Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Combinatorial Theory, Series B 96 (2006) 693–705

Journal of Combinatorial Theory Series B

www.elsevier.com/locate/jctb

Finding maximum square-free 2-matchings in bipartite graphs ☆

David Hartvigsen

Mendoza College of Business, University of Notre Dame, Notre Dame, IN 46556-5646, USA

Received 7 November 2000

Available online 20 February 2006

Abstract

A 2-*matching* in a simple graph is a subset of edges such that every node of the graph is incident with at most two edges of the subset. A *maximum* 2-*matching* is a 2-matching of maximum size. The problem of finding a maximum 2-matching is a relaxation of the problem of finding a Hamilton tour in a graph. In this paper we study, in bipartite graphs, a problem of intermediate difficulty: The problem of finding a maximum 2-matching that contains no 4-cycles. Our main result is a polynomial time algorithm for this problem. We also present a min–max theorem.

Keywords: Matchings; Hamilton cycles

1. Introduction

All graphs considered in this paper are simple (i.e., they contain no multiple edges) and undirected. A 2-*matching* of a graph $G = (V, E)$ is a subset of E such that every node of G is incident with at most two edges of the subset. (Hence a 2-matching induces simple paths and cycles in G that are pairwise node disjoint. In the literature, this is often referred to as a "simple" 2-matching; for the sake of conciseness, we drop the adjective "simple.") A 2-matching M of G is called a 2-*factor* if every node of G is incident with exactly two edges of M. A *maximum* 2-*matching* in

☆ An extended abstract "The square-free 2-factor problem in bipartite graphs" of this paper appeared in: G. Cornuéjols, R. Burkard, G.J. Woeginger (Eds.), Integer Programming and Combinatorial Optimization, Lecture Notes in Comput. Sci., vol. 1610, Springer, Berlin, 1999, pp. 234–240.

E-mail address: hartvigsen.1@nd.edu.

0095-8956/$ – see front matter
doi:10.1016/j.jctb.2006.01.004

G is a 2-matching of maximum size or cardinality. Observe that if a graph has a 2-factor, then any maximum 2-matching is a 2-factor.

A *k-restricted 2-matching*, for $k \geqslant 2$ and integer, is a 2-matching with no induced cycles of length k or less. Let us denote by P_k the problem of finding a k-restricted 2-matching of maximum size. Thus P_2 is the problem of finding a maximum 2-matching in a graph. Note that P_k, for $n/2 \leqslant k \leqslant n-1$ (where n is the number of nodes in the graph), includes the problem of finding a Hamilton cycle in a graph. (This matching generalization was first developed and studied in [3,9].)

The basic problem P_2 has been extensively studied. A polynomial-time algorithm exists due to a reduction (see Tutte [23]) to the classical matching problem (see Edmonds [8]). Also well known is a "good" characterization of those graphs that have a 2-factor (due to Tutte [22]) and a polyhedral characterization of the convex hull of incidence vectors of 2-matchings (which can also be obtained using Tutte's reduction and Edmonds' polyhedral result for matchings [7]). For a more complete history of this problem see Schrijver [20].

Of course a polynomial-time algorithm for P_k, for $n/2 \leqslant k \leqslant n-1$, is unlikely since the problem of deciding if a graph has a Hamilton cycle is NP-complete. In fact, Papadimitriou (see [3]) showed that deciding if a graph has a k-restricted 2-matching for $k \geqslant 5$ is NP-complete. Complexity results for other variations on this problem appear in [14]. A polynomial-time algorithm for P_3 was given in [12]. (The algorithm is quite complex.) This leaves P_4 whose status is open. (A polynomial-time algorithm for a special case appears in [17]; this algorithm is also quite complex.)

In this paper we consider the problems P_k on bipartite graphs. Observe that for bipartite graphs the problems P_k of interest are for $k \geqslant 2$ and even. Again, for $n/2 \leqslant k \leqslant n-1$, a polynomial-time algorithm is unlikely since the problem of deciding if a bipartite graph has a Hamilton cycle is NP-complete (see Krishnamoorty [16]). Indeed, Geelen [11] has shown that P_6 is NP-hard for bipartite graphs. Hence we focus our attention in this paper on the problem P_4 in bipartite graphs. (The idea of studying this problem was suggested to the author by Cunningham and Geelen [5].) We refer to a cycle with four edges as a *square*, hence we refer to the problem P_4 on bipartite graphs as the problem of finding a maximum square-free 2-matching in a bipartite graph.

Our main result is a polynomial-time algorithm for finding maximum square-free 2-matchings in bipartite graphs. The algorithm uses techniques similar to those used for the problem P_3 on general graphs, but the resulting algorithm is simpler (it is hoped that some of the ideas developed in this paper will result in a simpler algorithm for P_3 in general graphs). A key idea that is used in the algorithm is an operation called *square-shrinking* (where a square is "shrunk" to an edge). We also present a min–max theorem that characterizes the size of a maximum square-free 2-matching. This theorem is in the spirit of the so-called Tutte–Berge theorem for classical matchings due to Berge [2] (which builds on a theorem of Tutte [21]). The statement of the min–max theorem in this paper is an improvement on a related theorem that appeared in [13] and is due to Kiraly. (He gives a direct, non-algorithmic proof in [15].) The proof in this paper is algorithmic. It is essentially a bi-product of the proof of the algorithm's validity. A generalization of this result due to Frank appears in [10]. (He also gives a direct, non-algorithmic proof.)

Let us briefly mention a related class of problems: The "weighted" versions of the problems P_k. That is, given a graph with non-negative edge weights, let *weighted-P_k* be the problem of finding a k-restricted 2-matching such that the sum of the weights of its edges is a maximum. Vornberger [24] showed that weighted-P_4 is NP-hard. Thus, for general graphs, only the complexities of P_4 and weighted-P_3 are unknown. Furthermore, it has been observed in [4,19], and by Kiraly (see [10]), that Vornberger's approach can be used to show that weighted-P_4 is NP-hard

in bipartite graphs. Therefore, with this paper, the complexities of the weighted and unweighted problems P_k in bipartite graphs are resolved. Finally, Cunningham and Wang [6] have studied, in general graphs, the polytopes given by the convex hull of incidence vectors of k-restricted 2-factors.

The paper is organized as follows. We state our min–max theorem in Section 2. In Section 3, we present our algorithm for finding maximum square-free 2-matchings in bipartite graphs. Proofs of the algorithm's validity and complexity, and the min–max theorem, are given in Section 4.

2. Min–max theorem

The main purpose of this section is to state a min–max theorem for the maximum size of a square-free 2-matching in a bipartite graph. As a corollary we obtain a theorem that characterizes those bipartite graphs that have a square-free 2-factor. This corollary is in the spirit of Tutte's theorem in [21] for classical matchings.

The necessity of the condition in the min–max theorem is straightforward to establish. However, the sufficiency appears to be considerably more difficult; a proof based on the algorithm is presented in Section 4.

For a graph $H = (V, E)$, let $q(H)$ denote the number of connected components of H that consist of either a single node, two nodes connected by an edge, or a square. For $S \subseteq V$, we let $H[S]$ denote the subgraph of H induced by S.

Theorem 1. *Let $G = (V, E)$ be a bipartite graph. Then the maximum size a square-free 2-matching in G is*

$$\min_{V' \subseteq V} \{|V| + |V'| - q(G[V \setminus V'])\}.$$

This theorem immediately implies the following corollary.

Corollary 2. *A bipartite graph $G = (V, E)$ has a square-free 2-factor if and only if for every subset $V' \subseteq V$,*

$$|V'| \geqslant q(G[V \setminus V']).$$

3. An algorithm for finding square-free 2-matchings in bipartite graphs

In this section we present our algorithm for finding square-free 2-matchings in bipartite graphs. We prove its validity and study its complexity in the next section. We begin with a few definitions.

Consider a simple bipartite graph $G = (A \cup B, F)$ with a 2-matching M. A node in G is called *saturated* if it is incident with two edges of M; otherwise, it is called *deficient*.

We next define a new operation, whose application appears to significantly simplify the statement of the algorithm. Let Q be a subgraph of G that is a square and contains exactly three edges in M. Suppose the nodes in Q are, in cyclic order, a, b, c, d, where ab is the edge not in M (see Fig. 1 for an example). Then the *square-shrinking of Q* is an operation that results in a new graph, say G', with a new 2-matching, say M'. To perform this operation, identify the nodes a and c to get the node a' and identify the nodes b and d to get the node b'; call this graph G''. Observe that G'' contains multiple edges; for example, G'' has four edges between a' and b', and if

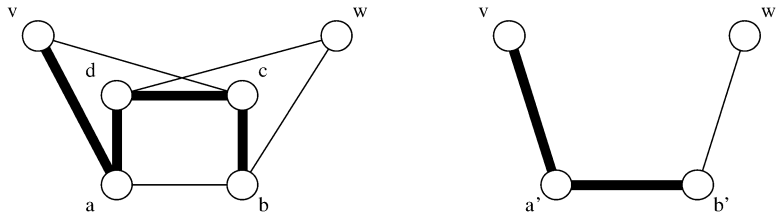


Fig. 1. The square-shrinking operation.

a node v in G (but not in Q) was adjacent to a and c, then there are two edges in G'' from v to a'. Let M'' denote the edges of G'' that were in M (M'' is not a 2-matching). G' is now obtained from G'' by replacing all multiple edges with single edges. Any edge in G' that was a multiple edge in G'', except $a'b'$, is called a *multi-edge* in G'. M' is obtained from M'' as follows: If xy is an edge in G' and if one or more of the edges from x to y in G'' is in M'', then we add xy to M'. Observe that M' is a 2-matching. We call the edge $a'b'$ in G' a *shrunk square*. The reverse operation of returning a shrunk square to its original state is called *expanding a shrunk square* and the resulting square is called *expanded*.

Note. In all figures, bold lines represent edges in the 2-matching and non-bold lines represent edges not in the 2-matching.

In the course of the algorithm for finding a maximum square-free 2-matching in G, a bipartite *surface graph* $\widetilde{G} = (\widetilde{A} \cup \widetilde{B}, \widetilde{F})$ is obtained from G by performing square-shrinkings on a set of pairwise node-disjoint squares of G (each with three edges in M). We let $\widetilde{A}$ denote the nodes of $\widetilde{G}$ that correspond to A in G. We define $\widetilde{B}$ similarly. (This correspondence is well defined by the definition of square shrinking.)

During the algorithm, we also construct a subgraph S of $\widetilde{G}$ called an *alternating forest* (hence S is the union of node disjoint trees). Each tree in S has a unique node called its *root*. Let v be a non-root node of the tree and let P be the unique path from v to a root. If P contains an even (odd) number of edges, then v and the edge of P that contains v are called *even* (*odd*). Roots are also called *even*. Let M recursively denote the original matching as it appears in $\widetilde{G}$ (this is what we do in the algorithm). We next describe some additional properties of S, with respect to M and the shrunk squares, that hold at the end of every step of the algorithm.

Properties of the alternating forest S:

(1) The roots are precisely the deficient nodes in $\widetilde{A}$.
(2) Every even edge is in M, and every odd edge is not in M.
(3) For every edge uv in M: If u is odd, then v is even.
(4) No shrunk square occurs in S.
(5) Precisely one endnode of each shrunk square is even.

From these definitions and properties, we can deduce the following:

(6) Odd nodes are incident with exactly one edge of S that is not in M. Odd nodes are incident with two edges of M (zero, one, or two of these edges can be in S); the other endnodes of these edges are even.

(7) Even nodes may be saturated or deficient. If an even node is saturated, then exactly one of these two edges of M must be in S; furthermore, the other endnode of this edge in S must be odd.
(8) Consider an arbitrary (non-trivial) path in S that contains a root as an endnode. Then the edges of the path are alternately in and not in M (and the edge incident with the root is not in M).
(9) No two shrunk squares are adjacent.

Algorithm *(Square-free 2-matchings in bipartite graphs).*

Input: A simple bipartite graph $G = (A \cup B, F)$.
Output: A maximum square-free 2-matching of G.
Step 0: Let M be any square-free 2-matching of G.
Step 1: If M is a 2-factor, done. Otherwise, let S consist of the set of deficient nodes of $\widetilde{G} := G$ in $\widetilde{A} := A$.
Step 2: If every edge of $\widetilde{G}$ that is incident with an even node is either an edge of M or has an odd node as its other endnode, then M (in G) is maximum; done. Otherwise, select an edge $vw \notin M$ where v is an even node and $w \notin S$.

Case 1: w is saturated and $\nexists$ a path of three edges (all non-shrunk) of M from v to w in $\widetilde{G}$. Go to Step 3.
Case 2: w is saturated and $\exists$ a path P of three edges (all non-shrunk) of M from v to w in $\widetilde{G}$. Go to Step 4.
Case 3: w is deficient and $\nexists$ a path of three edges (all non-shrunk) of M from v to w in $\widetilde{G}$. Go to Step 5.
Case 4: w is deficient and $\exists$ a path P of three edges (all non-shrunk) of M from v to w in $\widetilde{G}$. Go to Step 6.

Step 3: [Grow the forest.] Let wu_1 and wu_2 be the two edges of M incident with w. Since $\widetilde{G}$ is bipartite, each u_i is either even or $\notin S$. Add vw to S. If $u_i \notin S$, then add wu_i to S. Go to Step 2.
Step 4: Let P contain the edges wu_2, u_2u_3, and u_3v, which are in M and non-shrunk. For an example, see Fig. 2. Let wu_1 and wu_2 be the two edges of M incident with w. Since $\widetilde{G}$ is bipartite, each u_i is either even or $\notin S$. If u_1 is even, then go to Step 3 (this includes the case that wu_1 is shrunk). If the path from v to its root goes immediately to u_3, then go to Step 3. So we may assume that $u_1 \notin S$ (hence wu_1 is not shrunk, by property (5)) and that either v is deficient and a root, or the path from v to its root goes immediately through an edge $vu_4 \in M$ (where $u_4 \neq u_3$; see Fig. 2). Note that vu_4 is not shrunk (by property (4)).

Note that, since $\widetilde{G}$ is bipartite, u_2 is either even or $\notin S$. If u_2 is even, then u_3 must be odd (by property (7)). If $u_2 \notin S$, then $u_3 \notin S$. (If u_3 were odd, then u_2 would be even, by property (3); u_3 cannot be even, since $\widetilde{G}$ is bipartite.) In either case, shrink the square (with nodes v, w, u_2, u_3) and let $\widetilde{G}$ denote the new graph; let v' be the node resulting from the identification of u_2 and v; let w' be the node resulting from the identification of u_3 and w; and let M denote the new 2-matching.

We next update S. First, all nodes not in the square, and all edges not incident with a node in the square, that were in S before the shrinking, remain in S after the shrinking. If

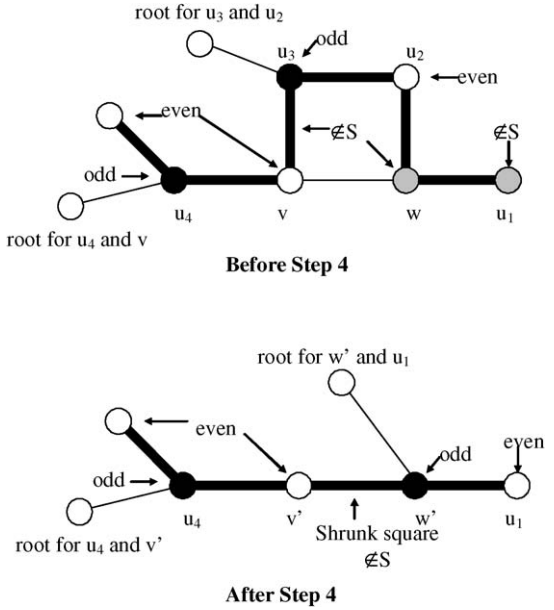


Fig. 2. Example for Step 4.

an edge of S was incident with v, u_3, or u_2 before the shrinking (and not in the square), then the corresponding edge after the shrinking is in S (now incident with v' or w'). If v was a root before the shrinking, then v' is a root after the shrinking.

Finally, if u_2 was even and u_3 was odd: Add $w'u_1$ to S. (Hence, $w'u_1$ becomes part of the tree of S that contained u_3, w' becomes odd, and u_1 becomes even.) (See Fig. 2.)

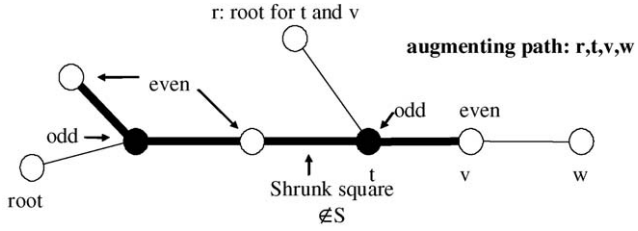
Go to Step 2.

Note 1. In Figs. 2–5, even nodes are white, odd nodes are black, and all other nodes are grey.

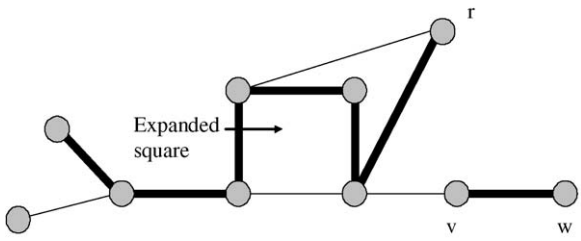
Step 5: [Augment the matching.] Consider the path consisting of the path from v to the root of its tree plus the edge vw (for an example, see Fig. 3). The edges along this path are alternately in and not in M; the endedges are not in M and the endnodes are deficient (see property (8)). Interchange edges in and out of the matching along this path to get a new 2-matching M. Expand the shrunk squares of $\widetilde{G}$. If a multi-edge was in M before expanding, then, after expanding, arbitrarily put one of the resulting two edges into M; otherwise put neither resulting edge into M. Rematch each resulting square with three edges to obtain a square-free 2-matching, again called M. "Throw away" S and go to Step 1.

Step 6: Let P contain the edges wu_1, u_1u_2, and vu_2. If the path from v to its root goes immediately to u_2, then go to Step 5.

So let us assume that v is deficient and a root, or the path from v to its root goes immediately through a matching edge, say vu_3, where $u_3 \neq u_2$ (hence vu_3 is not shrunk, by property (4)). (For an example, see Fig. 4.)

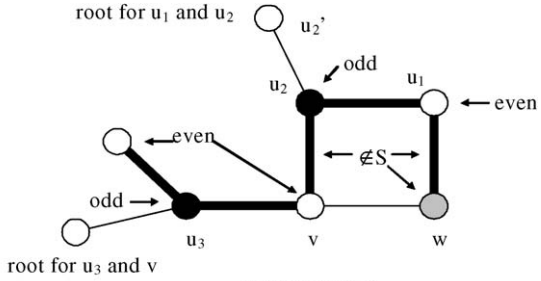


Before Step 5

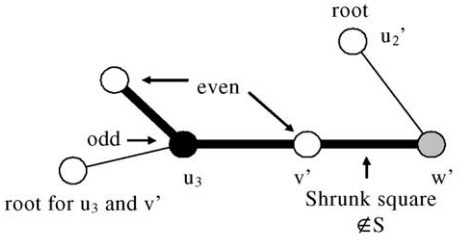


After Step 5

Fig. 3. Example for Step 5.



Before Step 6



After Step 6

Fig. 4. Example for Step 6.

Shrink the square (with nodes v, w, u_1, u_2) and let $\widetilde{G}$ denote the new graph; let v' be the node resulting from the identification of u_1 and v; let w' be the node resulting from the identification of u_2 and w; and let M denote the new 2-matching.

We next update S. First, all nodes not in the square, and all edges not incident with a node in the square, that were in S before the shrinking, remain in S after the shrinking. If an edge of S was incident with v, w, u_1, or u_2 before the shrinking (and not in the square), then the corresponding edge after the shrinking is in S (now incident with v' or w'). If v was a root before the shrinking, then v' is a root after the shrinking.

Note that, before the shrinking, u_1 was either even or $\notin S$, since $\widetilde{G}$ is bipartite; furthermore, u_1 was even if and only if u_2 was odd (by properties (3) and (7)). Hence, we do the following:

If u_1 was even and u_2 was odd: Let u_2u_2' be the unique non-matching edge in S that was incident with u_2 (see property (6)). Then u_2' is even. Remove $w'u_2'$ from S and go to Step 5 with u_2' and w' playing the roles of v and w, respectively.

Otherwise, go to Step 2.

4. Proofs

We begin this section by showing, in Proposition 3, that the algorithm outputs a square-free 2-matching (i.e., that augmentations in Step 5 can be implemented). We then show that the algorithm has polynomial-time worst case time complexity. After this, we prove two key propositions (6 and 7), which elaborate on the properties of the alternating forest. Finally, we prove that the square-free 2-matching output by the algorithm is maximum. This proof is similar in style to Edmonds' proof of the validity of the matching algorithm. We simultaneously prove the min–max theorem.

Proposition 3. *Let G be a bipartite graph and let M be a square-free 2-matching in G, which is not a 2-factor. Input G and M into the algorithm. Then we have the following*:

(1) *before making a pass through Step 5, the alternating forest S that is constructed by the algorithm satisfies properties* (1)–(9);
(2) *if the algorithm enters Step 5, then the expanded squares can be rematched so that the resulting set M' is a square-free 2-matching.*

Proof. Part (1) follows immediately from the design of the algorithm, hence we focus on the proof of part (2). We consider the first pass of the algorithm through Step 5. We state and then prove three claims from which the result follows.

Claim 1. *In Step 5 the expanded squares can be rematched, each with three edges, so that the resulting subgraph M' is a 2-matching.*

Claim 2. *Consider any rematching M' of the expanded squares as described in Claim 1. Then, no edge in an expanded square can be contained in a square of M'.*

Claim 3. *Consider any rematching M' of the expanded squares as described in Claim 1. Then, M' is square-free.*

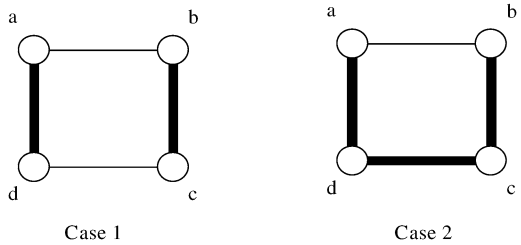


Fig. 5. Proof of Claim 3.

Proof of Claim 1. Consider the point in Step 5 after interchanging edges along the path, but before expanding the shrunk squares. Let M'' denote the matching at this point. Because the edges in the path alternated in and out of the matching and because the endnodes of the path were deficient, M'' is a 2-matching. Also, because shrunk squares do not occur in S, M'' contains all of the shrunk squares. Let $a'b'$ be an arbitrary shrunk square. Then a' is incident with one or two edges of M'' (similarly for b'). Let M' equal M'' minus the edge $a'b'$. Expand $a'b'$ to get a square with nodes a, b, c, d, where a and c were identified to give a'; b and d were identified to give b'; and ab was the edge not in M. Let G' denote the graph so obtained from $\widetilde{G}$. (Note that this expansion is well defined since no two shrunk squares are adjacent in $\widetilde{G}$, by property (9).) Consider any multi-edges that are split into two edges by this unshrinking. If such a multi-edge was in M, then arbitrarily put one of the resulting two edges into M'; otherwise put neither resulting edge into M'. If a' and b' were each incident with only one edge in M (i.e., $a'b'$ itself), then we may put any three edges of the square into M' to obtain a 2-matching of G' (let us arbitrarily choose to replace the original three edges of M). Suppose a' was incident with two edges of M. Then we may assume, without loss of generality, that there is an edge ae of G' in M' (where e is not in the square). If b' was incident with one edge of M, then we may add the edges ab, bc, cd, or the edges ad, cd, bc to M' to obtain a 2-matching of G' (let us arbitrarily choose to add those edges to M'' so that both b and d are incident with the same number of edges of M' as they were in G). Finally, if b' was incident with two edges of M, then there is a unique set of three edges in the square that we can add to M' to obtain a 2-matching of G'. We can recursively apply this same approach to any remaining shrunk squares in G'; the result follows. □

Proof of Claim 2. Consider an expanded square in G. By Claim 1, it contains precisely three edges of M'. Hence the only square of M that could contain one of these edges is the expanded square itself. The result follows. □

Proof of Claim 3. Consider a 2-matching M' of G as described in Claim 1. By Claim 2, if M' contains a square, then none of its edges is contained in an expanded square. Hence it was created in Step 5 immediately after the interchange along the path, call it P. In particular, the square, before the interchange, contained from zero to three edges in M and none of these edges was shrunk. In fact, because P is simple, no two adjacent edges of this square could be out of M before the interchange; thus the square must look like one of the two cases in Fig. 5.

Case 1: (See Fig. 5.) Every edge in P has the property that one endnode is even and the other is odd, except for vw, since w is not in S. Thus, at most one of the nodes a, b, c, d is not in S.

Case 1a: Assume a, b, c, d are all in S. Let us assume, without loss of generality, that a and c are even, and b and d are odd. Finally, let us assume that ab occurs closer to the root of this tree than cd. Then when the tree was being grown, the edge ab was considered in Step 2 before cd.

Thus, the edge bc was added to the tree when ab was considered. But this implies that ab, bc, cd are in P, which contradicts $abcd$ becoming a square when augmenting.

Case 1b: Assume $b = w$. Then $a = v$, which implies that a and c are even and d is odd (by bipartiteness). Also, cd must be in the same tree as a. Thus, when this tree was being grown, cd was considered before a became even. But then da must have been added to the tree when cd was considered, which contradicts $abcd$ becoming a square when augmenting.

Case 2: (See Fig. 5.) This case cannot occur since we would have shrunk the square during the algorithm.

This completes the proof of Claim 3 and the proposition. □

Proof that the algorithm has polynomial-time worst case time complexity. By Proposition 3, the algorithm maintains a 2-matching M throughout. Hence, the algorithm makes $O(|V|)$ passes through Step 1 since each such pass (except the first) in preceded by a pass through Step 5 where the number of edges in M is increased by 1. (Note that the worst case complexity of Step 5 is $O(|V|)$, which is dominated by the remaining work in the algorithm.) In between passes through Step 1 the algorithm passes repeatedly through Step 2. Each pass through Step 2 requires looking at $O(|V|^2)$ edges. Each such pass is preceded by a pass through Steps 3, 4, or 6 (except the first pass in each iteration of Step 1). Each pass through Steps 3, 4, or 6 requires only constant time. Each pass through Step 4 or 6 results in a square-shrinking. From the cases of the algorithm we know that no shrunk square is contained in a square that is later shrunk and we know from Proposition 3 that no two shrunk squares are adjacent. Thus, the number of times, between passes through Step 1, that the algorithm shrinks a square is $O(|V|)$. With every pass through Step 3 the forest S is enlarged by one edge. This also can happen only $O(|V|)$ times. Thus, the amount of work between passes through Step 1 is $O(|V|^3)$. Hence, the worst case time complexity of the algorithm is $O(|V|^4)$. Note that by keeping track of the edges that have been looked at in Step 2, the time required in this step can be reduced to $O(|V|)$, which reduces the overall complexity to $O(|V|^3)$. □

For the remainder of our proofs we make use of the following set up and terminology. Suppose we run the algorithm on a bipartite graph G and that the algorithm stops in Step 2 (hence a 2-factor has not been found). Let us run the algorithm one more time where, in Step 1, we let S be the alternating forest at the end of the first run together with the deficient nodes in $\widetilde{B}$ (which become roots). It is easy to see that no augmentations occur in Step 5 since this would imply that S was not completely grown in the previous run. It is easy to see that the final forest S still satisfies properties (1)–(9), except properties (1) and (5) become the following, respectively:

(1′) The roots are precisely the deficient nodes in $\widetilde{A} \cup \widetilde{B}$.
(5′) At least one endnode of each shrunk square is even.

Let $\widetilde{G} = (\widetilde{A} \cup \widetilde{B}, \widetilde{F})$ denote the bipartite surface graph at the end of the algorithm. Let $\widetilde{O}$ denote the set of odd nodes and let $\widetilde{E}$ denote the set of even nodes. Let $\widetilde{R}$ denote the nodes that are neither odd nor even. Let $\widetilde{R}'$ denote those nodes in $\widetilde{R}$ that are adjacent to two even nodes via edges in M. Let $\widetilde{A}' = \widetilde{A} \cap \widetilde{R}$. (Note that $\widetilde{A}' \cap \widetilde{R}'$ may or may not be empty.) Let

$$\widetilde{G}' = \widetilde{G}\big[\{\widetilde{A} \cup \widetilde{B}\} \setminus \{\widetilde{O} \cup \widetilde{R}' \cup \widetilde{A}'\}\big].$$

That is, $\widetilde{G}'$ is the graph obtained from $\widetilde{G}$ by deleting the nodes in $\{\widetilde{O} \cup \widetilde{R}' \cup \widetilde{A}'\}$. In $\widetilde{G}'$ we let M denote the edges that were contained in M of $\widetilde{G}$.

Observe that every node in G has a corresponding node in $\widetilde{G}$. In particular, if x and y are two nodes in G that were identified to give a node w in $\widetilde{G}$, then x and y correspond to w. Let O, R', and A' denote the sets of nodes in G that correspond to the sets $\widetilde{O}$, $\widetilde{R}'$, and $\widetilde{A}'$, respectively, in $\widetilde{G}$. We let M in G denote the final square-free 2-matching output by the algorithm. We let

$$G' = G\big[\{A \cup B\} \setminus \{O \cup R' \cup A'\}\big].$$

Proposition 4. *Each component of $\widetilde{G}'$ is an isolated node or isolated edge. Furthermore, each isolated edge of $\widetilde{G}'$ is in M.*

Proposition 5. *Every node of $\widetilde{G}$ in $\{\widetilde{O} \cup \widetilde{R}' \cup \widetilde{A}'\}$ is adjacent to two nodes in $\{\widetilde{A} \cup \widetilde{B}\} \setminus \{\widetilde{O} \cup \widetilde{R}' \cup \widetilde{A}'\}$ via edges in M.*

Proposition 6. *Each component of G' is an isolated node, isolated edge, or isolated square. Furthermore, each isolated edge of G' is in M and each isolated square of G' contains three edges in M.*

Proposition 7. *Every node of G in $\{O \cup R' \cup A'\}$ is adjacent to two nodes in $\{A \cup B\} \setminus \{O \cup R' \cup A'\}$ via edges in M.*

Proof of Proposition 4. Note that the nodes in $\widetilde{G}'$ were either even nodes of $\widetilde{G}$ or nodes in the set $\widetilde{R} \setminus \{\widetilde{A} \cup \widetilde{R}'\}$. Due to the stopping criterion of the algorithm (in Step 2), every edge incident with an even node of $\widetilde{G}'$ must be in M. Because the nodes in $\widetilde{R} \setminus \{\widetilde{A} \cup \widetilde{R}'\}$ are all contained in $\widetilde{B}$ in $\widetilde{G}$, there are no edges in $\widetilde{G}'$ between any pair of nodes in $\widetilde{R} \setminus \{\widetilde{A} \cup \widetilde{R}'\}$. Thus all the edges in $\widetilde{G}'$ are in M. Furthermore, every node in $\widetilde{R} \setminus \{\widetilde{A} \cup \widetilde{R}'\}$ must have degree 0 or 1 in $\widetilde{G}'$ since the only nodes in $\widetilde{R}$ that were adjacent to two even nodes via edges in M were in $\widetilde{R}'$.

Consider an edge ab in $\widetilde{G}'$ where a is even and $b \in \widetilde{R} \setminus \{\widetilde{A} \cup \widetilde{R}'\}$. By property (7) of the alternating structure, if a was adjacent in $\widetilde{G}$ to a second node, say c, via an edge in M, then c must have been odd in $\widetilde{G}$. Thus ab must be an isolated edge in $\widetilde{G}'$. Finally, consider an edge ab in $\widetilde{G}'$, where a and b are both even. Again by property (7), if a (respectively b) was adjacent in $\widetilde{G}$ to a second node, say c, via an edge in M, then c must have been odd in $\widetilde{G}$. Thus, again, ab must be an isolated edge in $\widetilde{G}'$. □

Proof of Proposition 5. The property clearly holds for nodes in $\widetilde{O}$ by property (6) of the alternating structure and it holds for nodes in $\widetilde{R}'$ by definition. So, let v be a node in $\widetilde{A}' \setminus \widetilde{R}'$. v is incident with two edges in M since otherwise v would be in $\widetilde{E}$. No such edge can be incident with a node in $\widetilde{O}$ or $\widetilde{R}'$ since all edges in M that are incident with an edge in $\widetilde{O}$ or $\widetilde{R}'$ are also incident with a node in $\widetilde{E}$. Since $\widetilde{G}$ is bipartite, and since $\widetilde{A}' \subseteq \widetilde{A}$, no such edge can be incident with a second node in $\widetilde{A}'$. The result follows. □

Proof of Proposition 6. This follows immediately from Proposition 4 where any isolated edge in $\widetilde{G}'$ that was a shrunk square in $\widetilde{G}$ corresponds to an isolated square in G' with three edges in M. □

Proof of Proposition 7. This follows immediately from Proposition 5 and the structure of the squares that are shrunk during the algorithm. □

Proof of validity of the algorithm and Theorem 1. We first show that for any square-free 2-matching M in G and any $V' \subseteq V$,

$$|M| \leqslant |V| + |V'| - q\big(G[V \setminus V']\big). \tag{1}$$

We then show that the matching M produced at the end of the algorithm satisfies this inequality at equality for a special V'. Hence the algorithm produces a maximum matching and the min–max theorem holds.

Consider an arbitrary square-free 2-matching M in the graph G. The *deficiency* of a node in G is defined to be 2 minus the number of edges of M that are incident with the node. The *deficiency* of M is defined to be the sum of the deficiencies at the nodes and is denoted $df_G(M)$. Observe that $|M| = |V| - \frac{1}{2}df_G(M)$, hence inequality (1) is equivalent to

$$df_G(M) \geqslant 2q\big(G[V \setminus V']\big) - 2|V'|. \tag{2}$$

To see that this inequality is true, observe that each component counted in $q(G[V \setminus V'])$ has a deficiency of at least 2 (when M is restricted to it), that these deficiencies can be satisfied only by edges of M with one endnode in V', and that there are at most $2|V'|$ such edges.

By Proposition 3, the algorithm outputs a square-free 2-matching M. If M is a 2-factor, then clearly the algorithm has worked. Furthermore, if we take $V' = \emptyset$, then inequality (2) is satisfied at equality. So let us assume that M is not a 2-factor. Let O, R', A', and G' be defined as above and set $V' = \{O \cup R' \cup A'\}$. By Proposition 6 we know that each component of $G' = G[V \setminus V']$ is an isolated node, isolated edge (in M), or an isolated square (with three edges in M); and, hence, each component of G' has deficiency 2. By Proposition 7 we know that every node in V' is incident with two edges of M whose other endnodes are in $V \setminus V'$. Hence inequality (2) is satisfied at equality, which finishes the proof. □

Remark 1. A well-known theorem due to Petersen [18] and Berge [1] characterizes maximum matchings in terms of augmenting paths. Using the algorithm in this paper one can prove a similar-style theorem for square-free 2-matchings (see [13]). A theorem of this type for square-free 2-matchings in general graphs appears in [19].

Acknowledgments

I thank Gérard Cornuéjols for several helpful discussions of this work. I thank Zoltan Kiraly for pointing out an error in an earlier statement (in [13]) of Theorem 1 and for suggesting the more elegant statement that appears in the paper. I also thank the anonymous referees for their numerous, thoughtful suggestions for improving the paper.

References

[1] C. Berge, Two theorems in graph theory, Proc. Natl. Acad. Sci. USA 43 (1957) 842–844.
[2] C. Berge, Sur le couplage maximum d'un graphe, C. R. Hebdomadaires Séances Acad. Sci. Paris 247 (1958) 258–259.
[3] G. Cornuéjols, W.R. Pulleyblank, A matching problem with side conditions, Discrete Math. 29 (1980) 135–159.
[4] W.H. Cunningham, Matching, matroids, and extensions, Math. Program. Ser. B 91 (2002) 515–542.
[5] W.H. Cunningham, J. Geelen, personal communication, 1998.
[6] W.H. Cunningham, Y. Wang, Restricted 2-factor polytopes, Math. Program. 87 (1) (2000) 87–111.
[7] J. Edmonds, Maximum matching and a polyhedron with 0,1 vertices, J. Res. Natl. Bur. Standards Sect. B 69 (1965) 73–77.

[8] J. Edmonds, Paths, trees, and flowers, Canad. J. Math. 17 (1965) 449–467.
[9] M.L. Fisher, G.L. Nemhauser, L.A. Wolsey, An analysis of approximations for finding a maximum weight Hamiltonian circuit, Oper. Res. 27 (1979) 799–809.
[10] A. Frank, Restricted t-matchings in bipartite graphs, Discrete Appl. Math. 131 (2003) 337–346.
[11] J. Geelen, personal communication, 2000.
[12] D. Hartvigsen, Extensions of matching theory, PhD thesis, Carnegie–Mellon University, 1984, under the supervision of Gérard Cornuéjols.
[13] D. Hartvigsen, The square-free 2-factor problem in bipartite graphs, in: G. Cornuéjols, R. Burkard, G.J. Woeginger (Eds.), Integer Programming and Combinatorial Optimization, in: Lecture Notes in Comput. Sci., vol. 1610, Springer, Berlin, 1999, pp. 234–240 (extended abstract).
[14] P. Hell, D.G. Kirkpatrick, J. Kratochvil, I. Kriz, On restricted two-factors, SIAM J. Discrete Math. 4 (1988) 472–484.
[15] Z. Kiraly, Square-free 2-matchings in bipartite graphs, Working paper, 1999.
[16] M.S. Krishnamoorty, An NP-hard problem in bipartite graphs, SIGACT News 7 (1) (1975) 26.
[17] Y. Nam, Matching theory: Subgraphs with degree constraints and other properties, PhD thesis, University of British Columbia, 1994, under the supervision of R. Anstee.
[18] J. Petersen, Die Theorie der regulären Graphen, Acta Math. 15 (1891) 193–220.
[19] M. Russell, Restricted 2-factors, Master's thesis, University of Waterloo, 2001, under the supervision of W. Cunningham.
[20] A. Schrijver, Combinatorial Optimization, Polyhedra and Efficiency, Springer, Berlin, 2003.
[21] W.T. Tutte, The factorization of linear graphs, J. London Math. Soc. 22 (1947) 107–111.
[22] W.T. Tutte, The factors of graphs, Canad. J. Math. 4 (1952) 314–328.
[23] W.T. Tutte, A short proof of the factor theorem for finite graphs, Canad. J. Math. 6 (1954) 347–352.
[24] O. Vornberger, Easy and hard cycle covers, preprint, Universitat Paderborn, 1980.