Algorithmica (2012) 64:362–383
DOI 10.1007/s00453-012-9642-6

Improved Algorithms for Even Factors and Square-Free Simple b-Matchings

Maxim A. Babenko

Received: 16 February 2011 / Accepted: 23 March 2012 / Published online: 19 April 2012

Abstract Given a digraph $G = (VG, AG)$, an *even factor* $M \subseteq AG$ is a set formed by node-disjoint paths and even cycles. Even factors in digraphs were introduced by Geelen and Cunningham and generalize path matchings in undirected graphs.

Finding an even factor of maximum cardinality in a general digraph is known to be NP-hard but for the class of *odd-cycle symmetric* digraphs the problem is polynomially solvable. So far the only combinatorial algorithm known for this task is due to Pap; its running time is $O(n^4)$ (hereinafter n denotes the number of nodes in G and m denotes the number of arcs or edges).

In this paper we introduce a novel *sparse recovery* technique and devise an $O(n^3 \log n)$-time algorithm for finding a maximum cardinality even factor in an odd-cycle symmetric digraph. Our technique also applies to other similar problems, e.g. finding a maximum cardinality square-free simple b-matching.

Keywords Even factors · Odd-cycle symmetric graphs · Square-free simple b-matchings · Alternating paths

1 Introduction

The notion of *independent path matchings* was introduced by Cunningham and Geelen [5] in the context of the separation problem for the matchable set polytope (which was earlier studied by Balas and Pulleyblank [2]). Finding an independent path matching of maximum size was recognized as an intriguing example of a

Supported by RFBR grant 09-01-00709-a.

M.A. Babenko (✉)
Dept. of Mechanics and Mathematics, Moscow State University, GSP-1, Leninskie Gory, Moscow 119991, Russia
e-mail: maxim.babenko@gmail.com

M.A. Babenko
Yandex LLC, 16, Leo Tolstoy St., Moscow 119021, Russia

graph-theoretic optimization problem that is difficult to tackle by purely combinatorial means. Two algorithms were given by Cunningham and Geelen: one relies on the ellipsoid method [5], and the other is based on deterministic evaluations of the Tutte matrix [6]. Later a rather complicated combinatorial algorithm was proposed by Spille and Weismantel [17].

The notion of an *even factor* is a further generalization of path matchings [4, 7]. An *even factor* is a set of arcs that decomposes into a node-disjoint collection of simple paths and simple cycles of even lengths. Since cycles of length 2 are allowed, it is not difficult to see that finding a maximum matching in an undirected graph G reduces to computing a maximum even factor in the digraph obtained from G by replacing each edge with a pair of oppositely directed arcs. On the other hand, no reduction from even factors to non-bipartite matchings is known.

Finding a maximum cardinality even factor in a general digraph is known to be NP-hard [4]. For the class of *weakly symmetric digraphs*, a min-max relation and an Edmonds–Gallai-type structure were established by Cunningham and Geelen [7] and by Pap and Szegő [16]. Later it was observed by Pap [14] that same arguments apply to a slightly broader class of *odd-cycle symmetric digraphs*. Kobayashi and Takazawa [12] pointed out a relation between even factors and jump systems and showed that "odd-cycle symmetric digraphs" form a natural class of problem instances.

The existence of a combinatorial solution to the maximum even factor problem in an odd-cycle symmetric digraph remained open for quite a while. Recently Pap gave a direct $O(n^4)$-time algorithm [14, 15]. His method can be slightly improved to run in $O(n^2(m+n\log n))$ time, as explained in Sect. 3. (Hereinafter n denotes for the number of nodes in G and m denotes the number of arcs or edges in G.) To compare: the classical algorithm of Micali and Vazirani that computes a maximum non-bipartite matching (which, as indicated above, is a special case of the maximum even factor problem) runs in $O(m\sqrt{n})$ time [13]. It is tempting to design a faster algorithm for the maximum even factor problem by applying ideas developed for matchings (e.g. blocking augmentation [9]). There are, however, certain complications making even the bound of $O(mn)$ nontrivial.

To explain the nature of these difficulties let us briefly review Pap's approach (a more detailed exposition will be given in Sect. 3). It resembles Edmonds' non-bipartite matching algorithm, and executes a series of iterations. Each iteration tries to increase the size of the current even factor M by one. At each iteration, a search for an augmenting path P is performed. If no such path is found, then M is maximum. Otherwise, the algorithm tries to apply P to M. If no odd cycle appears after the augmentation, the iteration completes. Otherwise a certain contracting reduction is applied to G and M.

Hence each iteration consists of *phases* and the number of nodes in the current digraph decreases with each phase. Totally there are $O(n)$ iterations and $O(n)$ phases during each iteration, which is quite similar to the usual blossom-shrinking method. The difference is that during a phase the reduction may change the alternating reachability structure completely, so the next phase is forced to start looking for P from scratch. (Compare this with Edmonds' algorithm where a blossom contraction changes the alternating forest in a predicable and consistent way thus allowing this forest to be reused over the phases.)

This paper aims to overcome the above issue by introducing a novel *sparse recovery* technique. Section 4 presents an $O(n^3 \log n)$-time algorithm for the maximum even factor problem. It is based on Pap's method but grows an alternating forest in a more careful fashion. When a contraction is made in the current digraph the forest gets destroyed. However we are able to restore it by running a *sparse recovery* procedure that carries out a reachability search in a specially crafted digraph with $O(n)$ arcs in $O(n \log n)$ time (where the $\log n$-factor comes from manipulations with balanced trees used for odd cycle testing).

Finally we extend our approach to the maximum square-free simple b-matching problem. The latter reads as follows. Let G be an undirected bipartite graph with node capacities $b\colon VG \to \{1, 2\}$. One looks for a maximum cardinality simple b-matching (i.e. a subset of edges $M \subseteq EG$ such that each $v \in VG$ is incident to at most $b(v)$ edges of M) that contains no circuits of length 4. Hartvigsen [8] established a nice characterization for this problem and also presented a rather complicated combinatorial algorithm. Király [11] gave another, non-algorithmic proof of the min-max formula. Pap [15] devised a conceptually simpler combinatorial algorithm, which is based on the above ideas of iterative augmentations and contractions. The latter algorithm runs in $O(mn^2)$, and like one for even factors it suffers from the need to recompute alternating forests. In Sect. 5.4 we demonstrate how sparse recovery helps to reduce the complexity to $O(n^3)$ (matching that claimed in [8]).

The extended abstract [1] of this paper previously appeared at ISAAC 2010.

2 Preliminaries

We employ some standard graph-theoretic notation throughout the paper. For an undirected graph G, we denote its sets of nodes and edges by VG and EG, respectively. For a directed graph, we speak of arcs rather than edges and denote the arc set of G by AG. A similar notation is used for paths, trees, and etc. We allow parallel edges and arcs but not loops. As long as this leads to no confusion, an arc from u to v is denoted by (u, v) and an edge connecting u and b by $\{u, v\}$.

A path or a cycle is called *even* (respectively *odd*) if is consists of an even (respectively *odd*) number of arcs or edges. The following definition is crucial:

Definition 1 For a digraph G, a *path-cycle matching* is a subset of arcs M that is a union of node-disjoint simple paths and cycles in G. When M contains no odd cycle it is called an *even factor*. The *size* of M is its cardinality and the *maximum even factor problem* consists in constructing an even factor of maximum size.

An arc (u, v) in a digraph G is called *symmetric* if (v, u) is also present in G. Following the terminology of [14], we call G *odd-cycle symmetric* (respectively *weakly symmetric*) if for each odd (respectively any) cycle C, all arcs of C are symmetric. As observed by Kobayashi and Takazawa [12], this symmetry is essential for the tractability of the problem. It also provides a link between odd cycles (in digraphs) and factor critical subgraphs (in undirected graphs) and makes uncontractions possible (see Lemma 1 below).

For a digraph G and $U \subseteq VG$, the set of arcs entering (respectively leaving) U is denoted by $\delta^{\text{in}}_G(U)$ (respectively $\delta^{\text{out}}_G(U)$). If G is undirected then $\delta_G(U)$ denotes the set of edges with exactly one endpoint in U. For a graph G (directed or undirected) we write $\gamma_G(U)$ to denote the set of arcs (or edges) with both endpoints in U and $G[U]$ to denote the subgraph of G induced by U, i.e. $G[U] = (U, \gamma_G(U))$. When G is clear from the context it is omitted from notation.

To *contract* a set $U \subseteq VG$ in a digraph G means to replace nodes in U by a single *composite node*. Arcs in $\gamma(VG - U)$ are not affected, arcs in $\gamma(U)$ are dropped, and arcs in $\delta^{\text{in}}(U)$ (respectively $\delta^{\text{out}}(U)$) are redirected so as to enter (respectively leave) the composite node. The resulting graph is denoted by G/U. We identify arcs in G/U with their pre-images in G. Note that G/U may contain multiple parallel arcs but not loops. If G' is obtained from G by an arbitrary series of contractions then $G' = G/U_1/\ldots/U_k$ holds for a certain family of disjoint subsets $U_1, \ldots, U_k \subseteq VG$ (called *maximum contracted sets*).

Maximizing the size of an even factor M in a digraph G is equivalent to minimizing the *deficiency* $\text{def}(G, M) := |VG| - |M|$. The minimum deficiency of an even factor in G is called the *deficiency* of G itself and is denoted by $\text{def}(G)$.

To give the reader some initial insight to the class of related problems let us explain the reduction from non-bipartite matchings to even factors. For this aim, let G be an undirected graph where a maximum matching is requested. We construct a digraph $\overrightarrow{H}$ by taking the node set of H and replacing each edge $\{u, v\}$ by a pair of oppositely directed arcs (u, v) and (v, u). Note that $\overrightarrow{G}$ is odd-cycle symmetric. Every matching M in G generates an even factor $\overrightarrow{M}$ in $\overrightarrow{G}$ that consists of cycles of length 2 and obeys $|\overrightarrow{M}| = 2|M|$. Vice versa, let $\overrightarrow{M}$ be a maximum even factor in $\overrightarrow{G}$. It consists of node-disjoint even cycles and even paths (odd paths cannot occur in $\overrightarrow{M}$ due to its maximality). A trivial reduction transforms $\overrightarrow{M}$ to an even factor having the same size and consisting only of cycles of length 2. This way, $\overrightarrow{M}$ gives rise to a maximum matching M in G.

3 Naive Algorithm

Consider an odd-cycle symmetric digraph G. The algorithm for finding a maximum even factor in G follows the standard augmentation scheme. Namely, it initially starts with $M = \emptyset$ and executes a series of *iterations* each aiming to increase $|M|$ by one. Each iteration invokes NAIVE-EVEN-FACTOR routine that, given an odd-cycle symmetric digraph G and an even factor M in G, either returns a larger even factor M^+ or NULL indicating that M is already maximum.

3.1 Augmentations

Let us temporarily allow odd cycles and focus on path-cycle matchings in G. Construct two disjoint copies of VG: $V^1 := \{v^1 \mid v \in VG\}$ and $V^2 := \{v^2 \mid v \in VG\}$. For each arc $a = (u, v) \in AG$ add an edge $\{u^1, v^2\}$ (*corresponding* to a). Denote the resulting undirected bipartite graph by $\widetilde{G}$.

Clearly a path-cycle matching M in G is characterized as follows: for each node $v \in VG$, M has at most one arc entering v and also at most one arc leaving v. Translating this to $\widetilde{G}$ one can readily see that M generates a matching $\widetilde{M}$ in $\widetilde{G}$. Moreover, this correspondence between matchings in $\widetilde{G}$ and path-cycle matchings in G is one-to-one. A node u^1 (respectively u^2) in $\widetilde{G}$ not covered by $\widetilde{M}$ is called a *source* (a *sink*, respectively).

Given a digraph G and a path-cycle matching M in G, we turn $\widetilde{G}$ into a digraph $\overrightarrow{G}(M)$ by directing edges $\{u^1, v^2\}$ that correspond to arcs $(u, v) \in M$ from v^2 to u^1 and other edges from u^1 to v^2.

Definition 2 A simple even path in $\overrightarrow{G}(M)$ that starts in a source node is called *M-alternating*. A simple path in $\overrightarrow{G}(M)$ that starts in a source node and ends in a sink node is called *M-augmenting*.

Clearly each M-augmenting path is odd. For a path P in $\overrightarrow{G}(M)$, let $A(P)$ denote the set of arcs in G that correspond to arcs of P in $\overrightarrow{G}(M)$. Hereinafter $A \triangle B$ denotes the symmetric difference of sets A and B. The next statements are well-known.

Claim 1 *M is a path-cycle matching of maximum size iff $\overrightarrow{G}(M)$ contains no M-augmenting path.*

Claim 2 *If P is an M-augmenting (respectively M-alternating) path in $\overrightarrow{G}(M)$, then $M' := M \triangle A(P)$ is a path-cycle matching with $|M'| = |M| + 1$ (respectively $|M'| = |M|$).*

The augmentation procedure (see Algorithm 1) constructs $\overrightarrow{G}(M)$ and searches for an augmenting path P. In case no such path exists, the current even factor M is maximum by Claim 1 (and also forms a maximum path-cycle matching), hence the algorithm terminates. Next assume $\overrightarrow{G}(M)$ contains an augmenting path P. Claim 2 indicates how a larger path-cycle matching M' can be formed from M, however M' may contain an odd cycle. The next definition focuses on this issue.

Definition 3 Given an even factor M, let P be an M-augmenting or an M-alternating path. Then P is called *feasible* if $M' := M \triangle A(P)$ is again an even factor.

If P is feasible, then NAIVE-EVEN-FACTOR exits with the updated even factor $M \triangle A(P)$. Consider the contrary, i.e. P is not feasible. Since P is M-augmenting, it must be odd, say it consists of $2k + 1$ arcs. Construct a sequence of M-alternating paths $P_0, \ldots, P_k$ where P_i is formed by taking the first $2i$ arcs of P $(0 \le i \le k)$. Also set $P_{k+1} := P$ and $M_i := M \triangle A(P_i)$ $(0 \le i \le k + 1)$.

Then there exists an index i $(0 \le i \le k)$ such that P_i is feasible while P_{i+1} is not feasible. In other words, M_i is an even factor obeying $\mathrm{def}(G, M_i) = \mathrm{def}(G, M)$ and M_{i+1} contains an odd cycle. Since M_i and M_{i+1} differ by at most two arcs, it can be easily shown that an odd cycle in M_{i+1}, call it C, is unique (see [14]). Moreover, C *fits* M_i, that is, $|M_i \cap AC| = |VC| - 1$ and $\delta^{\mathrm{out}}(VC) \cap M_i = \emptyset$. See Fig. 1(b) for an example.

Algorithm 1 NAIVE-EVEN-FACTOR(G, M)

```
1: Search for an augmenting path P in G→(M)
2: if P does not exist then
3:    return NULL
4: else if P exists and is feasible then
5:    return M △ A(P)
6: else [P exists but is not feasible]
7:    Define M_i ⇐ M △ A(P_i) for i = 0, ..., k + 1
8:    Find an index i such that M_i is an even factor while M_{i+1} is not
9:    Find the unique odd cycle C in M_{i+1}
10:   G' ⇐ G/C, M' ⇐ M_i/C
11:   M̄' ⇐ NAIVE-EVEN-FACTOR(G', M')
12:   if M̄' = NULL then [M' is maximum in G']
13:      return NULL
14:   else [M' is augmented in G' to a larger even factor M̄']
15:      Undo the contractions and transform M̄' to an even factor M+ in G
16:      return M+
17:   end if
18: end if
```

It turns out that when there exists an odd cycle fitting an even factor then a certain optimality-preserving contraction is possible. As long as no confusion is possible, for a digraph G and a cycle C in G, we abbreviate G/VC to G/C. Also for $X \subseteq AG$, we write X/C to denote $X \setminus \gamma_G(VC)$.

Claim 3 (Pap [14]) *Let N be an even factor in G and C be an odd cycle that fits N. Define $G' := G/C$ and $N' := N/C$. Then G' is an odd-cycle symmetric digraph and N' is an even factor in G'. Moreover, if N' is maximum in G', then N is maximum in G.*

Note that M is maximum in G if and only if M_i is maximum in G. NAIVE-EVEN-FACTOR contracts C in G. Let $G' := G/C$ and $M' := M_i/C$ denote the resulting digraph and the even factor. To check if M' is maximum in G' a recursive call NAIVE-EVEN-FACTOR(G', M') is made (thus starting a new phase). If NULL is returned, then M' is a maximum even factor in G', which implies by Claim 3 that the initial even factor M was maximum in G. In this case NAIVE-EVEN-FACTOR terminates returning NULL.

Suppose that the recursive call was able to turn M' into a larger even factor $\overline{M}'$ in G'. One can transform $\overline{M}'$ into an even factor of the same deficiency in G as follows:

Lemma 1 (Pap [14]) *Let C be an odd cycle in an odd-cycle symmetric digraph G. Denote $G' := G/C$ and let N' be some even factor in G'. Then there exists an even factor N in G obeying* $\mathrm{def}(G, N) = \mathrm{def}(G', N')$.

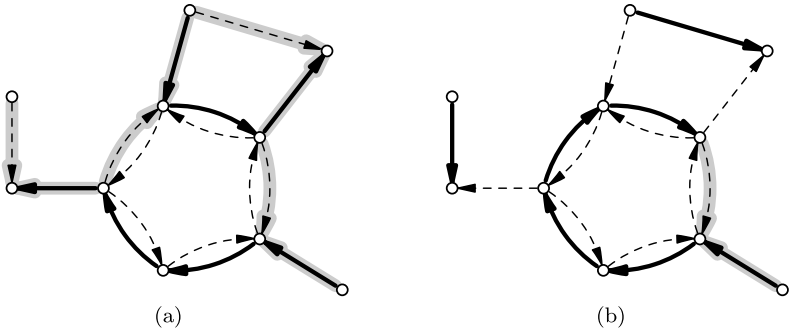

Fig. 1 Preparing to contract an odd cycle. (**a**) Arcs of M are *bold*, others are *dashed*, *grayed arcs* correspond to path P_{i+1}. (**b**) Path P_i is applied, arcs of M_i are *bold*, others are *dashed*, *grayed arcs* indicate the remaining part of P_{i+1}

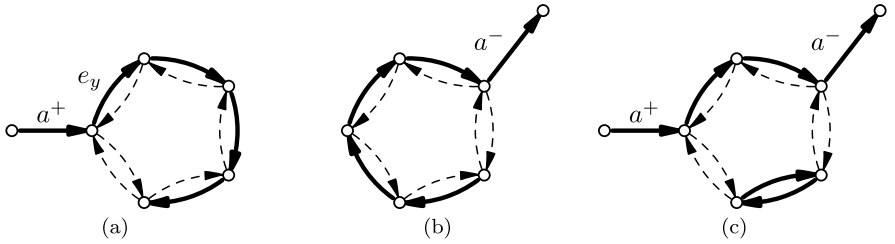

Fig. 2 Possible configurations in appearing in the proof Lemma 1. Digraph G and an even factor N are depicted. Arcs of N are *bold*, others are *dashed*. (**a**) Node z has an incoming arc $a^+ \in N'$ in G' but no such outcoming arcs. (**b**) Node z has an outcoming arc $a^- \in N'$ in G' but no such incoming arcs. (**c**) Node z has both an incoming arc $a^+ \in N'$ and an outcoming arc $a^- \in N'$ in H'. The subcase when no arc of N' is incident to z in G' is trivial and is thus omitted

Proof (Sketch) Let z be the composite node in G' corresponding to C in G. Consider the arcs of N' incident to z in G' and apply a simple case-splitting (see Fig. 2 for examples). □

3.2 Complexity

Clearly the above schema leads to a polynomial time algorithm. Let us estimate its complexity. There are $O(n)$ iterations each consisting of $O(n)$ phases (contractions). To bound the complexity of a single phase note that it takes $O(m)$ time to find an augmenting path P (or figure out that it does not exist). We may construct path-cycle matchings $M_0, \ldots, M_{k+1}$ and decompose each of them into node-disjoint paths and cycles in $O(n^2)$ time. Hence finding the index i and the cycle C takes $O(n^2)$ time. Contracting C in G takes $O(m)$ time. (After spending $O(m)$ time looking for P it is feasible to spend another $O(m)$ time to construct $G' = G/C$ explicitly.) An obvious bookkeeping allows to undo the contractions performed during the iteration and transform the final even factor in the contracted digraph into an even factor the initial digraph in $O(m)$ time. Totally the algorithm takes $O(n^4)$ time.

The above bound can be slightly improved as follows. Note that the algorithm needs an arbitrary index i such that M_i is an even factor and M_{i+1} is not, i.e. i is not required to be minimum. Hence we may carry out a binary search over the range $[0, k+1]$. At each step we keep a pair of indices (l, r) such that M_l is an even factor while M_r is not. Replacing the current segment $[l, r]$ by a twice smaller one takes $O(n)$ time and requires constructing and testing a single path-cycle matching M_t where $t := \lfloor (l+r)/2 \rfloor$. This way, the $O(n^2)$ term reduces to $O(n \log n)$ and the total running time becomes $O(n^2(m + n \log n))$. The ultimate goal of this paper is to get rid of the $O(m)$ term.

4 Faster Algorithm

4.1 Augmentations

The bottleneck of NAIVE-EVEN-FACTOR is augmenting path computations. To obtain an improvement we need better understanding of how these paths are calculated. Similarly to usual graph traversal algorithms we maintain a directed out-forest $\mathcal{F}$ rooted at source nodes. Nodes belonging to this forest are called $\mathcal{F}$*-reachable*. At each step a new arc (u, v) leaving an $\mathcal{F}$-reachable node u is scanned and is either added to $\mathcal{F}$ (thus making v $\mathcal{F}$-reachable) or skipped because v is already $\mathcal{F}$-reachable. This process continues until a sink node is reached or no unscanned arcs remain in the digraph.

Definition 4 Let G be a digraph and M be an even factor in G. An M*-alternating forest* $\mathcal{F}$ is a directed out-forest in $\overrightarrow{G}(M)$ obeying the following properties: (i) every root is a source node in $\overrightarrow{G}(M)$ and vice versa; (ii) every path from a root to a leaf is even.

The intuition behind the suggested improvement is to grow $\mathcal{F}$ carefully and to avoid exploring infeasible alternating paths.

Definition 5 An M-alternating forest $\mathcal{F}$ is called *feasible* if every M-alternating path in $\mathcal{F}$ is feasible. An M-alternating forest $\mathcal{F}$ is called *complete* if it contains no sink node and for each arc $(u, v) \in \overrightarrow{G}(M)$, if u is $\mathcal{F}$-reachable then v is also $\mathcal{F}$-reachable.

We replace NAIVE-EVEN-FACTOR by a more sophisticated recursive procedure called FAST-EVEN-FACTOR that obeys the following properties:

Lemma 2 FAST-EVEN-FACTOR *gets an odd-cycle symmetric digraph G, an even factor M in G, and an additional "sparsify" flag. It returns a digraph $\overline{G}$ obtained from G by a series of contractions and an even factor $\overline{M}$ in $\overline{G}$. Additionally it may return an $\overline{M}$-alternating forest $\overline{\mathcal{F}}$ in $\overline{G}$. Exactly one of the following cases applies*:

(a) $\mathrm{def}(\overline{G}, \overline{M}) = \mathrm{def}(G, M) - 1$ *and $\overline{\mathcal{F}}$ is undefined*; *or*
(b) $\mathrm{def}(\overline{G}, \overline{M}) = \mathrm{def}(G, M)$, *$\overline{M}$ is maximum in $\overline{G}$, M is maximum in G, and $\overline{\mathcal{F}}$ is a complete feasible $\overline{M}$-alternating forest in $\overline{G}$.*

Let us postpone the proof of Lemma 2 until the description of FAST-EVEN-FACTOR is given (see the end of this subsection and also Sect. 4.2). Note that the value of *sparsify* does not affect the public contract of FAST-EVEN-FACTOR. It is, however, used to control its recursive behavior and to achieve a good running time (see Sect. 4.3).

The algorithm performs iterations as follows. Given a current even factor M in G, it calls FAST-EVEN-FACTOR(G, M, TRUE) and examines the result. If $\mathrm{def}(\overline{G}, \overline{M}) = \mathrm{def}(G, M)$ then by Lemma 2(b), M is a maximum even factor in G, the algorithm stops. (Note that forest $\overline{\mathcal{F}}$, which is also returned by FAST-EVEN-FACTOR, is not used here. Like "*sparsify*", this forest is only needed due to the recursive nature of FAST-EVEN-FACTOR.)

Otherwise $\mathrm{def}(\overline{G}, \overline{M}) = \mathrm{def}(G, M) - 1$ by Lemma 2(a); this case will be referred to as a *breakthrough*. Applying Lemma 1, $\overline{M}$ is transformed to an even factor M^+ in G such that $\mathrm{def}(G, M^+) = \mathrm{def}(\overline{G}, \overline{M}) = \mathrm{def}(G, M) - 1$. This completes the current iteration.

It is clear that the algorithm constructs a maximum even factor correctly provided that Lemma 2 holds. Let us explain how FAST-EVEN-FACTOR works (see Algorithm 2 for a detailed pseudocode). During the course of its execution, FAST-EVEN-FACTOR maintains an M-alternating forest $\mathcal{F}$. It *scans* arcs of G in a certain order. For each node u in G we keep a list $L(u)$ of all unscanned arcs leaving u.

The following invariants are maintained during the execution of FAST-EVEN-FACTOR:

(a) $\mathcal{F}$ is a feasible M-alternating forest;
(b) If an arc $a = (u, v)$ is scanned, then either $a \in M$ or both u^1 and v^2 are $\mathcal{F}$-reachable. (1)

At Step 1 forest $\mathcal{F}$ is initialized to contain all source nodes.

Consider an $\mathcal{F}$-reachable node u^1 with $L(u) \neq \emptyset$. To enumerate unscanned arcs leaving u^1 in $\overrightarrow{G}(M)$, we fetch an unscanned arc $a = (u, v)$ from $L(u)$. If $a \in M$ or v^2 is $\mathcal{F}$-reachable, then a is marked as scanned and another arc is fetched. (In the former case a does not generate an arc leaving u^1 in $\overrightarrow{G}(M)$. In the latter case v^2 is already $\mathcal{F}$-reachable. In both cases making a scanned preserves (1).)

Otherwise consider the arc $a_1 := (u^1, v^2)$ in $\overrightarrow{G}(M)$ and let P_0 denote the feasible M-alternating path from a root of $\mathcal{F}$ to u^1. Note that each node x^2 in $\overrightarrow{G}(M)$ (for $x \in VG$) is either a sink or has a unique arc leaving x^2. A *single* step occurs when v^2 is a sink (Steps 8–9). The algorithm constructs an augmenting path $P_1 = P_0 \circ a_1$ leading to v^2. (Here $L_1 \circ L_2$ stands for the concatenation of L_1 and L_2.) If P_1 is feasible, the current even factor gets augmented according to Claim 2 and FAST-EVEN-FACTOR terminates. Otherwise forest growing stops and the algorithm proceeds to Step 19 to deal with a contraction.

A *double* step is executed when v^2 is not a sink (Steps 11–17). To keep all paths leading to leaves of $\mathcal{F}$ even, we add arcs in pairs. Namely, there is a unique arc leaving v^2 in $\overrightarrow{G}(M)$, say $a_2 = (v^2, w^1)$ (evidently $(w, v) \in M$). Moreover, w^1 is not a source node and (v^2, w^1) is the only arc entering w^1. Hence w^1 is not $\mathcal{F}$-reachable. If $P_1 := P_0 \circ a_1 \circ a_2$ is feasible, then a_1 and a_2 are added to $\mathcal{F}$ thus making v^2 and w^1 $\mathcal{F}$-reachable.

Algorithm 2 FAST-EVEN-FACTOR(G, M, *sparsify*)

```
1: Initialize forest F
2: while there exists an unscanned arc a = (u, v) ∈ AG with u^1 ∈ VF do
3:    Fetch an unscanned arc a
4:    if a ∈ M or v^2 ∈ VF then mark a as scanned and continue   [to Step 2]
5:    a_1 ⇐ (u^1, v^2)
6:    Let P_0 be the even M-alternating path to u^1 in F   [P_0 is feasible]
7:    if v^2 is a sink then   [single step]
8:       P_1 ⇐ P_0 ∘ a_1   [P_1 is augmenting]
9:       if P_1 is feasible then return (G, M △ A(P_1), NULL)
10:   else   [double step]
11:      Let a_2 = (v^2, w^1) be the unique arc leaving v^2 in G→(M)   [w^1 ∉ VF]
12:      P_1 ⇐ P_0 ∘ a_1 ∘ a_2   [P_1 is an M-alternating]
13:      if P_1 is feasible then
14:         Add nodes v^2 and w^1 and arcs a_1, a_2 to F
15:         Mark a as scanned
16:         continue   [to Step 2]
17:      end if
18:   end if
19:   M_0 ⇐ M △ A(P_0), M_1 ⇐ M △ A(P_1)
20:   Let C be the unique cycle in M_1
21:   Construct G', M' from G, M by contracting C
22:   if sparsify = FALSE then return FAST-EVEN-FACTOR(G', M', FALSE)
23:   Construct the digraph H' (see the text for an explanation)
24:   (H̄', M̄', F̄) ⇐ FAST-EVEN-FACTOR(H', M', FALSE)
25:   Compare VH̄' and VG; let Z_1, ..., Z_k be maximal contracted sets and z_1, ..., z_k be
      the corresponding composite nodes in H̄'
26:   Ḡ' ⇐ G/Z_1/.../Z_k
27:   if def(Ḡ', M̄') < def(G', M') then return (Ḡ', M̄', NULL)
28:   Mark as not scanned arcs of Ḡ' belonging to M and arcs entering z_1, ..., z_k
29:   (G, M, F) ⇐ (Ḡ', M̄', F̄)
30: end while
31: return (G, M, F)
```

Otherwise a contraction is necessary. At this point we have a feasible M-alternating path P_0 and an infeasible M-augmenting or M-alternating path P_1 (obtained from P_0 by appending one or two arcs). Define $M_0 := M \triangle A(P_0)$ and $M_1 := M \triangle A(P_1)$. Let C denote the unique odd cycle in M_1. Now C fits M_0, hence one can contract C and set $G' := G/C$ and $M' := M_0/C$. If *sparsify* = FALSE, then FAST-EVEN-FACTOR acts similar to NAIVE-EVEN-FACTOR, namely, it recurses to G', M' (with *sparsify* = FALSE) and, thus, restarts the whole augmenting path computation.

Let *sparsify* = TRUE. In this case the algorithm tries to *recover* some feasible M'-alternating forest in G'. To accomplish this, a certain sparse digraph H' containing M' is constructed (see below). Next FAST-EVEN-FACTOR is recursively called for H', M' (with *sparsify* = FALSE). The latter nested call may yield a breakthrough, that is, find an even factor of a smaller deficiency. In this case the outer call terminates

immediately. Otherwise by Lemma 2(b), the nested call returns a complete feasible $\overline{M}'$-alternating forest $\overline{\mathcal{F}}$ in $\overline{H}'$ (which is obtained from H' by contractions). This forest is used by the outer call to continue searching for augmenting paths. It turns out that nearly all arcs that were earlier fetched by the outer call need no additional processing and may remain scanned w.r.t. the new, recovered forest. This way arc scans are amortized.

Let us now describe how H' is constructed. First take the node set of G, add all arcs of M and all arcs $(u, v) \in AG$ such that $(u^1, v^2) \in A\mathcal{F}$. Denote the resulting digraph by H. Second, ensure that H is odd-cycle symmetric: if some arc (u, v) is already added to H and the reverse arc (v, u) exists in G, then add (v, u) to H. Third, set $H' := H/C$. Note that H' is a sparse spanning subgraph of G' (i.e. $VH' = VG'$, $|AH'| = O(n)$) and M' is an even factor in H'.

At Step 24 the algorithm recurses to FAST-EVEN-FACTOR(H', M', FALSE). Let $\overline{H}'$ and $\overline{M}'$ be the resulting digraph and the even factor, respectively. Compare the node sets of $\overline{H}'$ and H. Clearly $\overline{H}'$ can be viewed as obtained from H by a series of contractions (H/C being one of them). Let $Z_1, \ldots, Z_k$ denote maximal disjoint subsets of G such that $\overline{H}' = H/Z_1/\ldots/Z_k$. Let $z_1, \ldots, z_k$ denote composite nodes of $\overline{H}'$ that correspond to contracted subsets $Z_1, \ldots, Z_k$. The algorithm applies these contractions to G (rather than H) and constructs the digraph $\overline{G}' := G/Z_1/\ldots/Z_k$. Clearly $\overline{M}'$ is an even factor in both $\overline{G}'$ and $\overline{H}'$. If $\mathrm{def}(\overline{H}', \overline{M}') < \mathrm{def}(H', M') = \mathrm{def}(G, M)$, then a breakthrough is reached, so FAST-EVEN-FACTOR terminates returning $\overline{G}'$ and $\overline{M}'$.

Otherwise the recursive call in Step 24 also produces a complete feasible forest $\overline{\mathcal{F}}$ for $\overline{H}'$ and $\overline{M}'$. Recall that some arcs in G are marked as *scanned*. Since we identify arcs of $\overline{G}'$ with their pre-images in G, one may speak of scanned arcs in $\overline{G}'$. To ensure (1) the algorithm marks certain arcs $a = (u, v) \in A\overline{G}'$ as "not scanned", i.e. adds them back to their corresponding lists $L(u)$. Namely, it resets marks for arcs $a \in M$ that are present in $\overline{G}'$ and for arcs that enter $z_1, \ldots, z_k$. Finally, the algorithm resets $G := \overline{G}'$, $M := \overline{M}'$, $\mathcal{F} := \overline{\mathcal{F}}$ and proceeds with growing $\mathcal{F}$ (using the adjusted set of scanned arcs).

Suppose that FAST-EVEN-FACTOR had scanned all arcs of G and was unable to reach a sink. Then the resulting forest $\mathcal{F}$ is both feasible (by (1)(a)) and complete (by (1)(b)). In particular no augmenting path for M exists. By Claim 3 this implies the maximality of M in G. The algorithm returns the current digraph G, the current (maximum) even factor M, and also the forest $\mathcal{F}$ that certifies the maximality of M.

4.2 Correctness

The correctness of FAST-EVEN-FACTOR follows from Lemma 2, which, in turn, is a direct consequence of invariant (1). In this subsection we focus on proving the latter.

First, we shall need some more convenient characterization of $\mathcal{F}$-reachability. Consider an odd-cycle symmetric digraph Γ and a node $v \in V\Gamma$. Construct a new odd-cycle symmetric digraph $\Gamma * v$ from Γ by adding a new node v' and an arc (v, v').

Lemma 3 *For Γ and $v \in V\Gamma$ as above, either* $\mathrm{def}(\Gamma * v) = \mathrm{def}(\Gamma)$ *or* $\mathrm{def}(\Gamma * v) = \mathrm{def}(\Gamma) + 1$.

Proof Each even factor N in Γ is also an even factor in $\Gamma * v$, hence $\mathrm{def}(\Gamma * v) \leq \mathrm{def}(\Gamma) + 1$. Also for an even factor N^* in $\Gamma * v$, define $N := N^* \setminus \{(v, v')\}$. Then N is an even factor in Γ obeying $|N| \geq |N^*| - 1$. This implies $\mathrm{def}(\Gamma) \leq \mathrm{def}(\Gamma * v)$, as required. □

Lemma 4 *Let N be a maximum even factor in Γ and $\mathcal{T}$ be a feasible N-alternating forest. If v^1 is $\mathcal{T}$-reachable, then* $\mathrm{def}(\Gamma * v) = \mathrm{def}(\Gamma)$.

Proof Consider the N-alternating path P from a root of $\mathcal{T}$ to v^1. Since $\mathcal{T}$ is feasible, $N' := N \triangle A(P)$ is an even factor in Γ and no arc of N' leaves v. Now $N^* := N' \cup \{(v, v')\}$ is an even factor in $\Gamma * v$. Therefore $\mathrm{def}(\Gamma * v) \leq \mathrm{def}(\Gamma * v, N^*) = \mathrm{def}(\Gamma, N) = \mathrm{def}(\Gamma)$, which implies $\mathrm{def}(\Gamma * v) = \mathrm{def}(\Gamma)$ by Lemma 3. □

Lemma 5 *Let N be a maximum even factor in Γ and $\mathcal{T}$ be a complete N-alternating forest. If v^1 is not $\mathcal{T}$-reachable, then* $\mathrm{def}(\Gamma * v) = \mathrm{def}(\Gamma) + 1$ *and N is a maximum even factor in $\Gamma * v$.*

Proof Consider the N-alternating forest $\mathcal{T}^*$ obtained from $\mathcal{T}$ by adding a new root node $(v')^1$. One can see that $\mathcal{T}^*$ is complete for N in $\Gamma * v$. (Indeed, extending Γ to $\Gamma * v$ adds to $\widetilde{\Gamma}$ nodes $(v')^1$ and $(v')^2$ and an edge $\{v^1, (v')^2\}$ corresponding to the arc (v, v'). Note that $(v')^1$ has no incident edges and v^1 is not $\mathcal{T}$-reachable.) Now Claim 1 implies that N is also maximum in $\Gamma * v$, hence $\mathrm{def}(\Gamma * v) = \mathrm{def}(\Gamma) + 1$, as promised. □

Now consider the state of the algorithm right before Step 29. Let v be an arbitrary node in G. If $v \in VG - (Z_1 \cup \cdots \cup Z_k)$ then we say that v *survives contractions*. Nodes surviving contractions are both present in G and $\overline{G}'$ (and also in H and $\overline{H}'$).

Lemma 6 *Suppose that for $v \in VG$, node v^1 is $\mathcal{F}$-reachable in G. Let v_0 be the image of v in $\overline{G}'$ (and also in $\overline{H}'$) (i.e. $v_0 = v$ if v survives contractions or $v_0 = z_i$ for some i otherwise). Then v_0^1 is $\overline{\mathcal{F}}$-reachable in $\overline{H}'$.*

Proof The transformation of H into $\overline{H}'$ and of M into $\overline{M}'$ can be viewed as follows:

$$\begin{array}{ccccccccc} (H, M) & & (H, M_0) & & (H', M') & & & & (\overline{H}', \overline{M}') \\ \| & & \| & & \| & & & & \| \\ (\Gamma^0, N^0) & \to & (\Gamma^0, N_0^0) & \to & (\Gamma^1, N^1) & \to & (\Gamma^1, N_0^1) & \to \cdots \to & (\Gamma^s, N^s) \end{array} \tag{2}$$

Here $\Gamma^0, \ldots, \Gamma^s$ are odd-cycle symmetric digraphs, N^i, N_0^i are even factors in Γ^i obeying $|N^i| = |N_0^i|$. For each i there exists some odd cycle C^i in Γ^i that fits N_0^i. We inductively define $\Gamma^{i+1} := \Gamma^i / C^i$, $N^{i+1} := N_0^i / C^i$.

Recall that the nested call to FAST-EVEN-FACTOR in Step 24 did not yield a breakthrough, so $\overline{M}'$ is maximum in $\overline{H}'$ and $\overline{\mathcal{F}}$ is a complete $\overline{M}'$-alternating forest.

First assume that v survives contractions, i.e. $v_0 = v$. Suppose towards a contradiction that $v_0^1 = v^1$ is not $\overline{\mathcal{F}}$-reachable in $\overline{H}'$. By Lemma 5, $\overline{M}' = N^s$ is a maximum even factor in $\overline{H}' * v = \Gamma^s * v$. Then by Claim 3, N_0^{s-1} is a maximum even factor in $\Gamma^{s-1} * v$ and hence so is N^{s-1} (by the equality of sizes). Proceeding this way in the backward direction we conclude that for all $i = 0, \ldots, s$, N^i is a maximum even factor in $\Gamma^i * v$. In particular, $N^0 = M$ is maximum in $\Gamma^0 * v = H * v$. Since $(v, v') \notin M$, M is also maximum in H and $\mathrm{def}(H * v, M) = \mathrm{def}(H, M) + 1$. This contradicts Lemma 4 and the fact that v^1 is $\mathcal{F}$-reachable.

Now let $v_0 = z_i$ for some i and suppose that $v_0^1 = z_i^1$ is not $\overline{\mathcal{F}}$-reachable in $\overline{H}'$. Consider (2) and suppose that z_i is a composite node in Γ^j formed by contracting an odd cycle C^{j-1} in Γ^{j-1}. As shown earlier, N^j is a maximum even factor in $\Gamma^j * v$. The latter, however, is false since C^{j-1} fits N_0^{j-1} and therefore N^j has no arcs leaving v (cf. Fig. 1(b) for an example). Hence N^j can be enlarged to $N^j \cup \{(v, v')\}$—a contradiction. □

Lemma 7 *Suppose that for $v \in VG$ that survives contractions, node v^2 is $\mathcal{F}$-reachable in G. Then v^2 is $\overline{\mathcal{F}}$-reachable in $\overline{H}'$.*

Proof Since v^2 cannot be a source, it is reached by some arc $(u^1, v^2) \in A\mathcal{F}$. Here u^1 is $\mathcal{F}$-reachable and $a = (u, v) \notin M$. Let $a_0 = (u_0, v)$ be the image of a under contractions. According to Lemma 6 node u_0^1 is $\overline{\mathcal{F}}$-reachable. Observe that $a \in AH$ and $a_0 \in A\overline{H}'$. If $a_0 \notin \overline{M}'$, then v^2 is $\overline{\mathcal{F}}$-reachable (by completeness of $\overline{\mathcal{F}}$). Now assume $a_0 \in \overline{M}'$. Then u_0^1 is not a source (since $a_0 \in \overline{M}'$ leaves u_0) but is $\overline{\mathcal{F}}$-reachable and is entered by a unique arc, namely (v^2, u_0^1). Hence v^2 must be $\overline{\mathcal{F}}$-reachable, as required. □

Now everything is ready to complete the proof of correctness.

Lemma 8 *Properties* (1) *are maintained throughout the execution of* FAST-EVEN-FACTOR.

Proof Property (1)(a) is clearly preserved. Changes to the set of scanned arcs in Step 4 and Step 15 are consistent with (1)(b). It remains to prove that the latter is maintained when FAST-EVEN-FACTORY tries to recover $\mathcal{F}$ and updates the set of scanned arcs.

Consider the moment right before Step 29. Let $a = (u, v)$ be an arbitrary arc in $\overline{G}'$. We argue that either a is not marked as scanned, or $a \in \overline{M}'$, or both u^1 and v^2 are $\overline{\mathcal{F}}$-reachable.

Indeed, consider a scanned arc $a = (u, v)$ and let $a_0 = (u_0, v_0)$ be its pre-image in G. Here $u = u_0$ if u_0 survives contractions and $u = z_i$ if $u_0 \in Z_i$ for some i. Since all arcs entering $z_1, \ldots, z_k$ are marked as not scanned in Step 28, v_0 survives contractions and hence $v = v_0$, so $a_0 = (u_0, v)$. Since the algorithm only decreases the set of scanned arcs in Step 28, a_0 must also be scanned in G. Clearly $a_0 \notin M$ since arcs

belonging to M that are present in $\overline{G}'$ were marked as not scanned. Therefore, both u_0^1 and v^2 are $\mathcal{F}$-reachable in G by (1)(b). Applying Lemma 6 to u_0 and Lemma 7 to v we see that both u and v are $\overline{\mathcal{F}}$-reachable. □

4.3 Implementation Details and Complexity

We employ arc lists to represent digraphs. When a subset U in a digraph Γ is contracted we enumerate arcs incident to U and update the lists accordingly. If a pair of parallel arcs appears after contraction, these arcs are merged, so all the digraph remains simple. The above contraction of U takes $O(|V\Gamma| \cdot |U|)$ time. During FAST-EVEN-FACTOR the sum of sizes of the contracted subsets telescopes to $O(n)$, so graph contractions take $O(n^2)$ time in total. A usual bookkeeping allows to undo contractions and recover a maximum even factor in the original digraph in $O(m)$ time.

Let us explain how path feasibility checks, which are performed by FAST-EVEN-FACTOR at Steps 9 and 13, can be made efficient. That is, given an even factor N and a feasible N-alternating path P_0 we need to verify that an N-alternating or an N-augmenting path P_1 (obtained from P_0 by appending one or two arcs) is feasible. We make use of a certain data structure $\mathcal{D}$ that maintains $N \triangle A(P_0)$ as a collection of node-disjoint paths and cycles. The following operations are supported by $\mathcal{D}$:

- INSERT(u, v): assuming that u is the end node of some path P_u in $\mathcal{D}$ and v is the start node of some path P_v in $\mathcal{D}$, add the arc (u, v) thus linking P_u and P_v or, in case $P_u = P_v$, turning this path into a cycle;
- REMOVE(u, v): assuming that $a = (u, v)$ is an arc belonging to some path or cycle in $\mathcal{D}$, remove a thus splitting the path into two parts or turning the cycle into a path;
- IS-ODD-CYCLE(u, v): assuming that $a = (u, v)$ is an arc belonging to some path or cycle in $\mathcal{D}$, check if a belongs to an odd cycle.

We make use of balanced search trees additionally augmented with SPLIT and CONCATENATE operations (e.g. red-black trees or splay trees, see [3, 19]) to represent paths and cycles in $\mathcal{D}$. This way, INSERT, REMOVE, and IS-ODD-CYCLE take $O(\log |V\Gamma|)$ time each. Now checking if P_1 is feasible is done by calling INSERT(u, v) and, in case of a double step, REMOVE(w, v), and finally making IS-ODD-CYCLE(u, v) request. If the latter indicates that P_1 is not feasible, the changes in $\mathcal{D}$ are rolled back.

During FAST-EVEN-FACTOR$(\Gamma, N,$ FALSE$)$ we grow $\mathcal{F}$ in a depth-first fashion and maintain the structure $\mathcal{D}$ corresponding to the current $\mathcal{F}$-reachable node u^1 (i.e. $\mathcal{D}$ keeps the decomposition of $N \triangle A(P)$ where P is the path in $\mathcal{F}$ from a root to u^1). When $\mathcal{F}$ is extended by arcs (u^1, v^2) and (v^2, w^1), w^1 becomes the new current node and $\mathcal{D}$ is updated accordingly by the above INSERT(u, v) and REMOVE(w, v) calls. When the algorithm backtracks from w^1 to u^1, changes in $\mathcal{D}$ are reverted. This way, each feasibility check costs $O(\log |V\Gamma|)$ time.

Next, consider an invocation of FAST-EVEN-FACTOR$(\Gamma, N,$ TRUE$)$. The above time bound of $O(\log |V\Gamma|)$ per check is only valid if we grow $\mathcal{F}$ from scratch. However, the algorithm may reuse the forest that is returned by the nested FAST-EVEN-FACTOR call in Step 24. This incurs an overhead of $O(|V\Gamma| \log |V\Gamma|)$ per forest

recovery (this additional time is needed to traverse some arcs that are present in the recovered forest $\mathcal{F}$ and update $\mathcal{D}$ accordingly). There are $O(|V\Gamma|)$ forest recoveries during the call and the total overhead is $O(|V\Gamma|^2 \log |V\Gamma|)$. This does not affect the time bound of FAST-EVEN-FACTOR(Γ, N, TRUE).

Now consider an invocation FAST-EVEN-FACTOR(Γ, N, FALSE) and let us bound its complexity (including recursive calls). The outer loop of the algorithm (Steps 2–30) enumerates unscanned arcs. Since *sparsify* = FALSE, each arc can be scanned at most once, so the bound of $O(|A\Gamma|)$ for the number of arc scans follows. Reachability checks in Steps 9 and 13 cost $O(\log |V\Gamma|)$ time. Constructing M_0, M_1, and C takes $O(|V\Gamma|)$ time. This way, FAST-EVEN-FACTOR(Γ, N, FALSE) takes $O((k+1)|A\Gamma| \log |V\Gamma|)$ time, where k denotes the number of graph contractions performed during the invocation.

Next, we focus on an invocation of FAST-EVEN-FACTOR(Γ, N, TRUE). More than $|A\Gamma|$ arc scans are possible since forest recovery may produce new unscanned arcs (Step 28). Note that forest recovery totally occurs $O(|V\Gamma|)$ times (since each such occurrence leads to a contraction). During each recovery M generates $O(|V\Gamma|)$ unscanned arcs or, in total, $O(|V\Gamma|^2)$ such arcs for the duration of FAST-EVEN-FACTOR. Also each node z_i generates $O(|V\Gamma|)$ unscanned arcs (recall that we merge parallel arcs and keep the current digraph simple). The total number of these nodes processed during FAST-EVEN-FACTOR is $O(|V\Gamma|)$ (since each such node corresponds to a contraction). Totally these nodes produce $O(|V\Gamma|^2)$ unscanned arcs. Hence the total number of arc scans is $O(|A\Gamma| + |V\Gamma|^2) = O(|V\Gamma|^2)$.

Each feasibility check costs $O(\log |V\Gamma|)$ time, or $O(|V\Gamma|^2 \log |V\Gamma|)$ in total. Finally, we must account for the time spent in recursive invocations during FAST-EVEN-FACTOR(Γ, N, TRUE). Each such invocation deals with a sparse digraph and thus takes $O((k+1)|V\Gamma| \log |V\Gamma|)$ time (where, as earlier, k denotes the number of contractions performed by the recursive invocation). The total number of contractions is $O(|V\Gamma|)$, so the sum over all recursive invocations telescopes to $O(|V\Gamma|^2 \log |V\Gamma|)$.

The total time bound for FAST-EVEN-FACTOR(Γ, N, TRUE) (including recursive calls) is also $O(|V\Gamma|^2 \log |V\Gamma|)$. Therefore a maximum even factor in an odd-cycle symmetric digraph can be found in $O(n^3 \log n)$ time, as claimed.

5 Extension to Square-Free Simple 2-Matchings

The presented sparse recovery approach looks rather generic and applies to other problems that involve similar augmentation techniques. In this section we consider square-free simple 2-matchings and give an improved version of Pap's algorithm for constructing one of maximum cardinality.

5.1 Notation and Definitions

From now on we consider an undirected bipartite graph G together with a fixed bipartition $X \sqcup Y = VG$. Let G be endowed with node capacities $b\colon VG \to \{1, 2\}$. By a *square* we mean a simple circuit of length 4. The following is a counterpart of Definition 1:

Definition 6 Given G as above, a *simple b-matching* is a subset of edges M such that $|M \cap \delta(v)| \leq b(v)$ for each $v \in VG$. A simple b-matching is called *square-free* if it does not contain (the edge set) of a square. The *size* of M is its cardinality and the *maximum square-free simple b-matching problem* consists in finding a square-free simple b-matching of maximum size.

Let the *deficiency* of a simple b-matching M in G be $\mathrm{def}(G, b, M) := \sum_{v \in Y} b(v) - |M|$. The minimum of $\mathrm{def}(G, b, M)$ over all square-free simple b-matchings M will be denoted by $\mathrm{def}(G, b)$.

Since $b(v)$ is at most 2 for every node v, a simple b-matching decomposes into a collection of node-disjoint paths and circuits (that must be even since G is bipartite). The problem already looks non-trivial when $b(v) \equiv 2$ for all v. We allow $b(v) = 1$ since such nodes v eventually arise during contractions performed by the algorithm (see below).

Like for even factors, given a simple b-matching M, one may consider the standard residual bipartite digraph $\overrightarrow{G}(M)$, which captures information about possible augmentations. Since G is bipartite, we no longer need to split nodes. Instead we define $V\overrightarrow{G}(M) := VG$. Each edge $e = \{x, y\} \in EG$ (where $x \in X$, $y \in Y$) gives rise to an arc $(x, y) \in A\overrightarrow{G}(M)$ if $e \notin M$ and to an arc $(y, x) \in A\overrightarrow{G}(M)$ if $e \in M$. Set $c_M(v) := |M \cap \delta(v)|$ and call $v \in X$ (respectively $v \in Y$) a *source* (respectively a *sink*) if $c_M(v) < b(v)$, i.e. v is not saturated by M w.r.t. capacities b.

We borrow the notions of M-alternating and M-augmenting paths (see Definition 2) and extend them to simple b-matchings. For an M-augmenting or an M-alternating path P in $\overrightarrow{G}(M)$, let $E(P)$ be the set of edges of G that correspond to arcs of P. This enables to speak of feasible M-alternating and M-augmenting paths (cf. Definition 5):

Definition 7 Given a square-free simple b-matching M, let P be an M-augmenting or an M-alternating path. Then P is called *feasible* if the simple b-matching $M' := M \triangle E(P)$ is again square-free.

Now the optimality criterion (cf. Claim 1) and a counterpart of Claim 2 read as follows:

Claim 4 *M is a simple b-matching of maximum size iff $\overrightarrow{G}(M)$ contains no M-augmenting path.*

Claim 5 *If P is an M-augmenting (respectively an M-alternating) path in $\overrightarrow{G}(M)$, then $M' := M \triangle E(P)$ is a simple b-matching with $|M'| = |M| + 1$ (respectively $|M'| = |M|$).*

5.2 Contractions

Defining contractions for simple b-matchings is more delicate. First consider a square Q in G. It may be viewed as an isomorphic copy of $K_{2,2}$ induced by certain nodes

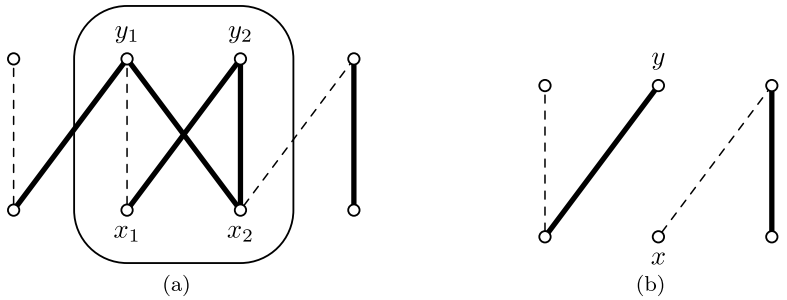

Fig. 3 Contracting a square. (**a**) Edges of N are *bold*, others are *dashed*, the outline of Q is marked. (**b**) Q is contracted, edges of N' are *bold*, others are *dashed*

$\{x_1, x_2, y_1, y_2\}$ where $x_1, x_2 \in X,\ y_1, y_2 \in Y$ (i.e. $EQ = \{\{x_i, y_j\} \mid i = 1, 2,\ j = 1, 2\}$). Suppose $b(x_i) = b(y_i) = 2$ for all i. Then Q is said to *fit* a square-free simple b-matching N if $N \cap EQ = \{\{x_1, y_2\}, \{x_2, y_1\}, \{x_2, y_2\}\}$ and x_1 is a source (w.r.t. N); see Fig. 3(a).

To contract Q in G, one merges x_1 and x_2 and also y_1 and y_2 into *composite* nodes x and y (respectively), removes the edges of Q from G, and finally redirects edges incident to x_i (respectively y_i) to be incident to x (respectively y). One also defines $b'(x) := b'(y) := 1$ and $b'(v) := b(v)$ for all $v \in VG - VQ$. This gives rise to a new undirected bipartite graph G' (denoted by G/Q) and a simple b'-matching $M' := M \setminus EQ$ (denoted by M/Q); see Fig. 3(b). The properties of this contraction procedure turn out to be analogous to those for even factors (see [15]; cf. Claim 3 and Lemma 1):

Claim 6 *Let N be a square-free simple b-matching in G and Q be a square that fits N. Then for G', b' and N' defined above, N' is a square-free simple b'-matching in G'. Moreover, if N' is maximum in G', then N is maximum in G.*

Lemma 9 *Let Q be a square in G and let N' be a square free simple b'-matching in G'. Then there exists a square-free simple b-matching N in G with* $\mathrm{def}(G, b, N) = \mathrm{def}(G', b', N')$.

Figure 4 demonstrates some cases arising in Lemma 9 during uncontraction of Q.

5.3 Naive Algorithm

One may solve the problem with an algorithm conceptually similar to that described in Sect. 3. The algorithm (called NAIVE-SQUARE-FREE) performs a series of iterations. On each iteration we are given a square-free simple b-matching and look for an augmenting path P. The iterations terminate when no such path exists. If P exists and is feasible then resetting $M := M \triangle E(P)$ increases the size of M. Otherwise (P exists but is not feasible) a reduction is needed. We find the maximum prefix P' of P that is a feasible M-alternating path and reset $M := M \triangle E(P')$. It can be shown [15] that there exists a square Q fitting the current M. Set $G' := G/Q$, $M' := M/Q$ and

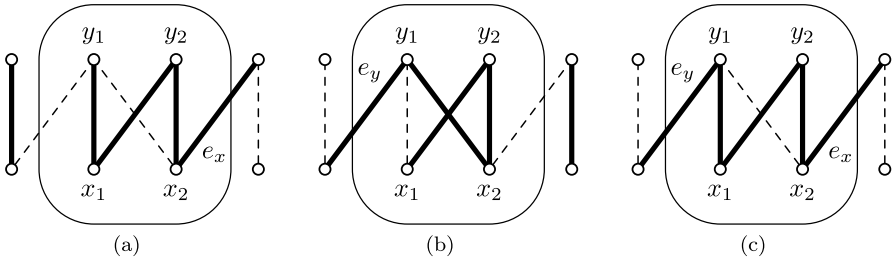

Fig. 4 Possible configurations appearing in the proof of Lemma 9. Graph G and a square-free simple b-matching N are depicted. Edges of N are *bold*, others are *dashed*. (**a**) Node x has an incident edge $e_x \in N'$ but y has no such edge. (**b**) Node y has an incident edge $e_y \in N'$ but x has no such edge. (**c**) Both x and y have incident edges $e_x, e_y \in N'$ (respectively). The subcase when neither x nor y has an incident edge of N' in G' is omitted due to its triviality

define $b'\colon VG' \to \{1, 2\}$ as explained above. Now in view of Claim 6 and Lemma 9 it remains to augment M' in G' (w.r.t. b').

To bound the complexity of NAIVE-SQUARE-FREE, observe that there are $O(n)$ iterations and $O(n)$ phases (contractions) during each iteration. Computing P from scratch takes $O(m)$ time. Finally, checking P for feasibility and computing the maximum feasible M-alternating prefix of P takes $O(n)$ time. (In contrast to even factors, where checking for evenness of cycles forming a path-matching N is a non-local operation that requires decomposing N, checking for absense of squares in $M \triangle E(P')$ is simpler and takes $O(1)$ per each pair of consequent edges of P'.) Thus we get an $O(mn^2)$-time algorithm. To do better we need to apply sparsification as explained below.

5.4 Faster Algorithm

As in Sect. 4 we need an explicit description of the forest-growing process. The notions of *M-alternating forest* $\mathcal{F}$, *complete* and *feasible* alternating forests, *$\mathcal{F}$-reachable*, and *$\mathcal{F}$-unreachable* nodes remain the same. (Note that they only depend on $\overrightarrow{G}(M)$ but not on the underlying G.) From the high-level perspective the algorithm (called FAST-SQUARE-FREE, see Algorithm 3 for a pseudocode) works similar to FAST-EVEN-FACTOR. There are several noticeable differences, though.

First, G is undirected so FAST-SQUARE-FREE deals with edges of G rather than arcs. Also nodes of G directly correspond to nodes of $\overrightarrow{G}(M)$.

During recovery we construct a sparse spanning bipartite subgraph H' of G' as follows. First take the node set of G, add all edges of M, and edges $\{u, v\} \in EG$ such that $(u, v) \in A\mathcal{F}$. Denote the resulting graph by H. Then define $H' := H/Q$ thus obtaining a sparse spanning subgraph of G' with a square-free simple b'-matching M'.

Denote by $\overline{H}'$ and $\overline{M}'$ a graph and a square-free simple $\overline{b}'$-matching obtained by recursing to FAST-SQUARE-FREE(H', b', M', FALSE). Compare of $\overline{H}'$ and H. Let $Q_1, \ldots, Q_k$ be squares contracted in H. (Unlike even factors where contractions can be nested, these squares are always disjoint. Indeed, contractions only apply to nodes with capacity 2 and produce nodes with capacity 1. This simplification does

Algorithm 3 FAST-SQUARE-FREE(G, b, M, *sparsify*)

1: Initialize forest $\mathcal{F}$
2: **while** there exists an unscanned edge $e = \{u, v\} \in EG$ with $u \in X$, $v \in Y$ **do**
3: Fetch an unscanned edge e
4: **if** $e \in M$ or $v \in V\mathcal{F}$ **then** mark e as scanned and **continue** [*to Step 2*]
5: $a_1 \Leftarrow (u, v)$
6: Let P_0 be the M-alternating path to u in $\mathcal{F}$ [P_0 *is feasible*]
7: **if** v is a sink **then** [*single step*]
8: $P_1 \Leftarrow P_0 \circ a_1$ [P_1 *is augmenting*]
9: **if** P_1 is feasible **then return** $(G, M \triangle E(P_1), \text{NULL})$
10: **else** [*double step*]
11: Let $a_2 = (v, w)$ be the unique arc leaving v in $\overrightarrow{G}(M)$ [$w \notin V\mathcal{F}$]
12: $P_1 \Leftarrow P_0 \circ a_1 \circ a_2$ [P_1 *is* M*-alternating*]
13: **if** P_1 is feasible **then**
14: Add nodes v and w and arcs a_1, a_2 to $\mathcal{F}$
15: Mark e as scanned
16: **continue** [*to Step 2*]
17: **end if**
18: **end if**
19: $M_0 \Leftarrow M \triangle E(P_0)$, $M_1 \Leftarrow M \triangle E(P_1)$
20: Let Q be the unique square in M_1
21: Construct G', b', M' from G, b, M by contracting Q
22: **if** *sparsify* = FALSE **then return** FAST-SQUARE-FREE$(G', b', M'$, FALSE$)$
23: Construct graph H' (see the text for an explanation)
24: $(\overline{H}', \overline{b}', \overline{M}', \overline{\mathcal{F}}) \Leftarrow$ FAST-SQUARE-FREE$(H', b', M'$, FALSE$)$
25: Compare $V\overline{H}'$ and VG; let $Q_1, \ldots, Q_k$ be contracted squares and $(x_1, y_1), \ldots, (x_k, y_k)$ be the corresponding pairs of composite nodes in $\overline{H}'$ ($x_i \in X$, $y_i \in Y$)
26: $\overline{G}' \Leftarrow G/Q_1/\ldots/Q_k$
27: **if** $\text{def}(\overline{G}', \overline{b}', \overline{M}') < \text{def}(G', b', M')$ **then return** $(\overline{G}', \overline{b}', \overline{M}', \text{NULL})$
28: Mark as not scanned edges of $\overline{G}'$ belonging to M and edges incident to $y_1, \ldots, y_k$
29: $(G, b, M, \mathcal{F}) \Leftarrow (\overline{G}', \overline{b}', \overline{M}', \overline{\mathcal{F}})$
30: **end while**
31: **return** $(G, b, M, \mathcal{F})$

not seem to help much, however.) Define $\overline{G}' := G/Q_1/\ldots/Q_k$ and let $x_1, \ldots, x_k \in X$ and $y_1, \ldots, y_k \in Y$ be the composite nodes of $\overline{H}'$ that correspond to contracted squares $Q_1, \ldots, Q_k$. Assuming that the deficiency check in Step 27 did not succeed, the algorithm marks as "not scanned" edges of M that are still present in $\overline{G}'$ and also edges incident to $y_1, \ldots, y_k$. Then it continues to grow the M-alternating forest $\mathcal{F}$.

5.5 Correctness

The correctness of FAST-SQUARE-FREE is implied by the following

Lemma 10 (Cf. Lemma 2) FAST-SQUARE-FREE *gets an undirected graph* G, *node capacities* $b\colon VG \to \{1, 2\}$, *a square-free simple* b*-matching* M *in* G, *and an addi-*

tional "sparsify" flag. It returns an undirected graph $\overline{G}$ *obtained from* G *by a series of contractions, capacities* $\overline{b}\colon V\overline{G} \to \{1,2\}$ *and a simple square free* $\overline{b}$*-matching* $\overline{M}$ *in* $\overline{G}$*. Additionally it may return an* $\overline{M}$*-alternating forest* $\overline{\mathcal{F}}$ *in* $\overline{G}$*. Exactly one of the following cases applies*:

(a) $\operatorname{def}(\overline{G},\overline{b},\overline{M}) = \operatorname{def}(G,b,M) - 1$ *and* $\overline{\mathcal{F}}$ *is undefined; or*
(b) $\operatorname{def}(\overline{G},\overline{b},\overline{M}) = \operatorname{def}(G,b,M)$, $\overline{M}$ *is maximum in* $\overline{G}$, M *is maximum in* G, *and* $\overline{\mathcal{F}}$ *is a complete feasible* $\overline{M}$*-alternating forest in* $\overline{G}$.

The proof proceeds analogously to that of Lemma 2 and relies on the following invariants that are maintained during the execution of FAST-SQUARE-FREE (cf. (1)):

(a) $\mathcal{F}$ is a feasible M-alternating forest;
(b) If an edge $e = \{u,v\}$ with $u \in X$, $v \in Y$ is scanned, then either $a \in M$ or both u and v are $\mathcal{F}$-reachable. (3)

Property (3)(a) is obvious while (3)(b) will be established below. □

As in Sect. 4.2, we characterize the set of reachable nodes. For an undirected bipartite graph Γ with a bipartition $X \sqcup Y = V\Gamma$, capacities $b\colon V\Gamma \to \{1,2\}$, and a node $v \in X$, let $\Gamma * v$ denote the bipartite graph formed from Γ by adding a new node v', connecting it to v, and setting $b(v') := 1$. (Obviously v' goes to Y.)

The next statements are direct analogues of Lemmata 3, 4, and 5:

Lemma 11 *For a bipartite graph* Γ *and a node* v *as above, either* $\operatorname{def}(\Gamma * v, b) = \operatorname{def}(\Gamma, b)$ *or* $\operatorname{def}(\Gamma * v, b) = \operatorname{def}(\Gamma, b) + 1$.

Lemma 12 *Let* N *be a maximum square-free simple* b*-matching in* Γ *and* $\mathcal{T}$ *be a feasible* N*-alternating forest. If* $v \in X$ *is* $\mathcal{T}$*-reachable then* $\operatorname{def}(\Gamma * v, b) = \operatorname{def}(\Gamma, b)$.

Lemma 13 *Let* N *be a maximum square-free simple* b*-matching in* Γ *and* $\mathcal{T}$ *be a complete* N*-alternating forest. If* $v \in X$ *is not* $\mathcal{T}$*-reachable then* $\operatorname{def}(\Gamma * v, b) = \operatorname{def}(\Gamma, b) + 1$ *and* N *is a maximum square-free simple* b*-matching in* $\Gamma * v$.

Based on these results we now establish $\overline{\mathcal{F}}$-reachability of certain nodes in $\overline{G}'$. Consider the state of the algorithm right before Step 29. Nodes in $VG - (VQ_1 \cup \cdots \cup VQ_k)$ are said to *survive contractions*. These nodes are both present in G and $\overline{G}'$.

Lemma 14 (Cf. Lemma 6) *Suppose that a node* $v \in X$ *is* $\mathcal{F}$*-reachable. Let* v_0 *be the image of* v *in* $\overline{G}'$ *(and also in* $\overline{H}'$*) under contractions (i.e.* $v_0 = v$ *if* v *survives contractions or* $v_0 = x_i$ *for some* i*). Then* v_0 *is* $\overline{\mathcal{F}}$*-reachable.*

The proof is parallel to that of Lemma 6. □

Lemma 15 (Cf. Lemma 7) *Suppose that an* $\mathcal{F}$*-reachable node* $v \in Y$ *survives contractions. Then* v *is* $\overline{\mathcal{F}}$*-reachable in* $\overline{G}'$.

The proof proceeds similar to Lemma 7. Namely, since $v \in Y$, it cannot be a source, so v must be reached by some arc $(u, v) \in A\mathcal{F}$. Here u is $\mathcal{F}$-reachable and $e = \{u, v\} \notin M$. Let $e_0 = \{u_0, v\}$ be the image of e under contractions, i.e. $u_0 = u$ if u survives contractions and u_0 is the composite node containing u otherwise. In both cases u_0 is $\overline{\mathcal{F}}$-reachable (cf. Lemma 14). Note that $e \in EH$ and $e_0 \in E\overline{H}'$. If $e_0 \notin \overline{M}'$, then v is $\overline{\mathcal{F}}$-reachable (by completeness of $\overline{\mathcal{F}}$). Let $e_0 \in \overline{M}'$. Then u_0 cannot be a source. (This is where we distinguish ourselves from Lemma 7: e_0 is incident to u_0 and the latter, being a composite node, has unit capacity.) Hence u_0 is $\overline{\mathcal{F}}$-reachable and is entered by a unique arc, namely (v, u_0). Therefore v must be $\overline{\mathcal{F}}$-reachable, as required. □

Now Lemmata 14 and 15 imply that (3)(b) is preserved.

5.6 Complexity

The analysis from Sect. 4.3 extends to FAST-SQUARE-FREE in a straightforward way. Also when the current square-free simple b-matching is altered by adding one edge and (possibly) removing another, the new b-matching can be easily checked for presence of squares in $O(1)$ time. In particular, there is no need for a special data structure for maintaining connected components of the current solution. This saves $O(\log n)$-factor in complexity so the whole algorithm runs in $O(n^3)$ time.

6 Conclusions

We have presented the *sparse recovery* technique which enables to achieve a speed-up in various algorithms that rely on augmenting paths and contractions. Certain related problems are still to be addressed. For example, the algorithm of Takazawa [18] solves the weighted even factor problem in $O(mn^3)$ time and also involves recomputing the alternating forest from scratch on each phase. The *maximum independent even factor problem* [10], which involves matroids, is also solvable by the methods similar to those discussed above.

Acknowledgement The author is thankful to the anonymous referees for useful comments and suggestions.

References

1. Babenko, M.: A faster algorithm for the maximum even factor problem. In: Proc. 21st International Symposium on Algorithms and Computation, pp. 451–462 (2010)
2. Balas, E., Pulleyblank, W.: The perfectly matchable subgraph polytope of an arbitrary graph. Combinatorica **9**, 321–337 (1989)
3. Cormen, T., Stein, C., Rivest, R., Leiserson, C.: Introduction to Algorithms. McGraw-Hill Higher Education, Boston (2001)
4. Cunningham, W.H.: Matching, matroids, and extensions. Math. Program. **91**(3), 515–542 (2002)
5. Cunningham, W.H., Geelen, J.F.: The optimal path-matching problem. Combinatorica **17**, 315–337 (1997)
6. Cunningham, W.H., Geelen, J.F.: Combinatorial algorithms for path-matching. Manuscript (2000)

7. Cunningham, W.H., Geelen, J.F.: Vertex-disjoint dipaths and even dicircuits. Manuscript (2001)
8. Hartvigsen, D.: Finding maximum square-free 2-matchings in bipartite graphs. J. Comb. Theory, Ser. B
9. Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM J. Comput. **2**(4), 225–231 (1973)
10. Iwata, S., Takazawa, K.: The independent even factor problem. SIAM J. Discrete Math. **22**, 1411–1427 (2008)
11. Király, Z.: C_4-free 2-factors in bipartite graphs. EGRES Technical Report TR-2001-13 (2001)
12. Kobayashi, Y., Takazawa, K.: Even factors, jump systems, and discrete convexity. J. Comb. Theory, Ser. B **99**(1), 139–161 (2009)
13. Micali, S., Vazirani, V.: An $O(\sqrt{|V|}\cdot|E|)$ algorithm for finding maximum matching in general graphs. In: Proc. 45st IEEE Symp. Foundations of Computer Science, pp. 248–255 (1980)
14. Pap, G.: A combinatorial algorithm to find a maximum even factor. In: Proc. 11th Integer International IPCO Conference on Programming and Combinatorial Optimization, pp. 66–80 (2005)
15. Pap, G.: Combinatorial algorithms for matchings, even factors and square-free 2-factors. Math. Program. **110**(1), 57–69 (2007)
16. Pap, G., Szegö, L.: On the maximum even factor in weakly symmetric graphs. J. Comb. Theory, Ser. B **91**(2), 201–213 (2004)
17. Spille, B., Weismantel, R.: A generalization of Edmonds' matching and matroid intersection algorithms. In: Proc. 9th International IPCO Conference on Integer Programming and Combinatorial Optimization, pp. 9–20 (2002)
18. Takazawa, K.: A weighted even factor algorithm. Math. Program., Ser. A, B **115**(2), 223–237 (2008)
19. Tarjan, R.: Data Structures and Network Algorithms. Society for Industrial and Applied Mathematics, Philadelphia (1983)