the dashed edges.) Similarly $ET$ is the $\bar{G}$-image of that set in $G$. (So for an expanded $\eta$-edge, all three of its expansion edges belong to $ET$.)

Proposition 4.7 The bound of (4.10), interpreted for $\bar{G}$ as described above, holds after edge expansion.

Proof Consider an edge expansion. The middle edge $(u\bar{v}, v\bar{u})$ is always tight. For end edges recall the definition (4.2) of $\hat{\gamma}z(u, u\bar{v})$ and also the distribution of $\Delta(uv)$ into quantities $\Delta(u, u\bar{v})$ and $\Delta(v, v\bar{u})$ of no greater magnitude (for $a \geq 0$ and $\sigma = \pm 1$, $\sigma(4a + 2) = \sigma(2a + 2) + \sigma(2a)$). Thus the inequalities of (4.10) can only get stronger.

Equation (4.10) shows the unmatched edges are nearly dominated. Figure 19 ensures the matched edges are undervalued. Thus the duals $y_0, z_0$ are nearly optimum when the $f$-DISMANTLER begins.

The edges $st$ matched in Fig. 19 avoid the problem of unit distributions in the Difficulties section. In detail these edges have $B_t = B_t$ (for $I(B)$ edges recall Fig. 17). This implies a unit translation of any e-blossom $B$ preserves the value of $\hat{\gamma}z(st)$ (since $B$ has no $I(B)$ edges). So the undervalued edges matched in Fig. 19 behave like any other matched edge in the $f$-DISMANTLER.

Note the upper bound on $M_0$ edges will be used in the analysis of the algorithm in Sect. 5.3. Also the problem of unmatched $\eta$-edges that are ineligible has already been avoided by Lemma 4.1.
4.2.4 Remaining Details: Scaling and the f-DISMANTLER

The overall algorithm operates in $\Theta(\log \Phi \hat{W})$ scales. We give the justification, which is analogous to 1-matching.

Let $\hat{w}$ be the given weight function, $\hat{W} = \max \hat{w}$, and $\hat{W}_f$ the maximum given weight of an $f$-factor. Let $a$ be a parameter to be determined. The algorithm replaces $\hat{w}$ by the weight function $\bar{w} = a\hat{w}$. There are $s = \lceil \log a \hat{W} \rceil$ scales. For $i = 1, \ldots, s$, the $i$th scale finds a near-optimum structured $f$-factor for the weight function

$$w(e) = 2 \times \text{the leading } i \text{ bits of } \bar{w}(e).$$

It is easy to see the last scale uses weights $\geq a\hat{w}$.

In each scale let $\bar{f}$ be the degree constraint function of the expanded graph $\bar{G}$, and let $W_f$ be the maximum weight of an $\bar{f}$-factor (using the scale’s weight function $w$). A near optimum $\bar{f}$-factor weights $\geq W_f - \bar{f}(V)$. (The proof is the same as [15, Lemma 2.1], [4, Lemma 2.2], as well as [3].) Let the $f$-factor of the last scale weigh $aW$, so $aW \geq a\hat{W}_f - \bar{f}(V)$. If $a > \bar{f}(V)$ we get $aW > a\hat{W}_f - a$. This implies $W = \hat{W}_f$, so the last scale gives a maximum weight $f$-factor on the expanded graph $\bar{G}$, as desired. We will prove the expanded graph has $\bar{f}(V) \leq 3f(V)$ [see (5.1)]. So we take $a = 3f(V) + 1$. Thus the number of scales is $\Theta(\log \Phi \hat{W})$ as desired.

Next consider the $f$-DISMANTLER. It is essentially the same as the DISMANTLER. It uses the $f$-factor algorithm of Gabow [10]. Phase 1 finds a maximal set of disjoint augmenting trails using the algorithm of Gabow [9]. The only other change is to interpret quantities appropriately for the new context. This occurs in two places, as follows. Note that in light of (5.1), for convenience we omit overlines, e.g., writing $f$ instead of $\bar{f}$.

The first change is in the definition of major path. We use the blossom tree of e-blossoms to define the major paths and associated notions. To choose the major child, each node $B$ has size $f(B)$ rather than $|B|$. (Recall from the start of Sect.3.1 that the size of a vertex set like the mpr $X$ always gets replaced by $f(X)$.)

The second change is interpretation of the termination test in Phases 1 and 2. In both tests each free vertex is counted according to its deficiency, i.e., a vertex $v$ in an atomic shell contributes $f(v) - \deg(v)$ to the count, for $\deg(v)$ the degree of $v$ in the current matching. In Phase 1 the definition of $\pi$, (3.14), becomes $\pi = c \sqrt{f(X)} \log f(X) + 1$. In Phase 2, the termination condition requires that the matching on the atomic shells has total deficiency $\leq 1$.

With these changes Lemma 3.5 becomes $d(F_X, X) \leq cf(X) \log f(X)$ for $f$-factors. We conclude DISMANTLEPATH($X$) uses total time $O(\sqrt{\bar{f}(X) \log \bar{f}(X)} m(X))$.

As in 1-matching, using major paths gives the desired time bound for the $f$-DISMANTLER. We reiterate the timing argument, which is essentially that given at the end of Sect.3. As in that section the time for DISMANTLEPATH($Q$) is $O(\sqrt{\bar{f}(Q) \log \bar{f}(Q)} m(Q))$. The mpr’s $Q$ with $2^{i-1} < f(Q) \leq 2^i$ are vertex disjoint. So the total time for invocations of DISMANTLEPATH on these mpr’s is $O(\sqrt{2^i} \ m)$. Summing over all $i \leq \log f(V)$ gives total time $O(\sqrt{f(V) \log f(V)} m)$. (5.1) shows
\( f \) and \( m \) in \( \overline{G} \) and \( G \) are the same to within constant factors. Using \( \Phi = \sum_v f(v)/2 \) from Sect. 1 the time for the \( f \)-DISMANTLER is \( O(\sqrt{\Phi} \log \Phi m) \).

5 The \( f \)-Factor Scaling Inequality

The fundamental inequality is essentially unchanged from 1-matching [(5.16) and (5.3)]. It is derived by the same sequence of steps. There are two basic differences: using \( f \) for degree bounds rather than degree bound 1, and using \( e \)-blossoms (and the family \( \Omega^e \)) instead of standard blossoms (and the family \( \Omega^- \)). Section 5.1 proves basic properties of \( e \)-blossoms that will be needed. Section 5.2 gives the statement of the scaling inequality and related notions and properties. With this background, Sect. 5.3 proves the scaling inequality, using an argument that is a minor modification of Sect. 3.4.

5.1 Preparation for the Analysis

The \( f \)-DISMANTLER is executed on the expanded graph \( \overline{G} \). We start by showing \( \overline{G} \) has size similar to \( G \):

\[
\begin{align*}
\overline{n} & \leq n + 2f(V) \\
\overline{m} & \leq m + 2f(V) \\
f(V) & \leq 3f(V).
\end{align*}
\]  

Each expanded edge adds 2 vertices, 2 edges, and 2 units of \( f \). So it suffices to show

\[ \sum_{B \in \Omega^-} |I(B)| \leq f(V). \]

We complete the proof by showing

\[ \sum_{B \in \Omega^-} 2|I(B)| \leq n + f(V) \leq 2f(V). \]

Note the second inequality is obvious.

Let \( M \) be the \( f \)-factor of \( G \) from the preceding scale. Let \( s \) be the number of edges of \( M \) that are singleton blossoms. Clearly

\[ \sum_{B \in \Omega^-} 2|I(B) \cap M| \leq 2(|M| - s) = f(V) - 2s. \]

So it suffices to show

\[ \sum_{B \in \Omega^-} 2|I(B) - M| \leq n + 2s. \]

An unmatched edge in \( I(B) \) must be \( \eta(B) \). So the above sum is \( \leq 2|\Omega^- \cap H| \) for \( H \) the set of heavy blossoms. There are \( \leq n/2 + s \) inherited heavy blossoms of \( G \). (In proof they form a laminar family on \( V \). The family has \( s \) singletons. Each nonsingleton blossom contains \( \geq 3 \) maximal subblossoms.) Thus

\[ 2|\Omega^- \cap H| \leq n + 2s \]

as desired.

Consider the expansion of the \( f \)-factor of the previous scale. \( \overline{G} \) denotes this expanded graph and \( \overline{M} \) denotes the natural matching on \( \overline{G} \). (Recall the definition of the natural matching in Sect. 4.2.1, and its illustration in Fig. 18.) The matching \( \overline{M} \) will be used throughout Sect. 5 to analyze the expanded graph. (We do not use the matching constructed by the algorithm in Sect. 4.2.3.)

The next several lemmas show that \( e \)-blossoms are much like 1-matching blossoms. This allows our credit system to extend, essentially unchanged, to \( f \)-factors. To start, 1-
matching blossoms have \(|B|\) odd. Analogously \(f\)-factor blossoms should have \(f(B)\) odd. In Fig. 20 this fails for a blossom in the original graph \(G\) but it holds after expansion. The next lemma shows this holds in general.

**Lemma 5.1** For any \(e\)-blossom \(B \neq V\), \(|\delta_{\overline{M}}(B)| = 1\). The matched edge leaving \(B\) is either the \(G\)-edge \(\eta(B)\) or an edge of its expansion.

**Proof** We will characterize the edges of \(\delta_{\overline{M}}(B)\) as follows. Consider an edge \(e \in \overline{M}\). Clearly \(e \in \delta_{\overline{M}}(B)\) iff \(e \in \delta(B)\). We will characterize when the latter holds by considering two cases. Then we will observe these characterizations imply the lemma.

**Case** \(e\) is an edge of \(G\): We show

\[ e \in \delta(B) \iff \eta(B) \text{ is matched and } e = \eta(B). \]

If \(e \in \delta(B)\) then \(e = \eta(B)\), since the matched edge \(e\) was not expanded. Conversely \(e = \eta(B)\) is obviously in \(\delta(B)\).

**Case** \(e\) is an expansion edge: We show

\[ e \in \delta(B) \iff \eta(B) \text{ is unmatched and } e \text{ is the middle edge of its expansion.} \]

Suppose \(e \in \delta(B)\). Let \(uv\) be the source of the expansion edge \(e\). So \(uv \in I(B)\). Lemma 4.3(i) shows \(uv\) crosses \(B\). Wlog \(u \in B \neq v\). Lemma 4.3(ii) shows the middle edge \((uv, vu)\) is the unique expansion edge crossing \(B\). So this edge is matched. This makes \(uv\) unmatchted. Furthermore \(uv \in I(B)\) makes \(uv = \eta(B)\).

Conversely suppose \(e\) is the middle edge of the expansion of \(\eta(B)\), which is unmatched. Since \(\eta(B)\) crosses \(B\), Lemma 4.3(i) shows an expansion edge crosses \(B\), and Lemma 4.3(ii) shows it is the middle edge \(e\).

It is easy to see the two characterizations above imply the lemma. \(\square\)
Note the lemma implies $f(B)$ is odd: Since $\overline{M}$ is a perfect matching $f(B) = 2|\gamma_M(B)| + |\delta_M(B)| = 2|\gamma_M(B)| + 1$. So any shell with $S^- \neq \emptyset$ has $f(S)$ even.

Using the lemma define the e-base edge of an e-blossom $B \neq V$ to be the edge of $\delta_M(B)$, and the e-base vertex to be the vertex of $B$ on that edge. We denote these as $\overline{\eta}(B)$ and $\overline{\beta}(B)$ respectively, writing the latter as $\overline{\beta}$ when $B$ is understood. A light blossom has $\overline{\beta} = \beta$. A heavy blossom has $\overline{\beta} \neq \beta$ with $(\beta, \overline{\beta})$ an unmatched edge.

Next we show how expanded edges in e-blossoms modify the internal blossom structure. First recall the structure of an $f$-factor blossom $B$ (see the review in Appendix C): $B$ is defined in a graph where the maximal subblossoms of $B$ are contracted. The remaining edges of $E(B)$ form a closed path $C(B)$ in this contracted graph. An atom of $B$ is a vertex in $B$ that is not in a maximal subblossom. The two ends of $C(B)$ are either the atomic vertex $\beta(B)$ or the contracted subblossom containing $\beta(B)$. In a heavy blossom $B$, $\overline{\beta}(B)$ is an atom not in $C(B)$.

An e-vertex on a cycle $C(B)$ behaves like a $G$-vertex. Other e-vertices have the following structure.

**Lemma 5.2** An e-vertex $w \notin C(B_w)$, $B_w \neq V$, is on a matched edge $wx$ where either $x$ is an atomic $G$-vertex of $B_w$, or $wx = \overline{\eta}(B_w)$ (and $x \notin B_w$).

**Remark** The first possibility for $w$ is illustrated by the three vertices labelled $w$ in Fig. 18. The second possibility is illustrated by $w = u\overline{v}$ in Fig. 20b. The alternative $w \notin C(B_w)$ is illustrated by the two vertices labelled $w'$ in Fig. 18.

**Proof** Let $w$ belong to the expansion of the $G$-edge $uv$, say $w = u\overline{v}$. Consider two cases.

**Case** $uv \in I(B_u)$: By definition $B_w = B_u$. If edge $uw = (u, u\overline{v})$ is matched take $x$ of the lemma to be $u$, noting that $u$ is atomic in $B_u$. The other possibility is that the middle edge $(u\overline{v}, v\overline{u})$ is matched. That makes $uv$ unmatched. Since $uv$ is expanded it must be $\eta(B_u)$. The second alternative of the lemma holds, since $x = v\overline{u}$ and $x \notin B_u = B_w$.

**Case** $uv \in I(B_u) - I(B_u)$: Suppose the middle edge $(u\overline{v}, v\overline{u})$ is matched. Thus $uv$ is unmatched but expanded, whence $uv = \eta(B_u)$ in graph $G$. In $G$, $B_{uv}$ is the smallest blossom containing $B_v$ and its base edge. Thus, in $G$, $uv$ is in the closed path $C(B_{uv})$ that defines blossom $B_{uv}$. The expansion of that cycle contains the middle edge $(u\overline{v}, v\overline{u})$. In other words $w = u\overline{v}$ is in the closed path $C(B_{uv})$. This negates the supposition of the lemma ($w \notin C(B_w)$), contradiction.

Suppose the end edge $(u, u\overline{v})$ is matched. If $u$ is an atom of $B_{uv}$ the first alternative of the lemma holds (since $B_{uv} = B_{u\overline{v}} = B_w$). If $u$ is in blossom $B_u$ then $uv$ is the base edge (since $uv$ is a matched edge not in $I(B_u)$). So as in the previous case the supposition of the lemma fails.

As in 1-matching we analyze DismantlePath($Q$) using a matching $M_\omega$ on $Q$ that comes from the previous scale. We show how to construct $M_\omega$ for $f$-factors in Lemmas 5.3–5.4 below. To set the stage we first review the construction of $M_\omega$ for 1-matching. Then we overview the analogous construction for $f$-factors, and finally we prove the two lemmas.
Section 3.4 uses the blossom structure to derive $M_\omega$ from the algorithm’s matching at the end of the previous scale. In detail, let $M$ be the algorithm’s perfect matching at the end of the previous scale. The analysis requires a matching $M_\omega$ that has a certain vertex $\omega$ free. $M_\omega$ is constructed by rematching $M$ along an alternating path $P(\omega, \beta(Q))$, that starts with the matched edge incident to $\omega$ and ends at the base vertex of $Q$. Gabow [11] shows how to construct the $P(\cdot, \cdot)$ paths from the blossom structure.

We proceed similarly for $f$-factors as follows. Consider an mpr $Q$ and its major path $P(Q)$, both as defined in $\overline{G}$. As with 1-matching we are given an arbitrary $G$-vertex $\omega \in Q$. The task is to construct an $f$-factor $M_\omega$ that makes $Q$ a light blossom with base vertex $\omega$. We construct $M_\omega$ by rematching $\overline{M}$ (the natural matching from the previous scale) along an alternating trail, called a $\overline{P}$ trail.

The $\overline{P}$ trails are of two types, $\overline{P}_M(x, \beta)$ and $\overline{P}_U(x, \beta)$. We use the notation $\overline{P}_\mu(x, \beta)$, $\mu = M, U$. By definition the first edge of $\overline{P}_\mu(x, \beta)$ has M-type $\mu$, i.e., a $\overline{P}_M(\cdot, \beta)$ trail begins with a matched (unmatched) edge. The $\overline{P}$ trails are in graph $\overline{G}$, so the matching is $\overline{M}$.

The $\overline{P}$ trails are analogs of the trails $P_1(v, \beta)$ for $f$-factors from Gabow [10] (see Appendix C). These $P_1$ trails are in turn analogs of the $P$ trails of 1-matching [11]. Expansion vertices complicate the construction of $\overline{P}$. For example in Fig. 20b there is no possible trail $\overline{P}_U(u\bar{v}, \beta(B) = u\bar{v})$.

The next lemma constructs the $\overline{P}$ trails. (The lemma is a simple extension of the construction in [10], but we include the details here for completeness.) Recall $M_\omega$ is a matching on an mpr $Q$. We can assume $Q \neq V$. (The analysis of mpr $Q = V$ requires no rematching—the unique shell $(V, \emptyset)$ has both boundaries uncrossed, so any matching on $V$ can be used for the analysis.) Let $\overline{\beta}$ be the e-base vertex of the e-blossom $Q$.

We require the following properties for the $\overline{P}$ trails, precise analogs of properties of the $P_1$ paths:

(a) $\overline{P} = \overline{P}_\mu(x, \overline{\beta})$, $\mu \in \{M, U\}$ is an alternating $x\overline{\beta}$-trail contained in $B_{x, \overline{\beta}}$.
(b) $\overline{P}$ is composed of $\Omega^-$-blossom edges from $G$ and expansion edges from $\overline{G}$. Its last edge is unmatched.
(c) Any e-blossom $B$ containing a vertex of $\overline{P}$ has $|\delta(B, \overline{P})| \leq 2$, and $\overline{P}$ contains either $\overline{\beta}$ or $\overline{\eta}(B)$.

In property (a) note there is no danger of interpreting $x\overline{\beta}$ as an e-vertex! For some examples let $x$ be either of the two e-vertices in $S^-$ in Fig. 18. The last part of property (b) is needed to rule out $P_M(x, \beta(S^-))$ being the single matched edge $(x, \beta)$. (Rematching this trail creates two free vertices, but $M_\omega$ must have only one.)

Regarding property (c), unlike [10] (c) may be impossible: In Fig. 18 any trail $\overline{P}_U(x, \overline{\beta})$ must leave $S^-$ on an unmatched edge, violating (c). Also in Fig. 20b for $x = u\bar{v}$ a trail $P_U(x, x)$ does not even exist. These examples illustrate the exceptions in the following lemma.

**Lemma 5.3** For any e-blossom $B$ and any $\overline{G}$-vertex $x \in B$, the alternating trails $\overline{P}_\mu(x, \overline{\beta}(B))$ for $\mu \in \{M, U\}$ exist unless

$$x = \overline{\beta}, \text{ or } x \text{ an e-vertex and } \mu = U.$$
Proof We give a recursive definition of the desired trails. Let the sequence of e-blossoms that contain $v$ and are subsets of the minimal e-blossom containing both $v$ and $\beta$ be $B_i$, $i = 1, \ldots, k$, where $B_1 = B_v$, $B_k = B_\beta$. We will define the edges of $P$ in each set $E(B_i) - E(B_{i-1})$, taking $E(B_0)$ to be $\emptyset$.

We inductively assert that for every $i$, the edges of $P$ in $E(B_i)$ form a trail $\overline{P}_\mu(v, \overline{P}(B_i))$ that satisfies properties (a)–(c). Clearly this assertion for $i = k$ shows the construction is correct.

We start by giving the inductive step $i > 1$. Then we give the base case $i = 1$. (Note that the base case has some overlap with the inductive step, specifically it treats the path $CP$ exactly as presented for the inductive step.)

Inductively assume the portion of $P$ in $E(B_{i-1})$ ends at the $G$-vertex $x = \beta(B_{i-1})$. $P$ follows a trail from $x$ to $\overline{P}(B_i)$ in $E(B_i) - E(B_{i-1})$. If $x = \overline{P}(B_i)$ we are done. Otherwise, recall that blossoms $B$ have a closed path $C(B)$, whose vertices are either atoms or maximal subblossoms of $B$ (see Appendix C).

We first define a subset $CP^+$ of edges that belong to $P$. $CP^+$ starts with a path of edges $CP \subseteq C(B_i)$. $CP$ ends at the image of $\beta(B_i)$ in $C(B_i)$. If that image is a blossom then $CP^+ = CP$. Suppose the image is atomic. If $B_i$ is a light blossom then $\beta(B_i) = \overline{P}(B_i)$ and $CP^+ = CP$. If $B_i$ is heavy then $\beta(B_i) \neq \overline{P}(B_i)$ and $CP^+$ is $CP$ extended with the edge $(\beta(B_i), \overline{P}(B_i))$.

$CP$ starts with edge $\overline{P}(B_{i-1})$. This edge alternates with the last edge of the trail in $B_{i-1}$ by property (b). After $\overline{P}(B_{i-1})$ $CP$ follows the edges of $C(B_i)$ that continue in the direction avoiding $B_{i-1}$. It ends at the image of $\beta(B_i)$.

Next we specify how $P$ traverses $CP$ (i.e., how it traverses the maximal subblossoms of $C(B_i)$ on $CP$). Consider a maximal subblossom $A$ on $CP$. First assume $A$ is an interior vertex of $CP$. So $\delta(A, CP) = \delta(A, C(B_i)) = \{\overline{P}(A), \theta(A)\}$. Here $\theta(A)$ is an edge $st$ where $s$ and $t$ are $G$-vertices, $s \in A \neq t$ and $st$ unmatched (recall Lemma 5.1). Define $P$ to contain the trail $P_M(s, \overline{P}(A))$. It alternates with $st$ at $s$. Also it exists even if $s$ is an e-vertex (i.e., it is not a $P_U$ trail). It is clear that properties (a)–(c) are preserved by including $P_M(s, \overline{P}(A))$ in $P$.

Now assume the subblossom $A$ is the end of $C(B_i)$. ($A$ is the image of $\beta(B_i)$.) Note that $B_{i-1}$ is disjoint from $A$ (otherwise $B_{i-1} \subseteq A \subset B_i$, contradicting the definition of $B_i$ as the minimal blossom containing $B_{i-1}$.) Although $A$ has two $\theta(A)$ edges, this implies only one belongs to $CP$. It is treated like $st = \theta(A)$ above. Again (a)–(c) are preserved, by the inductive hypothesis for $A$. (Note that (c) always holds for $B = B_i$ since it has $\delta(B, \overline{P}) = \emptyset$ and $\beta(B) = \overline{P}(B_i)$.)

It remains to consider the case where the end of $C(B_i)$ is atomic, i.e., $\beta(B_i)$ is atomic. If $B_i$ is a light blossom then $\beta(B_i) = \overline{P}(B_i)$ and $CP^+ = CP$. The last part of (b) holds since $CP$ ends with an edge of $e_B$, which is unmatched. (Recall $e_B$ from the definition of blossom. Definition C.1) If $B_i$ is heavy then $\beta(B_i) \neq \overline{P}(B_i)$ and $CP^+$ is $CP$ extended with the edge $(\beta_i, \overline{P_i})$. The last edge of $CP$ is an $e_B$ edge, so it is matched. It alternates with the unmatched edge $(\beta_i, \overline{P_i})$. So (a) and (b) hold.

We omit the treatment of a subblossom $A$ that is the first end of $CP$, since it is $B_{i-1}$.

We turn to the base case $i = 1$. $B_1 = B_v$, so $v$ is atomic. Let $\beta$ and $\overline{P}$ be defined for $B_v$. 
We first assume $v$ is a vertex ($G$- or $e$-) in $C(B_v)$ or $v$ is the $e$-base of a heavy blossom $B_v$. Consider the two possible types of blossoms $B_v$.

**Case $B_v$ is light.**

*Subcase $v \neq \beta$: $C(B_v)$ alternates at $v$. $CP$ starts with the edge of $\delta(v, C(B_v))$ of M-type $\mu$. The rest of analysis is identical to the inductive step.*

*Subcase $v = \beta = \beta$*: The $PM$ trail for $v$ has no edges in $B_v$. (In $B_2$ the trail starts with the matched edge $\eta(B_v)$. There is no trail for $v = \beta$ in $B_k$, an exceptional case of the lemma, so we can assume $k > 1$.) The $PU$ trail for $v$ is constructed by taking $CP^+$ to be the entire closed path $C(B_v)$, starting and ending at $v$.

**Case $B_v$ is heavy:** This case is similar to the previous case.

*Subcase $v \neq \beta, \beta$: $CP$ starts with the edge of $\delta(v, C(B_v))$ of M-type $\mu$. This path ends at $\beta$, so as before $CP^+$ adds the edge $(\beta, \beta)$.*

*Subcase $v = \beta$: The $PU$ trail for $v$ is edge $(\beta, \beta)$. The $PM$ trail is constructed by taking $CP^+$ to be the entire closed path $C(B_v)$ (starting and ending at $v$), and appending $(\beta, \beta)$.*

*Subcase $v = \beta$: The $PM$ trail for $v$ has no edges in $B_v$. The $PU$ trail does not exist ($v$ is an $e$-vertex).*

It remains to treat the possibility that $v$ is an $e$-vertex, $v \notin C(B_v) \cup \beta$. Lemma 5.2 shows the matched edge of $v$ joins it to a $G$-vertex $w \in B_v$. So $\overline{P} = \overline{PM}$ consists of edge $vw$ followed by the $PU$ trail for $w$. Clearly the latter trail does not contain $vw$ (in $B_v$ that trail consists of edge $(\beta, \beta)$ if it exists and edges in $C(B_w)$). (a)–(c) are preserved. The $PU$ trail for $v$ does not exist. $\square$

We can now define the $f$-factor $M_\omega$ for the analysis. Let $B_\omega$ denote the minimal blossom of $P(Q)$ that contains a vertex that is free at the chosen instant of time of the analysis. Let $\omega \in B_\omega$ be such a free vertex. Let $\overline{\beta}$ be the $e$-base vertex of $Q$. Define

$$M_\omega = (\overline{M} \oplus \overline{PM}(\omega, \overline{\beta})) \cap \gamma(Q),$$

where we interpret the term $\overline{PM}$ term to be $\emptyset$ if $\omega = \overline{\beta}$.

**Lemma 5.4**

(i) Every $e$-blossom $B$ has $|\gamma(B, M_\omega)| = \lceil f(B)/2 \rceil$.

(ii) Every $e$-blossom $B$ containing $\omega$ has $\delta(B, M_\omega) = \emptyset$.

(iii) The duals $y_0, z_0$ continue to satisfy (4.10) in $\overline{G}$ with matching $M_\omega$, i.e.,

$$\overline{y}_0 \overline{z}_0(e) = w(e) + \Delta_0(e), \text{ where } \Delta_0(e) \begin{cases} \geq -2 & e \notin M_\omega \\ \leq 12 & e \in M_\omega \\ \in [-2, 12] & e \in \cup\{E(B) : B \text{ a blossom}\} \cup ET. \end{cases}$$
Proof We will use properties (a)–(c) for $\overline{P} = \overline{P}_M(\omega, \bar{\beta})$.

(i) Every vertex of $Q$ is perfectly matched in $M_\omega$ except for $\omega$ which lacks one matched edge [properties (a) and (b)]. Any e-blossom $B$ has $f(B)$ odd. So (i) holds for $B$ when either

1. $\omega \in B$ and $\delta(B, M_\omega) = \emptyset$, or
2. $\omega \notin B$ and $|\delta(B, M_\omega)| = 1$.

Consider the four possibilities for $B$ depending on vertices $\omega$ and $\beta$:

Case $\omega \in B \notin \bar{\beta}$: We show (1) holds. (c) Shows $|\delta(B, \overline{P})| \leq 2$ and $\overline{P}$ contains $\overline{\eta}(B)$. $\overline{P}$ leaves $B$ an odd number of times, so we get $\delta(B, \overline{P}) = \{\overline{\eta}(B), \theta(B)\}$. Rematching $\overline{P}$ makes $\overline{\theta}(B)$ matched. This gives (2).

Case $\beta \in B \notin \omega$: We show (2) holds. (c) shows $|\delta(B, \overline{P})| \leq 2$. $\overline{P}$ crosses $B$ an even number of times, so we get $\delta(B, \overline{P}) = \{\overline{\theta}(B)\}$. This gives (1).

Case $\omega, \beta /\in B$: We show (2) holds. (c) Shows $|\delta(B, \overline{P})| \leq 2$ and $\overline{\eta}(B) \in \overline{P}$. $\overline{P}$ crosses $B$ an odd number of times, so we get $\delta(B, \overline{P}) = \{\overline{\eta}(B), \theta(B)\}$. Rematching $\overline{P}$ makes $\overline{\theta}(B)$ matched. This gives (2).

Case $\omega, \beta \in B$: We show (1) holds. (a) Shows $\delta(B, \overline{P}) = \emptyset$. Thus $\delta(B, M_\omega) = \delta(B, \overline{M}) \cap \gamma(Q)$. The latter is $\emptyset$, since $\overline{\eta}(B) \in \delta(Q)$ leaves $\bar{\beta}$. This gives (1).

(ii) $\omega \in B$ iff the first or last case above holds. Both satisfy (1).

(iii) The proof of Proposition 4.7 applies to the edges of the natural matching $\overline{M}$ (i.e., we can ignore the changes in the matching of Fig. 19).

Thus we need only consider edges $e \in \overline{P}_M(\omega, \bar{\beta})$. Recall the construction of this trail. Suppose $e$ is in the $CP^+$ set of some blossom $B$. So $e \in C(B)$ or $e = (\beta(B), \bar{\beta}(B))$. Recall $(\beta(B), \bar{\beta}(B)) \in ET$. Thus $e$ satisfies the last case of (4.10) (both before and after it gets rematched).

The remaining possibility for $e$ is that it is the first edge $vw$ of a trail $\overline{P}_M(v, \bar{\beta})$, $v$ an e-vertex (recall the last case of the construction of $\overline{P}$ for $i = 1$). This makes $v = \omega$. We claim $vw$ is not undervalued. The claim implies $\Delta_0(vw) \geq -2$, so (4.10) holds when $vw$ becomes unmatched in $M_\omega$.

To prove the claim assume $vw$ is undervalued (in the duals $y_0, z_0$). Figure 19 matches $vw$ right before the $f$-DISMANTLER begins. So the e-vertex $v$ remains matched throughout the execution of the $f$-DISMANTLER. But as noted $v$ is the free vertex $\omega$, contradiction. \qed

5.2 Notions of the f-Factor Scaling Inequality

This section introduces notation and states the scaling inequality for $f$-factors. To further prepare for the proof it derives preliminary properties of the objective function.

In this section we omit all $\phantom{o}^-$ overlines, for notational simplicity. For example we write $f$ instead of $\overline{f}$. Also since the graph is $\overline{G}$ the inherited blossoms are e-blossoms. We remind the reader of this at important points.
At times it is convenient to view sets of vertices as multisets. For a shell $S$ let $\overline{S}$ be the multiset where each vertex $v \in S$ has multiplicity $f(v)$. Let $F$ denote the set of free vertices of $S$ where each $v \in F$ has multiplicity equal to its deficiency, i.e., $f(v) - \deg(v)$ for $\deg(v)$ the degree of $v$ in the current matching $M$. As an example $y(S) = \sum_{v \in S} y(v)$ but $y(\overline{S}) = \sum_{v \in S} f(v) y(v)$.

A shell $S$ is even if $f(S)$ is even. We consider two types of shells, depending on whether $S^-$ is a blossom or $\emptyset$. Suppose $S^-$ is a blossom. $S$ is an even shell if neither boundary of $S$ is crossed by a matched edge (the usual choice of $S$).

Suppose $S^- = \emptyset$. We consider only one shell of this type, denoted as $S = (B_\omega, \emptyset)$.

The precise choice of blossom $B_\omega$ is given below. We treat this shell as a degenerate case, with special notation. It uses a modified graph. By definition shell $S$ is the vertex set $B_\omega - \emptyset = B_\omega$. $\omega$ will always be chosen as a vertex $x$ of $\overline{G}$ that is free at the chosen point in time of the analysis. The shell $(B_\omega, \emptyset)$ is formed by decreasing $f(x)$ by 1 and decreasing the deficiency of $x$ by 1. In this modified graph shell $S$ has $f(S)$ even, as desired. We refer to the modified graph via the notation $\omega, \omega^-$ and $\omega^+$ for three versions of the same vertex, distinguished as follows:

- $\omega$ is the chosen vertex of $\overline{G}$.
- $\omega^-$ is its replacement in the modified graph, characterized by $f(\omega^-) = f(\omega) - 1$.
- $\omega^+$ is one copy of $\omega$ in its multiset representation, and this copy is free in the current matching.

$\omega^+$ is viewed as the copy of $\omega$ that gets deleted while the remaining $f(\omega) - 1$ copies remain in the graph. Note that $\omega^-$ may be on an arbitrary number of matched edges (between 0 and $f(\omega^-)$ matched edges). $\omega^-$ is in the modified graph even if its $f$ value is 0. (Such vertices serve no function in other $f$-factor algorithms but are used in the “Objective functions” section below.)

Throughout this section we point out other modifications for the degenerate shell $(B_\omega, \emptyset)$. Call a shell $S$ nondegenerate if $S^-$ is a blossom. The appropriate interpretations for the degenerate case always make it similar to the nondegenerate case.

As another example let $S$ be a shell and $M$ a matching contained in $\gamma(S)$ with free vertex set $F \subseteq S$. Any set $B \subseteq V$ satisfies

$$f(S \cap B) = |\overline{F} \cap B| + 2|\gamma_M(B)| + |\delta_M(B)|.$$  \hspace{1cm} (5.2)

This follows since each of the $f(v)$ copies of $v \in S \cap B$ is either free or matched. The identity holds for degenerate $S$ with the obvious interpretation of $f(S \cap B)$. (Note that the set $S \cap B$ is interpreted as a set in the modified graph. We always have $S \cap B \subseteq V - \omega + \omega^-$.)

**The setup** As for 1-matching we choose an arbitrary point in the execution of Phase 1 or 2 of DISMANTLEPATH($X$), and analyze a shell $S$ of a major path $P(Q)$, $Q \subseteq X$.

---

\textsuperscript{6} This restriction is only for notational convenience. The properties of any shell $(S^+, \emptyset)$ are essentially the same.
The setup is entirely analogous to Sect. 3.2 (although the graph is $\overline{G}$, so $Q \neq V$ is an e-blossom rather than a standard blossom). We describe it in the next several paragraphs.

Let $M_Q$ be the current matching on $Q$. We analyze an even shell $S = (C, D)$ of $P(Q)$ whose boundaries are not crossed by $M_Q$.

Suppose $Q \neq V$, i.e., $Q$ is a blossom. For the degenerate shell $(B_\omega, \emptyset)$ choose $B_\omega$ as the smallest blossom of $P(Q)$ that is uncrossed by $M_Q$. (Assume this exists. If not there is no scaling inequality for a shell of $Q$.) Let $\omega \in B_\omega$ be a free vertex of $M_Q$. (\omega exists since $f(B_\omega)$ is odd, as $B_\omega$ is a blossom, and no matched edge crosses $B_\omega$. When there is more than one candidate for $\omega$ choose arbitrarily.)

The scaling inequality To state the inequality recall the graph is the expanded graph $\overline{G}$ and blossoms are e-blossoms. Let $S$ be an even shell of a major path $P(Q)$, $S$ either nondegenerate or $S = (B_\omega, \emptyset)$. Consider an arbitrary point in time during the execution of DismantlePath$(X)$ for $X \supseteq Q$. Assume the current matching on $X$ does not cross a boundary of $S$; let $M \subseteq \gamma(S)$ be the current matching restricted to $S$. Let $F \subseteq S$ be the set of free vertices of $M$ with $\overline{F}$ the corresponding multiset. The $f$-factor version of (3.4) is

$$d(\overline{F}) + \kappa_0(INT(S)) + \kappa_1(SUB(S)) \leq cn + \tau(U)$$

(5.3)

where all terms are defined as in Sect. 3 except for the multiset $\overline{F}$ and

$$c = 8$$

$$n = f(S).$$

The setup continued $y$, $z$, and $u$ denote the algorithm’s current duals (reviewed in Appendix C). Note that $\omega \notin F$ even though it is free. $z$ can have positive values on e-blossoms and $\Omega$-blossoms. As before an $\Omega$-blossom can cross $S^+$ or $S^-$ on unmatched edges. This includes unmatched $I(B)$ edges.

$y_0$, $z_0$, $u_0$ are the dual functions on entry to the $f$-Dismantler. (Hence $z_0$ deals with e-blossoms.) The matching $M_\omega$ corresponding to these duals is chosen as follows.

If $S = V$ then $M_\omega$ is the $f$-factor on $V$ after edge expansion. ($M_\omega$ has the natural matching.) If $S \neq V$ then $Q$ is a blossom. Take $M_\omega$ to be the $f$-factor of Lemma 5.4 that has free vertex $\omega$, restricted to $S$. Lemma 5.4(ii) shows $M_\omega$ does not cross any e-blossom containing $\omega$. So $M_\omega$ does not cross the shell $S$ of the analysis.

Note that $\omega$ is free in the two matchings of interest,

$$\delta(\omega, M_Q) = \delta(\omega, M_\omega) = \emptyset,$$

so $\omega$ is essentially irrelevant to properties concerning matchings.

As before every free vertex $v$ satisfies

$$y(v) = y_0(v) - d(v) + \sum_{v \in B} \tau(B).$$

(5.4)
Consider a shell $S$ and an e-blossom $B$ with $B \cap S \neq \emptyset$. As before the laminarity of $\Omega^e$ implies
\begin{equation}
B \subseteq S \text{ or } B \in \text{INT}(S) \text{ or } C \subseteq B. \tag{5.5}
\end{equation}

The alternatives are mutually exclusive for nondegenerate $S$, and as before they can be collapsed to
\begin{equation*}
B \subseteq S \text{ or } D \subseteq B.
\end{equation*}

For degenerate $S$ we make the three cases mutually exclusive by narrowing the first case to $B \subseteq S - \text{INT}(S)$. In the collapsed version the second alternative is $B_\omega \subseteq B$. In proof $B \cap S \neq \emptyset$ implies as sets of $\overline{G}$, $B \cap B_\omega \neq \emptyset$. Laminarity implies $B \subseteq B_\omega$ or $B_\omega \subseteq B$. If $B \subseteq B_\omega$ in $\overline{G}$ then the same holds in the modified graph.

**Objective functions** We give the details of the two dual functions, $y_0, z_0$ and $y, z$, and their objective functions.

Recall the discussion of $f$-factors duals in Appendix C. We extend the general definition of objective function to shells $S$ by the relation
\begin{equation}
\hat{y}z[S] = y(S) + \sum_B z(B) c(S \cap B) + u(\gamma(S)), \tag{5.6}
\end{equation}

where the “capacity” function $c$ is the number of edges $e \in \gamma(S)$ that can include $z(B)$ in their dual value $\hat{y}z(e)$. (Also recall the dual function $u$ that is used for undervalued edges, i.e., optimum duals have $u(e) = w(e) - \hat{y}z(e)$ if $e$ is not dominated.) We specify this capacity function first for the previous duals $y_0, z_0$ and then for the current duals $y, z$. Note the $u$ function for undervalued edges is not included in $\hat{y}z(e)$ (see Appendix C).

The duals $y_0, z_0$ from the previous scale use e-blossoms to define $z$. Such an e-blossom contributes to $\hat{y}z(e)$ if $e \in \gamma(B) \cap \gamma(S)$ (this set is identical to $\gamma(S \cap B)$). So the capacity function is
\begin{equation*}
c(S \cap B) = \lfloor f(S \cap B)/2 \rfloor \quad \forall B \in \Omega^e.
\end{equation*}

Clearly any matching contains $\leq c(S \cap B)$ edges of $\gamma(S \cap B)$ (even for degenerate $S$).

We show the expanded matching of the previous scale $M_\omega$ achieves this upper bound, i.e.,
\begin{equation}
\gamma(S \cap B, M_\omega) = c(S \cap B). \tag{5.7}
\end{equation}

In proof recall (5.5) and observe that when $S \cap B \neq \emptyset$,
\begin{equation}
|\gamma(S \cap B, M_\omega)| = \begin{cases} 
\lfloor f(B)/2 \rfloor & B \subseteq S \\
f(S \cap B)/2 & B \in \text{INT}(S) \\
f(S)/2 & C \subseteq B.
\end{cases} \tag{5.8}
\end{equation}
This holds for all $S$ (degenerate or nondegenerate). It follows since $M_\omega$ is a perfect matching of $\overline{G} - \omega$ that does not cross a boundary of $S$. (This is trivial for $S = (V, \emptyset)$.) All three quantities equal $\lceil f(S \cap B)/2 \rceil$.

Turning to the current duals $y, z$ we consider two cases. First we show the matching $M$ in the $f$-DISMANTLER satisfies

$$c(S \cap B) = |\gamma_M(B)| + |\overline{F} \cap B|/2 \quad \text{if } B \in \Omega^c \text{ is undissolved.} \quad (5.9)$$

As with 1-matching the case “$B \in \Omega^c$ dissolved” is irrelevant.

Suppose $B \in \Omega^c$ is undissolved at the chosen instant of time. Either $B \in INT(S)$ or $S \subseteq B$. In both cases $S \cap B$ is an even shell. Thus $c(S \cap B) = f(S \cap B)/2$. No matched edge crosses $B$ (undissolved), i.e., $\delta_M(B) = \emptyset$. So (5.2) gives (5.9).

Now consider the contribution of $\Omega$-blossoms to the current dual function $y, z$. An $\Omega$-blossom $B$ contributes to $\tilde{y}(e)$ if $e \in \gamma(B) \cup I(B)$. Thus

$$c(S \cap B) = \left\lfloor \frac{f(S \cap B) + |\gamma(S, I(B))|}{2} \right\rfloor \quad \text{for } B \in \Omega.$$

Recall that an $\Omega$-blossom $B$ can cross $S$ on unmatched edges. The same holds for an unmatched edge in $I(B)$ (i.e., $\eta(B)$). The capacity function only concerns edges with both ends in $S$, hence we use the restrictive term $\gamma(S, I(B))$.

Note also that in $\overline{G}, \omega$ may be the nonblossom end of an unlimited number of unmatched base edges $\eta(B) \in I(B)$. This holds even if $f(\omega^-) = 0$, which is why we keep such vertices in the modified graph.

We proceed to validate the definition of capacity. Clearly any matching contained in $\gamma(S)$ contains $\leq c(S \cap B)$ edges of $\gamma(B) \cup I(B)$. We will show the matching $M$ achieves this upper bound, i.e.,

$$c(S \cap B) = |\gamma_M(B)| + |I(B) \cap M| \quad (5.10)$$

Equivalently we show

$$f(S \cap B) + |\gamma(S, I(B))| = 2(|\gamma_M(B)| + |I(B) \cap M|) + \epsilon, \quad 0 \leq \epsilon \leq 1. \quad (5.11)$$

By (5.2) and the definition of $I(B)$, the left-hand side of (5.11) is

$$\left(|\overline{F} \cap B| + 2|\gamma_M(B)| + |\delta_M(B)| + (|I(B) \cap M| + |\eta(B) \cap \overline{M} \cap \gamma(S)|)\right).$$

Using $\delta_M(B) = |I(B) \cap M| + |\eta(B) \cap M|$ we rewrite this as

$$2(|\gamma_M(B)| + |I(B) \cap M|) + \left(|\eta(B) \cap M| + |\eta(B) \cap \overline{M} \cap \gamma(S)| + |\overline{F} \cap B|\right).$$

We complete the proof by showing $\epsilon \leq 1$. Let $\beta$ be the base vertex of $B$. 
If $\beta$ is free then $B$ has no base edge. So $\epsilon = |F \cap B| \leq 1$. If $\beta$ is not free then $\eta(B)$ exists, and is either matched or unmatched. It may not be in $\gamma(S)$, so again $\epsilon \leq 1$.

### 5.3 Proof of the Scaling Inequality

The argument is that of Sect. 3.4 with simple modifications to incorporate e-blossoms and the $f$ function. As before the first four steps apply the two dual functions $y_0, z_0$ and $y, z$, and the last step analyzes crossings of e-blossoms.

The initial duals $y_0, z_0$ satisfy

$$\hat{y_0 z_0}[S] = \hat{y_0 z_0}(M_\omega).$$

This follows since $M_\omega$ is a perfect matching on $S$, (5.7) shows its $z_0(B)$ contributions are at capacity, and we use the obvious definition of $u_0$. Every edge $e \in M_\omega$ has $\hat{y_0 z_0}(e) \leq w(e) + 12$ [Lemma 5.4(iii)]. Summing these inequalities, with the previous inequality, gives

$$\hat{y_0 z_0}[S] - 12(n/2) \leq w(M_\omega) \text{ relaxed near tightness of } \hat{y_0 z_0}.$$  

An edge $e \in M_\omega$ is nearly dominated by the $y, z$ duals, unless it is undervalued. $M_\omega$ fills each $\Omega$-blossom to at most its capacity. So

$$w(M_\omega) - 2(n/2) \leq \hat{y z}(S) + u(M) \text{ near domination of } \hat{y z}.$$  

Every edge $e \in M$ is nearly tight in the $y, z$ duals. Equation (5.10) shows $M$ fills each $\Omega$-blossom to its capacity. So

$$\hat{y z}(S) + u(M) \leq y(F) + w(M) + \sum_{B \in \Omega^e} z(B)|F \cap B|/2 \text{ near tightness of } \hat{y z}.$$  

Recall that the quantity $z(B)$ in the summation for $B \in \Omega^e$ only counts contributions coming from $z_0(B)$ (it does not count contributions made after $B$ is dissolved and $B$ enters $\Omega^-$).

To summarize at this point, combine the last 3 inequalities to get

$$\hat{y_0 z_0}[S] - 7n \leq y(F) + w(M) + \sum_{B \in \Omega^e} z(B)|F \cap B|/2. \quad (5.12)$$

To upper bound the right-hand side of (5.12) first sum (5.4) for every $v \in F$:

$$y(F) = y_0(F) - d(F) + \sum_B |F \cap B| \tau(B),$$
and then bound the matched edges by

\[ w(M) \leq y_0(\overline{S} \cap V(M)) + \sum_B z_0(B)\gamma_M(B) \]

\[ + u_0(M_\omega) + 2(n/2) \quad \text{near domination of } \overline{y_0z_0}. \]

Combining the last two inequalities gives

\[ y(\overline{F}) + w(M) \leq y_0(\overline{S}) - d(\overline{F}) + \sum_B \left( \overline{|F \cap B|}\gamma_M(B) + z_0(B)\gamma_M(B) \right) + u_0(M_\omega) + n. \]

(5.13)

### 5.3.1 Analysis of Blossom Crossings

This section is almost identical to the case of 1-matching, with simple changes for the \( f \) function. (This is unsurprising since the analysis concerns e-blossoms, which are simple versions of ordinary matching blossoms.) We include the details for completeness.

As before we will show

\[ \beta \leq \sum_B z_0(B)\lfloor f(B \cap S)/2 \rfloor - \sum_{B \in \Omega^e} |\overline{F \cap B}|z(B)/2 + \Delta \]  

(5.14)

where \( \Delta \) consists of the \( \tau \) and \( \kappa_\epsilon \) terms of (5.3),

\[ \Delta = \tau(\mathcal{U}) - \kappa_0(\text{INT}(S)) - \kappa_1(SUB(S)). \]

Figure 21 illustrates the latter two for \( f \)-factors. The blossoms \( B \) contributing to \( \beta \) belong to \( \Omega^e \) (\( \tau(B) \) or \( z_0(B) \) is positive) with \( B \cap S \neq \emptyset \) (\( |\overline{F \cap B}| \) or \( |\gamma_M(B)| \) is positive). Thus they satisfy (5.5). We break this into two possibilities, \( (B \in \text{INT}(S) \) or \( C \subseteq B \)) and \( B \subseteq S - \text{INT}(S) \). We choose the latter when \( S = (V, \emptyset), Q = X \).
Each possibility will have two subcases, and all four of these subcases are similar. In particular they use (5.2) in the form

\[ |\overline{F} \cap B| + 2|\gamma_M(B)| = f(S \cap B) - |\delta_M(B)|. \]

We always have \( z(B)/2 \leq z_0(B)/2 - \tau(B) \) (equality holds during Phases 1 and 2).

**Case** \( B \in INT(S) \) or \( C \subseteq B \): By assumption \( S \neq V \) so \( Q \) is a blossom. If \( B \in INT(S) \) then \( f(S \cap B) \) is even. If \( C \subseteq B \) then \( S \cap B = S \) and again \( f(S \cap B) \) is even. We conclude

\[ f(S \cap B)/2 = \lfloor f(S \cap B)/2 \rfloor \quad (5.15) \]

in this case.

**Subcase** \( z(B) = 0 \): Using \( \tau(B) \leq z_0(B)/2 \) the term for \( B \) in \( \beta \) is at most

\[ z_0(B)(|\overline{F} \cap B| + 2|\gamma_M(B)|)/2 = z_0(B)(f(S \cap B) - |\delta_M(B)|)/2 \leq z_0(B)f(S \cap B)/2. \]

Using (5.15) shows the upper bound matches the term for \( B \) in the first summation of (5.14). There is no contribution to the second summation when \( z(B) = 0 \).

**Subcase** \( z(B) > 0 \): \( B \) is undissolved, so either \( B \in P(Q) \) or \( Q \subset B \). When \( B \in P(Q) \) the unit translations of DismantlePath(\( Q \)) in Phases 1 and 2 maintain the invariant \( z_0(B)/2 = \tau(B) + z(B)/2 \). This holds trivially in the second case \( Q \subset B \), since \( \tau(B) = 0 \), \( z(B) = z_0(B) \).

Rearranging the invariant to \( \tau(B) = z_0(B)/2 - z(B)/2 \) shows \( B \)'s term in \( \beta \) is

\[ |\overline{F} \cap B|\tau(B) + z_0(B)|\gamma_M(B)| = (z_0(B)/2)(|\overline{F} \cap B| + 2|\gamma_M(B)|) - (z(B)/2)|\overline{F} \cap B| \]

\[ = z_0(B)(f(S \cap B) - |\delta_M(B)|)/2 - (z(B)/2)|\overline{F} \cap B|. \]

Using (5.15) and rearranging terms changes the last line to

\[ z_0(B)\lfloor f(S \cap B)/2 \rfloor - (z(B)/2)|\overline{F} \cap B| - |\delta_M(B)|z_0(B)/2. \]

The first two terms match the terms for \( B \) in the two summations of (5.14). The third term, \(-|\delta_M(B)|z_0(B)/2\), is nonzero only when \( B \in INT(S) \). (In proof \( B \notin INT(S) \) in this case makes \( C \subseteq B \). So \( S \subseteq B \) and no edge of \( M \) crosses \( B \).) We include this third term in the \( \kappa_0 \) quantity of \( \Delta \).

**Case** \( B \subset S - INT(S) \):

**Subcase** \( B \) crossed: Again using \( \tau(B) \leq z_0(B)/2 \), the term for \( B \) in \( \beta \) is at most

\[ z_0(B)(|\overline{F} \cap B| + 2|\gamma_M(B)|)/2 = z_0(B)(f(B) - |\delta_M(B)|)/2. \]
Substituting $f(B)/2 = \lfloor f(B)/2 \rfloor + 1/2$, the displayed quantity becomes
\[ z_0(B)\lfloor f(B \cap S)/2 \rfloor + z_0(B)(1 - |\delta_M(B)|)/2. \]

The first term of the last line matches the term for $B$ in the first summation of (5.14). $B$ is dissolved in this case, so it does not occur in the second summation. Recalling the assumption $|\delta_M(B)| \geq 1$, we include the second term, $-z_0(B)(|\delta_M(B)| - 1)/2$, in the $\kappa_1$ quantity of $\Delta$.

**Subcase B uncrossed**: This implies $B$ contains a free vertex since $f(B)$ is odd. Thus $|\overline{F} \cap B|\tau(B) \leq \tau(B) + (|\overline{F} \cap B| - 1)z_0(B)/2$. Using this and proceeding as before, $B$’s term in $\beta$ is
\[ |\overline{F} \cap B|\tau(B) + z_0(B)|\gamma_M(B)| \leq \tau(B) + z_0(B)(f(B) - 1)/2 = \tau(B) + z_0(B)\lfloor f(B \cap S)/2 \rfloor. \]

In the last displayed line the second term matches the term for $B$ in the first summation of (5.14). As before there is no contribution to the second summation. We include the first term $\tau(B)$ in $\tau$ quantity of $\Delta$.

Using the upper bound on $\beta$ in (5.13) gives
\[
y(\overline{F}) + w(M) \leq y_0(S) - d(\overline{F}) + \sum_B z_0(B)\lfloor f(B \cap S)/2 \rfloor - \sum_{B \in \Omega^2} |\overline{F} \cap B|z(B)/2 + \Delta + u_0(M_0) + n
\]
\[= y_0z_0[S] - d(\overline{F}) - \sum_{B \in \Omega^2} |\overline{F} \cap B|z(B)/2 + \Delta + n. \]

Combining this with (5.12) gives
\[ \widehat{y}_0z_0[S] - 7n \leq \widehat{y}_0z_0[S] - d(\overline{F}) + \Delta + n. \]

Hence
\[ d(\overline{F}) \leq 8n + \Delta \tag{5.16} \]
as desired.

**Acknowledgements** The author thanks Seth Pettie for helpful conversations regarding Ref. [4]. Thanks also to an anonymous referee for very careful reading and many suggestions to clarify the narrative.

**A Edmonds’ Algorithm for Matching**

For background we first review the linear program for maximum weight perfect matching. The variables are given by the function $x : E \to \mathbb{R}_+$ which indicates whether or not an edge is matched. Recall our summing convention, e.g., $x(\delta(v)) = \sum_{e \in \delta(v)} x(e)$. 
Make every free vertex or blossom the root of an $\mathcal{S}$-tree. Then repeat the following statement until an augmenting path is found.

if $\exists$ eligible edge $e = xy$, $x \in S$, $y \notin S$, alternating with $B_x$ then
  /* grow step */
  add $xy$, $B_y$ to $S$
else if $\exists$ eligible edge $e = xy$, $x, y \in S$, alternating with both $B_x$ and $B_y$ then
  if $x$ and $y$ are in the same search tree then
    /* blossom step */
    merge all blossoms in the fundamental cycle of $e$ in $\mathcal{S}$
  /* if $x$ and $y$ are in different trees an augmenting path has been found */
else if $\exists$ nonsingleton inner blossom $B$ with $z(B) = 0$ then
  Expand($B$)
else
  AdjustDuals

Algorithm Expand($B$):
  in $\mathcal{S}$ replace $B$ by the even length alternating path of subblossoms $B_0, \ldots, B_k$, where
  the unmatched edge entering $B$ enters $B_0$
  the matched edge entering $B$ enters $B_k$
  /* the remaining subblossoms of $B$ are no longer in $\mathcal{S} */$

Algorithm AdjustDuals:
  $\delta_1 \leftarrow \min \{|y(e) - t(e)| : e = xy$ with $x \in S, y \notin S, xy$ alternates with $B_x\}$
  $\delta_2 \leftarrow \min \{|y(e) - t(e)|/2 : e = xy$ with $x, y \in S, xy$ alternates with both $B_x$ and $B_y\}$
  $\delta_3 \leftarrow \min \{|z(B)/2 : B$ an inner blossom of $\mathcal{S}\}$
  $\delta \leftarrow \min \{\delta_1, \delta_2, \delta_3\}$
  For every vertex $v \in S$
    $y(v) \leftarrow y(v) + (\text{if } B_v \text{ is inner then } \delta \text{ else } -\delta)$
  For every blossom $B$ in $\mathcal{S}$
    $z(B) \leftarrow z(B) + (\text{if } B \text{ is inner then } -2\delta \text{ else } 2\delta)$

Fig. 22 Pseudocode for an Edmonds search, for perfect matching with near-optimum duals

\[
\begin{align*}
\text{maximize } & \sum_{e \in E} w(e) x(e) \\
\text{subject to } & x(\delta(v)) = 1 \quad \forall v \in V \\
& x(y(B)) \leq \frac{|B|}{2} \quad \forall B \subseteq V \\
& x(e) \geq 0 \quad \forall e \in E
\end{align*}
\]

The dual LP uses dual functions $y : V \to \mathbb{R}, z : 2^V \to \mathbb{R}_+$. Define $\hat{yz} : E \to \mathbb{R}$ by

\[
\hat{yz}(e) = y(e) + z\{B : e \subseteq B\}. \tag{A.1}
\]

(Note for $e = uv$, $y(e)$ denotes $y(u) + y(v)$ and $z\{B : e \subseteq B\}$ denotes $\sum_{e \subseteq B} z(B)$.)

\[
\begin{align*}
\text{minimize } & y(V) + \sum_{B \subseteq V} \frac{|B|}{2} \cdot z(B) \\
\text{subject to } & \hat{yz}(e) \geq w(e) \quad \forall e \in E \\
& z(B) \geq 0 \quad \forall B \subseteq V
\end{align*}
\]
A precise statement of Edmonds’ algorithm is given in Fig. 22. We briefly summarize the algorithm, although we assume the reader is familiar with a complete treatment.

The search structure $S$ is a forest. Its nodes are contracted blossoms or vertices that are not in blossoms. For simplicity in this section we call the latter “blossoms” as well.

The roots of $S$ are the free blossoms, i.e., the blossoms that contain an unmatched vertex. The forest is alternating, i.e., the edges from a node to its root alternate between being matched and unmatched. The last edge (incident to the root) is necessarily unmatched. If the first edge is unmatched the node is inner. Otherwise (i.e., the first edge is matched or the node is a root) the node is outer.

For $B$ a blossom in $S$, an edge $e \notin S$ alternates with $B$ if $e$ is unmatched for $B$ outer and matched for $B$ inner. (Note if $e$ alternates with $B$ then $B$ can be replaced by an appropriate alternating path so there is an alternating path of edges of $G$ from $e$ to its forest root.)

For any vertex $v$ of $G$, $B_v$ denotes the maximal blossom containing $v$ (or $v$ itself if no bigger blossom exists). Let $M$ denote the current matching. In a grow step, if $B_x$ is outer then $xy \notin M$ and $B_y$ becomes inner; if $B_x$ is inner then $xy \in M$ and $B_y$ becomes outer. A blossom step necessarily creates a new outer node. When an augmenting path is discovered, the algorithm proceeds to augment a maximal collection of disjoint augmenting paths of eligible edges.

In the dual adjustment step, define the “target” $t(e)$ of edge $e$ that is not in the search structure to be the value of $yz(e)$ that makes $e$ eligible, i.e., $t(e)$ is the value of (2.3). (When we execute Edmonds’ algorithm in Phases 2 or 3 the definition of eligibility changes to (2.4) and the target becomes the value closest to the current value of $yz(e)$. For instance an unmatched edge with $yz(e) > w(e)$ has target $t(e) = w(e)$.)

### B Implementation of the Dismantler

This section presents the data structures and implementation details for Phases 1 and 2 of DismantlePath($Q$). We show the total time is $O(\sqrt{|Q| \log |Q| m(Q)})$ for matching on graphs. The discussion applies equally well to $f$-factors (since the $f$-Dismantler is identical to the Dismantler) making obvious textual changes, e.g., the time is $O(\sqrt{f(Q) \log f(Q)m(Q)})$.

Edmonds’ search is implemented with known data structures. (These include non-trivial data structures for set merging [13] and split-findmin [26].) We use a data structure for set merging to track the current partition of $V$ into atomic shells. It suffices to use the simple “relabel-the-smaller-half” strategy: Over the course of an entire scale, the atomic shell containing a given vertex $v$ changes by merging operations (even as DismantlePath transitions from one major path to the next). So $v$ can be examined every time its shell is merged into a larger shell. Thus the total overhead for set-merging, over an entire scale, is $O(n \log n)$.

Each current atomic shell $S$ records various items. The size $|S|$ is known from the set-merging data structure. $S$ has a list of its free vertices. $S$ has pointers to its boundary blossoms. Each inherited blossom $B$ records the initial dual value $z_0(B)$ and the current value $\tau(B)$. All current atomic shells of $P(Q)$ are maintained in a linked list that is
ordered as in $P(Q)$. The list has a pointer to the maximal undisolved blossom of $P(Q)$.

The graph is represented using a simple adjacency structure. When scanning the adjacencies of a vertex in shell $S$ it is easy to distinguish edges that remain in $S$ from edges that leave $S$.

The current dual functions $y, z$ are maintained using the quantities

$$y'(v) = y(v) + z\{B : v \in B \in Omega^-\}/2.$$ 

These quantities are initialized at the start of the scale using $y_0$ and $z_0$. They are maintained in Edmonds searches, i.e., a dual adjustment changing $y$ by $\pm 1$ makes the same change to $y'$. $y'(v)$ does not change in any unit translation of an inherited blossom (unlike $y'$ of Gabow and Tarjan [15]).

We compute $\hat{yz}$ values (as needed in Edmonds’ algorithm) by

$$\hat{yz}(uv) = y'[u, v] + z\{B : u, v \in B \in Omega\}.$$ 

Here $uv$ is an edge in a current shell $S$. To see this is correct consider a blossom $B \in Omega^-$. If $B$ contains both $u$ and $v$ $y'(u)$ and $y'(v)$ contribute a total of $z(B)/2 + z(B)/2 = z(B)$ to the right-hand side, as desired. If $B$ contains only one of $u, v$, say $u$, then blossom $B$ dissolves before SHELLSEARCH($S$), so $y'(u)$ correctly includes the contribution of $z(B)/2$ to $y(u)$.

Further details depend on the phase.

B.1 Phase 1

The list $A$ is implemented using buckets $B[i], 1 \leq i \leq |Q|$, where $B[i]$ contains the atomic shells of size $i$ that contain $\geq 1$ free vertex.

The rematch step of Phase 1 computes a maximal set of disjoint augmenting paths using the algorithm of Gabow and Tarjan [12, 15] for ordinary matching, and Gabow [9] for $f$-factors. Both run in linear time $O(m)$.

B.2 Phase 2

We use a priority queue $PQ$ of $c|Q| \log |Q|$ buckets. An entry $PQ[t]$ is a list of the events of Phase 2 that are predicted to occur in the $t$th dual adjustment of the current shell $S$ being searched.

To keep the space linear $O(n)$, $PQ$ is divided into $c \log n$ “pages” of $n$ buckets. Only the current page is implemented as an array of $n$ buckets. The other pages are simply lists of events scheduled for time units in that page.

An event in $PQ[t]$ is either a step of Edmonds’ search in the current shell $S$, or a shell boundary $S^-$ or $S^+$ that will dissolve in time unit $t$. When such a boundary dissolves, merging $S$ with an adjacent shell $S'$, we scan the adjacency list of each vertex $v \in S'$. This entails adding new Edmonds events to $PQ$, and possibly executing Edmonds
events that occur in the current time unit. In particular new search trees are added for free vertices in $S'$.

The adjacency lists scans use total time $O(m)$ per augment. as desired for our time bound. (In addition a scan may occur in a search whose shell eventually gets deactivated, again within our time bound.)

C The $f$-Factor Algorithm

The LP for $f$-factors is similar to matching (Appendix A), incorporating the degree constraint function $f$ and $I$-sets of blossoms. (It is derived in [10]. Alternatively it follows from Schrijver [24, Theorem 33.2] by increasing edge weights by a large amount). We allow the graph to have parallel edges. Each copy of an edge has its own $x$ variable indicating membership in the $f$-factor. Recall $\gamma(v)$ is the set of all loops at $v$.

Maximize $\sum_{e \in E} w(e)x(e)$ subject to
\[ x(\delta(v)) + 2x(\gamma(v)) = f(v) \forall v \in V \]
\[ x(\gamma(B) \cup I) \leq \left\lfloor \frac{f(B) + |I|}{2} \right\rfloor \forall B \subseteq V, I \subseteq \delta(B) \]
\[ x(e) \leq 1 \forall e \in E \]
\[ x(e) \geq 0 \forall e \in E \]

We call $e$ dominated, tight, or underrated depending on whether $\hat{y}z(e)$ is $\geq w(e)$, $= w(e)$, or $\leq w(e)$, respectively; strictly dominated and strictly underrated refer to the possibilities $> w(e)$ and $< w(e)$ respectively.

The dual LP uses dual functions $y : V \rightarrow \mathbb{R}$, $z : 2^V \times 2^E \rightarrow \mathbb{R}_+$. Define $\hat{y}z : E \rightarrow \mathbb{R}$ by

\[ \hat{y}z(e) = y(e) + z((B, I) : e \in \gamma(B) \cup I). \tag{C.1} \]

minimize $\sum_{v \in V} f(v)y(v) + \sum_{B \subseteq V, I \subseteq \delta(B)} \left\lfloor \frac{f(B) + |I|}{2} \right\rfloor z(B, I) + u(E)$ subject to
\[ \hat{y}z(e) + u(e) \geq w(e) \forall e \in E \]
\[ u(e) \geq 0 \forall e \in E \]
\[ z(B, I) \geq 0 \forall B \subseteq V, I \subseteq \delta(B) \]

In the $f$-factor algorithm every nonzero $z$ value has the form $z(B, I(B))$ for $B$ a mature blossom. So we write $z(B)$ as a shorthand for $z(B, I(B))$.

Gabow [10] treats $f$-factors by first generalizing algorithmic concepts from ordinary matching, and then modifying the matching procedure of Fig. 22 to apply to $f$-factors. We briefly review this presentation, referring the reader to [10] for a complete development.

We begin by defining $f$-factor blossoms. We start with some notation before giving the complete definition. We are given a graph $G$ with degree constraints $f(v)$, and
a subgraph $M$ (called a “matching”) with each $d(v, M) \leq f(v)$.\footnote{Throughout this appendix $d$ denotes the degree function, not to be confused with the $d$ counting dual adjustments in the analysis of the scaling algorithm. Note $d(v) = |\delta(v)| + 2|\gamma(v)|$.} A blossom $B$ is a subgraph of $G$. $B$ has a base vertex $\beta(B) \in V(B)$, denoted $\beta$ if the blossom is clear. The subgraph $B$ contains various matched edges but there may also be matched edges in $(\gamma(B) \cup \delta(B)) - E(B)$. Every vertex $v \in V(B) - \beta$ is perfectly matched, i.e., $d(v, M) = f(v)$. The base vertex has $d(\beta, M) \geq f(\beta) - 1$.

If $d(\beta, M) = f(\beta) - 1$ the blossom is free. A free blossom has $\eta(B) = \emptyset$ in the formal definition. But for the purpose of defining blossoms it is convenient to add an artificial edge $\beta \beta'$ incident to $B$, call it matched, and set $\eta(B) = \beta \beta'$. So a free blossom is light.

The following inductive definition of a blossom is taken from Gabow [10], with minor changes that incorporate the above notation. Figure 23 partially illustrates the definition.

**Definition C.1** Let $\bar{G}$ be a graph derived from $G$ by contracting a family $A$ of zero or more vertex-disjoint blossoms. A vertex of $\bar{G}$ but not $A$ is an atom. Let $C$ be a closed path in $\bar{G}$ that starts and ends at a vertex $\alpha$ of $\bar{G}$. The preimage of $C$ in $G$ is a blossom $B$ with base vertex $\beta$ if $C$ has the following properties:

- If $\alpha$ is an atom then $\alpha = \beta(B)$. $C$ starts and ends with two edges $e_B$ incident to $\beta$ that are both matched in a heavy blossom, both unmatched in a light blossom.
- If $\alpha \in A$ then $\beta(B) = \beta(\alpha)$. The $e_B$ edges are also those of $\alpha$.
- If $v \in A \cap C - \beta$ then the two edges of $\delta_C(v)$ alternate.
- If $v \in A \cap C - \alpha$ then $\eta(v) \in \delta_C(\beta(v))$.

The closed path $C$ is denoted $C(B)$ if the blossom needs to be specified. The subgraph of $B$, denoted $(V(B), E(B))$, is a subgraph of $G$ consisting of the subgraphs of all subblossoms $A$ of $B$ plus the atoms of $B$ and the edges of $C(B)$. We sometimes treat $B$ as just the vertex set $V(B)$.

A key property of blossoms is that every vertex $v \in V(B)$ has two alternating trails from $v$ to $\beta$: $P_0(v, \beta)$ which has even length and $P_1(v, \beta)$ which has odd length. Each $P_1$ is contained in $B$’s subgraph, $E(P_1) \subseteq E(B)$. The $P_1$’s are recursively defined: For every blossom $A \in A$ containing a vertex of $P_1$, $E(P_1) \cap E(A)$ is some trail $P_i'(v', \beta(A))$, and $P_i$ contains $\eta(A)$ unless $\beta(A) = \beta$. ($P_i'$ may occur in reverse order in $P_i$.) The definition easily gives this generalization:

For every blossom $A \neq \beta$, $P_i(v, \beta)$ contains $\eta(A)$ if it contains any vertex of $A$. One of the trails $P_0$, $P_1$ starts with a matched edge, the other starts with an unmatched edge. Both trails end with an $e_B$ edge with one exception, $P_0(\beta, \beta)$ has length 0. $P_1(\beta, \beta)$ is nonsimple, it contains both $e_B$ edges. So any $P_i(v, \beta)$ extended with the edge $\eta(B)$ is still an alternating trail. (In 1-matching, blossoms are always light. The $P_0$ are required to be paths, and $P_1$ paths need not exist.)

A blossom $B$ has an associated set

$$I(B) = \delta_M(B) \oplus \eta(B).$$
A free $B$ has $I(B) = \delta_M(B) \oplus \emptyset = \delta_M(B)$. These sets play the role of $I$ in the above LP, i.e., instead of writing $z(B, I)$ we denote $z$ duals as $z(B)$ and define

$$\hat{y}z(e) = y(e) + z\{B : e \in \gamma(B) \cup I(B)\}.$$ 

Although Fig. 23 draws $I(B)$ edges as directed, an edge $xy$ may belong to both $I(B_x)$ and $I(B_y)$.

The $f$-factor algorithm constructs a search forest analogous to matching. The roots of the forest are the free blossoms and free atoms (vertices with $d(v, M) < f(v)$). For a nonroot node $v$ of the forest, $\tau(v)$ denotes the first edge of the path from $v$ to its root. As in matching we say an edge $xy \notin \tilde{S}$ alternates with node $x \in \tilde{S}$ if $xy$ can be added to the forest, i.e., there is an alternating path of edges of $G$ that starts with $xy$. If $x$ is in a contracted blossom the path includes $P_i(x, \beta)$, where $i$ is chosen so $P_i$ starts with an edge alternating with $xy$. The classification of nodes as inner and outer is illustrated in Fig. 23. (In 1-matching possibilities (c) and (f) do not exist and there are no $I(B)$ edges.)

We turn to the $f$-factor algorithm. As in matching the search forest is rooted at every free atom and free blossom. Eligibility is defined exactly as in matching. Pseudocode for a search is the same as matching Fig. 22 except for new versions of the blossom step, the Expand routine, and minor changes to AdjustDuals, all given in Fig. 24.

Figure 25 illustrates a simple execution of the algorithm. It shows that the baseless blossoms treated in the compression step of our algorithm (Fig. 15) can actually be formed.

We conclude with a property closely related to the tightening step of our algorithm (Fig. 15). Consider a blossom $B_u$ with base edge $\eta(B_u) = uv$. $\eta(B_u)$ is eligible if $B_u$ is in a maximal blossom with a different base vertex. This need not hold if $u$ is the base of a maximal blossom, say $B$. To illustrate assume $B$ is not in the search tree. If $uv$ is unmatched and $v$ is an inner vertex, dual adjustments increase $\hat{y}z(uv)$. If $uv$ is matched and $v$ is an outer atom or $v$ is in an inner blossom, dual adjustments decrease $\hat{y}z(uv)$.

The following lemma identifies another configuration where $uv$ is always eligible. We do not use the lemma in the main body of the paper—it is included for completeness, as well as illustrating the execution of the algorithm. Also for completeness we prove the lemma for duals that are optimum as well as near optimum. An $\eta$ pair is an edge $uv$ with $uv = \eta(B_u) = \eta(B_v)$.

**Lemma C.1** At any point in the $f$-factor algorithm, an $\eta$ pair $e$ has $\hat{y}z(e) = w(e) + \Delta$ where

$$\Delta \begin{cases} 0 & \text{for optimum duals} \\ \in [-2, 0] & \text{for near-optimum duals} \end{cases}$$

**Proof** The condition $uv = \eta(B_u) = \eta(B_v)$ becomes satisfied in a blossom step that creates one of the blossoms $B_u, B_v$, or in an expand step that changes $B$ from inner to outer, or in an augment step that makes $uv$ an $\eta$ pair. In all cases $uv$ is eligible, so $\Delta$ has the claimed value.
Fig. 23 The six types of nodes \( v \) in a search forest. \( a-c \) are outer nodes and \( d-f \) are inner nodes. Edges drawn heavy are matched, light edges are unmatched. As labelled in (a), \( \tau(v) \) edges are always drawn above \( v \) and edges alternating with \( v \) are always drawn below \( v \). (They need not be in the search forest.) \( \tau(v) \) does not exist for a search forest root, which has the form of either (a) or (b). For blossoms, the two \( e_B \) edges are always shown. Edges in \( I(B) \) sets are drawn directed to blossom \( B \). In \( e \) and \( f \) \( \tau(v) \) can be matched or unmatched.

Suppose a dual adjustment step changes \( \Delta \). Clearly \( u \) and \( v \) must be in different maximal blossoms, and at least one of those blossoms is in the search structure. Let \( B \) \((B')\) be the maximal blossom containing \( u \) \((v)\) respectively, and wlog assume \( B \) enters the search structure before \( B' \). \( B \) enters in a grow step for an edge \( xy \) with \( y \in B \neq x, \ xy \neq uv \). So \( B \) is an inner vertex (Fig. 23e or f).

We will show the dual adjustment maintains \( \Delta \) as claimed in the lemma. First suppose the algorithm is using optimum duals. \( uv \) is eligible, so a grow step for \( uv \) adds \( B' \) to the search structure as an outer vertex (Fig. 23b or c). The subsequent dual adjustment changes duals by

\[
\begin{align*}
y(u) &\leftarrow y(u) + \delta, \\
z(B) &\leftarrow z(B) - \delta \\
y(v) &\leftarrow y(v) - \delta, \\
z(B') &\leftarrow z(B') + \delta.
\end{align*}
\]
The changes in $y$ cancel each other. The changes in $z$ do not change $\hat{\gamma}_z(uv)$ when $uv$ is matched (since $uv \notin I(B) \cup I(B')$ and $z$ is irrelevant) or unmatched (since $uv \notin I(B) \cap I(B')$ and the changes in $z$ cancel).

Suppose the algorithm is using near optimum duals. If $uv$ is eligible the analysis for optimal duals applies. Suppose $uv$ is ineligible. The dual adjustment changes $y(u)$ and $z(B)$ as above. If $uv$ is matched then $uv \notin I(B)$ so $\hat{\gamma}_z(uv)$ increases. $\Delta$ increases from an initial value $< 0$, so the lemma’s condition is preserved. If $uv$ is unmatched
then $uv \in I(B)$ so $\hat{\gamma}(uv)$ decreases. $\Delta$ decreases from an initial value $> -2$, so the lemma’s condition is preserved. □

References

1. Brand, J.V.D., Lee, Y.T., Liu, Y.P., Saranurak, T., Sidford, A., Song, Z., Wang, D.: Minimum cost flows, MDPs, and $\ell_1$-regression in nearly linear time for dense instances. In: Proc. 53 Annual ACM Symp. on Theor. Comp., pp. 859–869 (2021)
2. Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., Schrijver, A.: Combinatorial Optimization. Wiley and Sons, New York (1998)
3. Duan, R., He, H., Zhang, T.: A scaling algorithm for weighted $f$-factors in general graphs. In: Proc. of the 47th International Colloquium on Automata, Languages, and Programming (ICALP 2020), vol. 168 of LIPIcs, pp. 41:1–41:17 (2020)
4. Duan, R., Pettie, S., Su, H.-H.: Scaling algorithms for weighted matching in general graphs. ACM Trans. Algorithms 14(1), 1–35 (2018)
5. Edmonds, J.: Maximum matching and a polyhedron with 0,1-vertices. J. Res. Nat. Bur. Stand. 69B, 125–130 (1965)
6. Even, S., Tarjan, R.E.: Network flow and testing graph connectivity. SIAM J. Comput. 4, 507–518 (1975)
7. Fredman, M.L., Tarjan, R.E.: Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM 34(3), 596–615 (1987)
8. Gabow, H.N.: A scaling algorithm for weighted matching on general graphs. In: Proc. 26th Annual Symp. on Found. of Comp. Sci., pp. 90–100 (1985)
9. Gabow, H.N.: Blocking trails for $f$-factors of multigraphs. arXiv:2112.04096
10. Gabow, H.N.: Data structures for weighted matching and extensions to $b$-matching and $f$-factors. ACM Trans. Algorithms 14(3), 1–80 (2018)
11. Gabow, H.N.: An efficient implementation of Edmonds’ algorithm for maximum matching on graphs. J. ACM 23(2), 221–234 (1976)
12. Gabow, H.N.: The weighted matching approach to maximum cardinality matching. Fund. Inform. 154(1–4), 109–130 (2017)
13. Gabow, H.N., Tarjan, R.E.: A linear-time algorithm for a special case of disjoint set union. J. Comput. Syst. Sci. 30(2), 209–221 (1985)
14. Gabow, H.N., Tarjan, R.E.: Faster scaling algorithms for network problems. SIAM J. Comput. 18(5), 1013–1036 (1989)
15. Gabow, H.N., Tarjan, R.E.: Faster scaling algorithms for general graph matching problems. J. ACM 38(4), 815–853 (1991)
16. Galil, Z., Micali, S., Gabow, H.N.: An $O(EV \log V)$ algorithm for finding a maximal weighted matching in general graphs. SIAM J. Comput. 15(1), 120–130 (1986)
17. Hopcroft, J., Karp, R.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM J. Comput. 2(4), 225–231 (1973)
18. Huang, D., Pettie, S.: Approximate generalized matching: $f$-matchings and $f$-edge covers. Algorithmica 84(7), 1952–1992 (2022)
19. Karzanov, A.V.: On finding maximum flows in network with special structure and some applications. Mat. Vopr. Upr. Proizvod. 5, 81–94 (1973). (In Russian.)
20. Kuhn, H.W.: The Hungarian method for the assignment problem. Naval Res. Log. Q. 2, 83–97 (1955)
21. Lawler, E.L.: Combinatorial Optimization: Networks and Matroids. Holt, Rinehart and Winston, New York (1976)
22. Lovász, L., Plummer, M.D.: Matching Theory, North-Holland Mathematical Studies, vol. 121. North-Holland, New York (1986)
23. Micali, S., Vazirani, V.V.: An $O(\sqrt{|V| \cdot |E|})$ algorithm for finding maximum matching in general graphs. In: Proc. 21st Annual Symp. on Found. of Comp. Sci., 21st, pp. 17–27 (1980)
24. Schrijver, A.: Combinatorial Optimization: Polyhedra and Efficiency. Springer, New York (2003)
25. Tarjan, R.E.: Applications of path compression on balanced trees. J. ACM 26(4), 690–715 (1979)
26. Thorup, M.: Undirected single-source shortest paths with positive integer weights in linear time. J. ACM 46(3), 362–394 (1999)
Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.