Choosability of Bipartite Graphs

Sheila Ferneyhough
and
Gary MacGillivray*
Department of Mathematics and Statistics
University of Victoria
Victoria, Canada V8W 3P4

Abstract

A graph G is called $(a : b)$-choosable if for every assignment of a-sets $L(v)$ to the vertices of G it is possible to choose b-subsets $M(v) \subseteq L(v)$ so that adjacent vertices get disjoint subsets. We give a different proof of a theorem of Tuza and Voigt that every 2-choosable graph is $(2k : k)$-choosable for any positive integer k. Our proof is algorithmic and can be implemented to run in time $O(k|V(G)|)$.

1 Introduction

The authors of this paper are a part of Professor E.J.Cockayne's mathematical family. The present work is contained in the first author's Master's thesis [2], for which Professor Cockayne was a member of the supervisory committee, with the second author as Supervisor. The second author was Professor Cockayne's first graduate student, and is now his colleague at the University of Victoria. We are very pleased to dedicate this paper to him on the occasion of his sixtieth birthday. Two of our favourite quotes, one by Ernie Cockayne and the other about him, follow.

You must be both idle and stupid to fail this course, one or the other will not suffice. –E.J.Cockayne

Professor Cockayne steadfastly refused to scale the grades, thereby depriving me of any chance of passing the course. –from one of Professor Cockayne's teaching evaluations.

*Research supported by NSERC.

The concept of vertex colouring can be generalized by imposing restrictions on the colours allowable for each vertex. A graph G is *k-choosable (k-list-colourable)* if for every assignment of k-sets $L(v)$ to the vertices of G, there is a proper vertex colouring m of $V(G)$ where $m(v) \in L(v)$ for each $v \in V(G)$. The *choice number (list-chromatic number)* $\chi_\ell(G)$ is the minimum integer k such that G is k-choosable.

The notion of choosability can be generalized further as follows. Let a and b be integers such that $a > b \geq 1$. A graph G is *$(a : b)$-choosable* [1] if for every assignment of a-sets $L(v)$ to the vertices of G, there exist b-subsets $M(v) \subset L(v)$ for every $v \in V(G)$ such that $M(u) \cap M(v) = \emptyset$ whenever u and v are adjacent in G. Note that a graph which is $(a : 1)$-choosable is a-choosable. For a graph G with a fixed assignment of a-sets, such a choice of b-subsets is called an *$(a : b)$-choice.*

Erdös, Rubin and Taylor [1] conjectured that every $(a : b)$-choosable graph is $(ak : bk)$-choosable, for positive integer k. Gutner [3] obtained the first partial result by proving that any 2-choosable graph is $(4 : 2)$-choosable. Gutner and Tarsi [4] proved that any even cycle is $(2k : k)$-choosable. Finally, Tuza and Voigt proved that any 2-choosable graph is $(2k : k)$-choosable. Their proof is algorithmic and leads to an $O(k^3|V|)$ algorithm for determining a $(2k : k)$-choice. We offer a different proof of the theorem of Tuza and Voigt. Our proof is also algorithmic, and leads to a $O(k^2|V|)$ algorithm for determining a $(2k : k)$-choice.

2 Preliminaries

Our proof relies on the following results of Erdös, Rubin and Taylor [1], and Lemma 2.3 below. The latter is a generalization of Lemma 2 in [5].

We first define two crucial concepts, and then state results relating them. The *core* of a graph G, denoted by core(G), is obtained by successively removing vertices of degree 1 until none remain. The *theta* graph, $\theta_{a,b,c}$ is the graph consisting of three paths of lengths a, b and c respectively, which share a pair of end vertices, but are otherwise vertex disjoint.

Theorem 2.1 *[1] For all $m \geq 1$, $\chi_\ell(\theta_{2,2,2m}) = 2$.*

As $C_{2m+2} \subset \theta_{2,2,2m}$, it follows that $\chi_\ell(C_{2m+2}) = 2$ for all $m \geq 1$ [1]. The next theorem characterizes the 2-choosable graphs.

Theorem 2.2 *[1] A graph G is 2-choosable if and only if core$(G) \in$ $\{K_1, C_{2m+2},$ $\theta_{2,2,2m} : m \geq 1\}$.*

The proof of the following Lemma is straightforward, and left to the reader.

Lemma 2.3 *If a and b are positive integers such that $\frac{a}{b} \geq 2$, and G is a graph such that core(G) is $(a:b)$-choosable, then G is $(a:b)$-choosable.*

3 2-Choosable Graphs

This section presents algorithms which determine a $(2k:k)$-choice for the core of any 2-choosable graph, thereby proving that any 2-choosable graph is $(2k:k)$-choosable.

Each of Algorithms 2 through 5 which follow acts on a particular class of 2-choosable graphs. If $2k$-sets have been assigned to the vertices of the input graph, then the appropriate algorithm can be used to output a $(2k:k)$-choice for that graph.

In Algorithms 2 through 5, an iteration involves removing edges of the input graph to obtain a spanning tree T with root r, choosing an element $m(r) \in L(r)$, and calling Algorithm 1, below. Algorithm 1 performs a breadth-first search on T, choosing and deleting an element from colour set $L(v)$ when reaching vertex v. If $d(r,v)$ is even, the chosen element is added to its choice set $M(v)$. The data structures for Algorithm 1 include R, which is a queue of vertices, and S, which is a set of vertices of T.

Algorithm 1

Input: tree T, root r, element $m(r)$, sets $M(v)$ and $L(v)$ for each $v \in V(T)$
Initialization: $S := \emptyset$, $i := 0$

BEGIN
  $R :=$ queue_insert$(r,$ empty_queue$)$;
  $M(r) := M(r) \cup m(r)$; (1.1)
  $L(r) := L(r) - m(r)$;
  WHILE not_empty(R) DO BEGIN
    $i := i + 1$;
    $v :=$ queue_remove(R);
    FOR each neighbour u of v DO
      IF u is not in R or S THEN BEGIN
        $R :=$ queue_insert(u,R); (1.2)
        IF $m(v) \in L(u)$ THEN
          $m(u) := m(v)$ (1.3)
        ELSE
          Choose $m(u) \in L(u) - L(v)$; (1.4)
        $L(u) := L(u) - m(u)$; (1.5)
        IF i is even THEN
          $M(u) := M(u) \cup m(u)$ (1.6)

END;
$S := S \cup \{v\}$
END
END.

Complexity: $O(k|V(T)|)$

That the time complexity of this algorithm is $O(k|V(T)|)$ is clear from its breadth-first search nature, in which a fixed set of operations (set union, set difference, membership testing, and queue operations) requiring time $O(k)$ is performed for each vertex of T.

For the proofs which follow, note that after each call to Algorithm 1, the cardinality of each colour set $L(v)$ has decreased by one, and for each vertex v such that $d(r, v)$ is even, the cardinality of $M(v)$ has increased by one.

Lemma 3.1 *Suppose there is a call to Algorithm 1 where $m(v) = x$ is added to $M(v)$. Then for any vertex u which is adjacent to v in T, x will not be in $L(u)$ on completion of the call, and x will not be added to $M(u)$ during the call.*

Proof. If $v = r$, then $d(r, v)$ is even. Otherwise, $m(v) = x$ and x is added to $M(v)$ at line (1.6), so $d(r, v)$ is even.

Suppose that $d(r, v) = d(r, u) - 1$. Then $m(v)$ is chosen before $m(u)$. If $m(v) = x \in L(u)$, then $m(u) = x$ at line (1.3). Now x is deleted from $L(u)$ at line (1.5), and as $d(r, u)$ is odd, u is not added to $M(u)$ at line (1.6).

Suppose that $d(r, v) - 1 = d(r, u)$. Then $m(u)$ is chosen before $m(v)$. Suppose that $m(u) := y$. Then y is deleted from $L(u)$ at line (1.5) and y is not added to $M(u)$ at line (1.6). Now, $m(v) := x$ is chosen either at line (1.3) or line (1.4). If chosen at line (1.3), then $m(v) = m(u)$, so $y = x$. If chosen at line (1.4), then $y \neq x$ and $m(v) \in L(v) - L(u)$, so $x \notin L(u)$. ∎

Given an assignment of $2k$-sets to the vertices of a tree T, Algorithm 2 determines a $(2k : k)$-choice for T. The initialization includes choosing an arbitrary root $r \in V(T)$. Algorithm 1 is called k times by Algorithm 2, resulting in a choice of k elements for each vertex at an even distance from r. For each vertex at an odd distance from r, the remaining k elements in its colour set provide its choice set.

Algorithm 2

Input: k, T, sets $L(v)$ for each $v \in V(T)$
Initialization: Arbitrarily choose a root vertex, $r \in V(T)$, $i := 0$, $M(v) := \emptyset$ for each $v \in V(T)$

BEGIN
FOR i:= 1 TO k DO BEGIN
Choose $m(r) \in L(r)$; (2.1)
Call Algorithm 1
END;
FOR all $v \in V(T)$ such that $d(r, v)$ is odd DO
$M(v) := L(v)$ (2.2)
END.

Output: sets $M(v)$ for each $v \in V(T)$

Complexity: $O(k^2|V(T)|)$

Theorem 3.2 *Any tree is* $(2k : k)$*-choosable for all* $k \geq 1$.

Proof. It suffices to show that the sets $M(v)$ for each $v \in V(T)$ are a $(2k : k)$-choice for T. As the root vertex r is fixed during Algorithm 2, there is a fixed set of vertices at an even distance from r, and a fixed set of vertices at an odd distance from r. So if v is a vertex such that $d(r, v)$ is odd, elements of $L(v)$ are added to $M(v)$ only at line (2.2). After Algorithm 1 is called k times, $|L(v)| = k$ for each v, and $|M(v)| = k$ for each v such that $d(r, v)$ is even. By line (2.2), $|M(v)| := |L(v)| = k$ for all v such that $d(r, v)$ is odd. Therefore $M(v) \subset L(v)$ and $|M(v)| = k$ for each $v \in V(T)$.

There is a fixed set of vertices at an odd distance from r, so if v is a vertex such that $d(r, v)$ is odd, elements of $L(v)$ are added to $M(v)$ only at line (2.2). Thus if $x \notin L(v)$ at any stage of Algorithm 2, then x is not in $M(v)$ on completion of the algorithm. By Lemma 3.1, if x is added to $M(v)$ during a call to Algorithm 1, then for any vertex u which is adjacent to v, x will not be in $L(u)$ on completion of the call, and x will not be added to $M(u)$ during the call. Therefore for each $x \in M(v)$, where $d(r, v)$ is even, $x \notin M(u)$ for any u adjacent to v.

Thus the sets $M(v)$ are a $(2k : k)$-choice for T. ∎

For some $m \geq 2$, let $C_{2m} = v_0, v_1, \ldots, v_{2m-1}, v_0$. Given an assignment of $2k$-sets to the vertices of C_{2m}, Algorithm 3 determines a $(2k : k)$-choice for the cycle. The algorithm searches for a pair u, v of adjacent vertices which have distinct colour sets. If such a pair exists, then $T := C_{2m} - (u, v)$ and Algorithm 1 is called. This search is iterated $i \leq k$ times, while such a pair of vertices exists, resulting in a choice set of i elements for each even-indexed vertex. If $i = k$, then for each vertex at an odd distance from r, the remaining k elements in its colour set provide its choice set. If $i < k$, then for each vertex at an even distance from r, an additional $k - i$ elements for its choice set are chosen from $L(v_0)$. For each vertex at an odd distance from r, the remaining $2k - i$ elements of its colour set, excluding the $k - i$ elements chosen above, provide a choice set of k elements.

Algorithm 3

Input: k, sets $L(v)$ for each $v \in V(C_{2m})$
Initialization: $i := 0$, $M(v) := \emptyset$ for each $v \in V(C_{2m})$

BEGIN
  WHILE [$i < k$] AND there exists an even j, $0 \le j < 2m - 1$ such that either $L(v_j) \neq L(v_{j+1})$ OR $L(v_j) \neq L(v_{j-1})$ DO BEGIN
    IF there exists an even j such that $L(v_j) \neq L(v_{j+1})$ DO BEGIN
      $T := C_{2m} - (v_j, v_{j+1})$;
      $r := v_j$;
      $m(r) := x$ where $x \in L(v_j) - L(v_{j+1})$ (3.1)
    END
    ELSE IF there exists an even j such that $L(v_j) \neq L(v_{j-1})$ DO BEGIN
      $T := C_{2m} - (v_j, v_{j-1})$;
      $r := v_j$;
      $m(r) := x$ where $x \in L(v_j) - L(v_{j-1})$ (3.2)
    END;
    Call Algorithm 1; (3.3)
    $i := i + 1$
  END;
  IF $i = k$ THEN
    $M(v_j) := L(v_j)$ for each odd j (3.4)
  ELSE BEGIN
    Choose $M'(v_0) \subset L(v_0)$ where $|M'(v_0)| = k - i$; (3.5)
    For each even j, $M(v_j) := M(v_j) - M'(v_0)$; (3.6)
    For each odd j, $M(v_j) := L(v_j) - M'(v_0)$ (3.7)
  END
END.

Output: sets $M(v)$ for each $v \in V(C_{2m})$

Complexity: $O(mk^2)$

Algorithm 3 is iterated at most k times, where each iteration consists of a fixed set of operations, including at most one call to Algorithm 1. As the searches take time $O(|V(T)|) = O(m)$, the other operations take time $O(k)$, and Algorithm 1 always acts on a tree T where $|V(T)| = 2m$, the time complexity of Algorithm 3 is $O(mk^2)$.

Theorem 3.3 *For all $k \ge 1$, C_{2m} is $(2k : k)$-choosable.*

Proof. It suffices to show that the sets $M(v)$ for each $v \in V(C_{2m})$ are a $(2k : k)$-choice for C_{2m}. For any call to Algorithm 1, $r = v_j$ where j is even, so the vertices at an even distance from r are v_j such that j is even,

and those at an odd distance from r are v_j such that j is odd. Suppose that $v_j \in V(C_{2m})$ where j is odd. Elements of $L(v_j)$ are added to $M(v_j)$ only at line (3.4) or (3.7).

Suppose that Algorithm 3 calls Algorithm 1 exactly i times. Then $|L(v_j)| = 2k - i$ for all v_j, and $|M(v_j)| = i$ for all even j. If $i = k$, then $|L(v_j)| = k$ for each v_j, and $|M(v_j)| = k$ for each even j. By line (3.4), $|M(v_j)| := |L(v_j)| = k$ for each odd j. Thus $M(v_j) \subset L(v_j)$ and $|M(v_j)| = k$ for each $v_j \in V(C_{2m})$.

If $i < k$, then for all even j, $L(v_j) = L(v_{j+1})$ and $L(v_j) = L(v_{j-1})$. Thus all vertices have the same set. At line (3.6), $|M(v_j)| := |M(v_j) \cup M'(v_0)| =$ i + (k - i) = k for each even j. As $M'(v_0) \subset L(v_0) = L(v_j)$, $M(v_j) \subset L(v_j)$ for each even j. At line (3.7), $|M(v_j)| := |L(v_1) - M'(v_0)| = 2k - i - (k - i) = k$ for each odd j. As $L(v_1) - M'(v_0) \subseteq L(v_1) = L(v_j)$, $M(v_j) \subset L(v_j)$ for all odd j.

The vertices at an odd distance from r are v_j where j is odd, so if $v_j \in V(C_{2m})$ and j is odd, then elements of $L(v_j)$ are added to $M(v_j)$ only at line (3.4) or (3.7). Thus if at any stage of Algorithm 3, $x \notin L(v_j)$, then x is not in $M(v_j)$ on completion of the algorithm. Suppose that $m(v) = x$ is added to $M(v)$ during a call to Algorithm 1 which acts on T. Then by Lemma 3.1, for any u which is adjacent to v in T, x will not be in $M(u)$ on completion of Algorithm 3.

Consider a call to Algorithm 1 where $T = C_{2m} - (v_j, v_{j+1})$. and j is even. Then by line (3.1), $x \in L(v_j) - L(v_{j+1})$. At line (1.1). x will be added to $M(v_j)$. As $x \notin L(v_{j+1})$ and $j + 1$ is odd, x will not be in $M(v_{j+1})$ on completion of Algorithm 3. Similarly if a call is made when $T = C_{2m} - (v_j, v_{j-1})$.

Suppose $x \in M(v_j)$ was chosen at line (3.5). Then at line (3.6), $M(v_j) := M(v_j) \cup M'(v_0)$, where $x \in M'(v_0)$. For each odd j, $M(v_j) := L(v_1) - M'(v_0)$ at line (3.7), so $x \notin M(v_j)$.

Therefore $M(u) \cap M(v) = \emptyset$ for each pair of adjacent vertices, and the sets $M(v)$ are a $(2k : k)$ choice for C_{2m}. ∎

Let $\theta_{2,2,2}$ consist of the paths v_0, v_4, v_2 and v_0, v_3, v_2 and v_0, v_1, v_2. Given an assignment of $2k$-sets to the vertices of $\theta_{2,2,2}$, Algorithm 4 determines a $(2k : k)$-choice for the graph. The algorithm searches for a pair of vertices, v_s, v_t, adjacent to v_0 (or v_2) such that $L(v_0)$ (or $L(v_2)$) contains an element which is not in $L(v_s) \cap L(v_t)$. If such vertices exist, $T := \theta_{2,2,2} - (v_0, v_s) - (v_0, v_t)$ (or $T := \theta_{2,2,2} - (v_2, v_s) - (v_2, v_t)$) and Algorithm 1 is called. This search is iterated $i \leq k$ times, while such a pair of vertices exists, resulting in choice sets of i elements for v_0 and v_2. If $i = k$, then for each of v_1, v_3 and v_4, the remaining k elements in its colour set provide its choice set. If $i < k$, then for v_0 and v_2, an additional $k - i$ elements for its choice set are chosen from $L(v_0) \cap L(v_2)$. For each of v_1, v_3 and v_4, the remaining

$2k-i$ elements of its colour set, excluding the $k-i$ elements chosen above, provide a choice set of k elements.

Algorithm 4

Input: k, sets $L(v)$ for each $v \in V(\theta_{2,2,2})$
Initialization: $i := 0$, $M(v) := \emptyset$ for each $v \in V(\theta_{2,2,2})$

BEGIN
  WHILE ($i < k$ AND (there exists $x \in L(v_0) - L(v_s) - L(v_t)$ OR there exists $x \in L(v_2) - L(v_s) - L(v_t)$ where $s,t \in \{1,3,4\}$, $s \neq t$ DO BEGIN
    IF there exists $x \in L(v_0) - L(v_s) - L(v_t)$, where $s,t \in \{1,3,4\}$, $s \neq t$ THEN BEGIN
      $T := \theta_{2,2,2} - (v_0,v_s) - (v_0,v_t)$; (4.1)
      $r := v_0$;
      $m(r) := x$ (4.2)
    END
    ELSE IF there exists $x \in L(v_2) - L(v_s) - L(v_t)$, where $s,t \in \{1,3,4\}$, $s \neq t$ THEN BEGIN
      $T := \theta_{2,2,2} - (v_2,v_s) - (v_2,v_t)$;
      $r := v_2$;
      $m(r) := x$
    END;
    Call Algorithm 1; (4.3)
    Set $i := i + 1$
  END;
  IF $i = k$ THEN
    $M(v_j) := L(v_j)$ for $j = 1,3,4$ (4.4)
  ELSE BEGIN
    Choose $M'(v_0) \subseteq M'(v_2) \subseteq L(v_0) \cap L(v_2)$ where $|M'(v_0)| = k - i$; (4.5)
    Choose $M'(v_j) \subseteq L(v_j) - M'(v_0)$, where $|M'(v_j)| = k$, for $j = 1,3,4$; (4.6)
    $M(v_j) := M(v_j) \cup M'(v_j)$ for each vertex v_j (4.7)
  END
END.

Output: sets $M(v)$ for each $v \in V(\theta_{2,2,2})$

Complexity: $O(k^2)$

Algorithm 4 is iterated at most k times, where each iteration consists of a fixed set of operations, including at most one call to Algorithm 1. As the searches take time $O(|V(T)|) = O(1)$, the other operations take time $O(k)$, and Algorithm 1 always acts on a tree T where $|V(T)| = 5$, the time complexity of Algorithm 4 is $O(k^2)$.

Theorem 3.4 *For all $k \geq 1$, $\theta_{2,2,2}$ is $(2k:k)$-choosable.*

Proof. It suffices to show that the sets $M(v)$ for each $v \in V(\theta_{2,2,2})$ are a $(2k:k)$-choice for $\theta_{2,2,2}$. For any call to Algorithm 1, $r = v_0$ or $r = v_2$, so the vertices at an even distance from r are v_0 and v_2, and those at an odd distance from r are v_1, v_3 and v_4. For $j = 1$, 3 and 4, elements of $L(v_j)$ are added to $M(v_j)$ only at line (4.4) or (4.7).

Suppose that Algorithm 4 calls Algorithm 1 exactly i times. Then $|L(v_j)| = 2k - i$ for all v_j, and $|M(v_j)| = i$ for $j = 0$ and $j = 2$. If $i = k$, then $|L(v_j)| = k$ for each v_j, and $|M(v_j)| = k$ for $j = 0$ and $j = 2$. At line (4.4), $|M(v_j)| := |L(v_j)| = k$ for $j = 1$, 3 and 4. Thus $M(v_j) \subset L(v_j)$ and $|M(v_j)| = k$ for each $v_j \in V(\theta_{2,2,2})$.

If $i < k$, it must be shown that at line (4.5), $|L(v_0) \cap L(v_2)| \geq k - i$. Each element of $L(v_0)$ occurs in at least two of $L(v_1)$, $L(v_3)$, $L(v_4)$ and each element of $L(v_2)$ occurs in at least two of $L(v_1)$, $L(v_3)$, $L(v_4)$. Therefore $2|L(v_0)| + 2|L(v_2)| - 2|L(v_0) \cap L(v_2)| \leq |L(v_1)| + |L(v_3)| + |L(v_4)|$. As $|L(v_j)| = 2k - i$ for each v_j, $2(2k - i) + 2(2k - i) - 2|L(v_0) \cap L(v_2)| \leq (2k - i) + (2k - i) + (2k - i)$. Therefore $|L(v_0) \cap L(v_2)| \geq k - \frac{i}{2} > k - i$.

At line (4.7), $|M(v_j)| := |M(v_j) \cup M'(v_j)| = i + (k - i) = k$ for $j = 0$ and $j = 2$. As $M'(v_j) \subset L(v_0) \cap L(v_2)$, $M(v_j) \subset L(v_j)$ for each even j. At line (4.7), $|M(v_j)| := |M(v_j) \cup M'(v_j)| = |M'(v_j)| = k$ for $j = 1, 3, 4$. As $M'(v_j) \subseteq L(v_j) - M'(v_0)$, $M(v_j) \subset L(v_j)$ for $j = 1, 3, 4$.

The vertices at an odd distance from r are v_1, v_3 and v_4. For $j = 1, 3$ or 4, elements of $L(v_j)$ are added to $M(v_j)$ only at line (4.4) or (4.7). Thus if at any stage of Algorithm 4, $x \notin L(v_j)$, then x is not in $M(v_j)$ on completion of the algorithm. Suppose that $m(v) = x$ is added to $M(v)$ during a call to Algorithm 1. Then by Lemma 3.1, for any u which is adjacent to v in T, x will not be in $M(u)$ on completion of Algorithm 4.

Consider a call to Algorithm 1 where $T = \theta_{2,2,2} - (v_0, v_s) - (v_0, v_t)$. By line (4.2), $m(v_0) = x$, and x will be added to $M(v_0)$ at line (1.1). As $x \in L(v_0) - L(v_s) - L(v_t)$, $x \notin L(v_s)$ and $x \notin L(v_t)$, so x will not be in $M(v_s)$ or in $M(v_t)$ on completion of Algorithm 4. Similarly if a call is made when $T = \theta_{2,2,2} - (v_2, v_s) - (v_2, v_t)$.

Suppose $x \in M(v_0)$ was chosen at line (4.5). Then at line (4.7), $M(v_0) := M(v_0) \cup M'(v_0)$, where $x \in M'(v_0)$. For $j = 1$, 3 and 4, $M(v_j) := M(v_j) \cup M'(v_j) = M'(v_j)$, where $M'(v_j) \subseteq L(v_j) - M'(v_0)$. Thus $x \notin M'(v_j)$, so $x \notin M(v_j)$.

Thus $M(v) \cap M(u) = \emptyset$ for each pair of adjacent vertices, and the sets $M(v)$ are a $(2k:k)$ choice for $\theta_{2,2,2}$. ∎

For some $m \geq 1$, let $\theta_{2,2,2m}$ consist of the paths v_0, v_{2m+2}, v_{2m} and v_0, v_{2m+1}, v_{2m} and $v_0, v_1, v_2, \ldots, v_{2m}$. Given an assignment of $2k$-sets to the vertices of $\theta_{2,2,2m}$, Algorithm 5 determines a $(2k:k)$-choice for the graph.

For each iteration, a rooted subtree T of $\theta_{2,2,2m}$ is chosen and Algorithm 1 is called. If the root r is such that $d(r, v_0)$ is even, then i is incremented after the call. Otherwise, i' is incremented. This iteration occurs $i + i'$ times, while $i \leq k$, $i' \leq k$, and certain other conditions are satisfied, resulting in choice sets of i elements for vertices v_j such that $j = 0, 2, \ldots, 2m$ and choice sets of i' elements for v_j such that $j = 1, 3, \ldots, 2m+1, 2m+2$. If $i = k$, then for each of $v_1, v_3, \ldots, v_{2m+1}, v_{2m+2}$, the remaining $k - i'$ elements in its colour set provide the remainder of its choice set. If $i' = k$, then for each of $v_0, v_2, \ldots, v_{2m}$, the remaining $k - i$ elements in its colour set provide the rest of its choice set. If $i < k$ and $i' < k$, then ordered pairs of elements $\{(x_{2m+1}, x_{2m+2}) : x_{2m+1} \in L(v_{2m+1}), x_{2m+2} \in L(v_{2m+2})\}$ are formed, reducing the remainder of the $(2k : k)$-choice for $\theta_{2,2,2m}$ to a $(2k : k)$-choice for C_{2m+2}.

Algorithm 5

Input: k, m, sets $L(v)$ for each $v \in V(\theta_{2,2,2m})$
Initialization: $i := 0$, $i' := 0$, $M(v) := \emptyset$ for each $v \in V(\theta_{2,2,2m})$, $L'(v_{2m+1}) := \emptyset$

BEGIN
  WHILE [$i < k$] AND [$i' < k$] AND [there exists $x \in L(v_0) - L(v_s) - L(v_t)$ such that $s, t \in \{1, 2m+1, 2m+2\}$, $s \neq t$ OR there exists $x \in L(v_{2m}) - L(v_s) - L(v_t)$ such that $s, t \in \{2m-1, 2m+1, 2m+2\}$, $s \neq t$ OR there exists an element x which is contained in exactly three of $L(v_0)$, $L(v_{2m})$, $L(v_{2m+1})$, $L(v_{2m+2})$] DO BEGIN
    IF there exists $x \in L(v_0) - L(v_s) - L(v_t)$, $s, t \in \{1, 2m+1, 2m+2\}$, $s \neq t$ THEN BEGIN
      $T := \theta_{2,2,2m} - (v_0, v_s) - (v_0, v_t)$; (5.1)
      $r := v_0$;
      $m(r) := x$ (5.2)
    END
    ELSE IF there exists $x \in L(v_{2m}) - L(v_s) - L(v_t)$, where $s, t \in \{2m-1, 2m+1, 2m+2\}$, $s \neq t$ THEN BEGIN
      $T := \theta_{2,2,2m} - (v_{2m}, v_s) - (v_{2m}, v_t)$; (5.3)
      $r := v_{2m}$;
      $m(r) := x$
    END
    ELSE IF there exists $x \in (L(v_0) \cap L(v_{2m+1}) \cap L(v_{2m+2})) - L(v_{2m})$ THEN BEGIN
      $T := \theta_{2,2,2m} - (v_{2m}, v_{2m+1}) - (v_{2m}, v_{2m+2})$; (5.4)
      $r := v_{2m+1}$;
      $m(r) := x$ (5.5)
    END

ELSE IF there exists $x \in (L(v_{2m}) \cap L(v_{2m+1}) \cap L(v_{2m+2})) - L(r_0)$
THEN BEGIN
$T := \theta_{2,2,2m} - (v_0, v_{2m+1}) - (v_0, v_{2m+2})$ (5.6)
$r := v_{2m+1}$;
$m(r) := x$
END
ELSE IF there exists $x \in (L(r_0) \cap L(v_{2m}) \cap L(v_{2m+1})) - L(v_{2m+2})$
THEN BEGIN
Find the minimum $t \geq 0$ such that $x \in L(r_j)$ for $j = t$,
$t+1, \ldots, 2m$ (5.7)
IF t is even THEN BEGIN
$T := \theta_{2,2,2m} - (r_t, r_{t-1}) - (v_{2m}, v_{2m+2})$ (5.8)
$r := v_{2m}$;
$m(r) := x$ (5.9)
END
ELSE BEGIN
$m(v_{2m+2}) := y \in L(v_{2m+2}) - L(v_0)$;
$C := \theta_{2,2,2m} - (v_{2m}, v_{2m+2}) - (v_0, v_{2m+2})$;
IF $y \notin L(v_{2m})$ THEN BEGIN
$T := C - (v_t, v_{t-1})$; (5.10)
$r := v_t$;
$m(r) := x$ (5.11)
$M(v_{2m+2}) := M(v_{2m+2}) \cup \{y\}$
END
ELSE BEGIN
Find the minimum $u > 0$ so that $y \in L(r_j)$ for $j = u$,
$u+1, \ldots, 2m$ (5.12)
$T := C - (v_u, v_{u-1})$ (5.13)
$r := v_u$;
$m(r) := y$; (5.14)
IF u is odd THEN $M(v_{2m+2}) := M(v_{2m+2}) - \{y\}$ (5.15)
END
$L(v_{2m+2}) := L(v_{2m+2}) - \{y\}$; (5.16)
END;
END;
ELSE IF there exists $x \in (L(r_0) \cap L(v_{2m}) \cap L(v_{2m+2})) - L(v_{2m+1})$
THEN BEGIN
Find the minimum $t \geq 0$ such that $x \in L(r_j)$ for $j = t$,
$t = 1, \ldots, 2m$
IF t is even THEN BEGIN
$T := \theta_{2,2,2m} - (r_t, r_{t-1}) - (v_{2m}, v_{2m+1})$ (5.17)
$r := v_{2m}$;
$m(r) := x$

END
ELSE BEGIN
$m(v_{2m+1}) := y \in L(v_{2m+1}) - L(v_0)$:
$C := \theta_{2,2,2m} - (v_{2m}, v_{2m+1}) - (v_0, v_{2m+1})$:
IF $y \notin L(v_{2m})$ THEN BEGIN
$T := C - (v_t, v_{t-1})$: (5.18)
$r := v_t$:
$m(r) := x$
$M(v_{2m+1}) := M(v_{2m+1}) \cup \{y\}$
END
END
ELSE BEGIN
Find the minimum $u > 0$ so that $y \in L(v_j)$ for $j = u$.
$u = 1, \ldots, 2m$
$T := C - (v_u, v_{u-1})$ (5.19)
$r := v_u$:
$m(r) := y$
IF u is odd THEN
$M(v_{2m+1}) := M(v_{2m+1}) \cup \{y\}$
END:
$L(v_{2m+1}) := L(v_{2m+1}) - \{y\}$: (5.20)
END:
Call Algorithm 1: (5.21)
IF $r \in \{v_0, v_2, \ldots, v_{2m}\}$ THEN
$i := i + 1$ (5.22)
ELSE
$i' := i' + 1$;
IF $i = k$ THEN
$M(v_j) := M(v_j) \cup L(v_j)$ for $j = 2m + 2$ and each odd j (5.23)
ELSE IF $i' = k$ THEN
$M(v_j) := M(v_j) \cup L(v_j)$ for each even j, $0 \le j \le 2m$ (5.24)
ELSE BEGIN {pairing}
FOR all $x \in L(v_{2m+1})$ DO
IF $x \in L(v_{2m+2})$ THEN BEGIN
$L'(v_{2m+1}) := L'(v_{2m+1}) \cup \{(x, x)\}$ (5.25)
$L(v_{2m+1}) := L(v_{2m+1}) - \{x\}$
$L(v_{2m+2}) := L(v_{2m+2}) - \{x\}$
END
ELSE IF $x \in L(v_0)$ THEN BEGIN
$L'(v_{2m+1}) := L'(v_{2m+1}) \cup \{(x, y)\}$, where $y \in L(v_{2m+2}) \cap L(v_0)$ (5.26)
$L(v_{2m+1}) := L(v_{2m+1}) - \{x\}$
$L(v_{2m+2}) := L(v_{2m+2}) - \{y\}$

```
            END
            ELSE BEGIN
                L'(v_{2m+1}) := L'(v_{2m+1}) ∪ {(x, y)}, where y ∈ L(v_{2m+2}) −
                  L(v_{2m})                                                    (5.27)
                L(v_{2m+1}) := L(v_{2m+1}) − {x}
                L(v_{2m+2}) := L(v_{2m+2}) − {y}
            END;
        Call Algorithm 6;
        FOR each (x, y) ∈ L'(v_{2m+1}) DO BEGIN
            M(v_{2m+1}) := M(v_{2m+1}) ∪ {x};
            M(v_{2m+2}) := M(v_{2m+2}) ∪ {y}
        END
    END
END.
```

Output: sets $M(v)$ for each $v \in V(\theta_{2,2,2m})$

Complexity: $O(mk^2)$

Algorithm 5 is iterated at most $2k$ times, where each iteration consists of a fixed set of operations, including at most one call to Algorithm 1, either directly from Algorithm 5, or indirectly via Algorithm 6. As the searches take time $O(|V(T)|) = O(m)$, the other operations take time $O(k)$, and Algorithm 1 always acts on a tree T where $|V(T)| = 2m + 2$, the time complexity of Algorithm 5 is $O(mk^2)$.

Algorithm 6

Input: k, sets $L(v)$ for each $v \in V(C_{2m+2})$

```
BEGIN
    WHILE [i < k] AND [there exists an even j, 0 ≤ j < 2m + 1 such that
      either L(v_j) ≠ L(v_{j+1}) OR L(v_j) ≠ L(v_{j−1}) DO BEGIN
        IF there exists an even j such that L(v_j) ≠ L(v_{j+1}) DO BEGIN
            T := C_{2m+2} − (v_j, v_{j+1});
            r := v_j;
            m(r) := x where x ∈ L(v_j) − L(v_{j+1})                        (6.1)
        END
        ELSE IF there exists an even j such that L(v_j) ≠ L(v_{j−1}) DO BEGIN
            T := C_{2m+2} − (v_j, v_{j−1});
            r := v_j;
            m(r) := x where x ∈ L(v_j) − L(v_{j−1})                        (6.2)
        END;
        Call Algorithm 1;                                                  (6.3)
        i := i + 1
```

END;
IF $i = k$ THEN BEGIN
$M(v_j) := M(v_j) \cup L(v_j)$ for each odd $j < 2m + 1$; (6.4)
$M(v_{2m+1}) := L(v_{2m+1})$
ELSE BEGIN
Choose $M'(v_0) \subseteq L(v_0)$ where $|M'(v_0)| = k - i$; (6.5)
$M'(v_1) := L(v_1) - M'(v_0)$;
For each even j, $M(v_j) := M(v_j) \cup M'(v_0)$; (6.6)
For each odd $j < 2m + 1$, $M(v_j) := M(v_j) - M'(v_1)$; (6.7)
$M(v_{2m+1}) \subseteq L(v_{2m+1}) - M'(v_0)$, where $|M(v_{2m+1})| = k - i$ (6.8)
END
END.

Note that Algorithm 6 is almost identical to Algorithm 3. Algorithm 6 acts on a $2m + 2$-cycle, rather than a $2m$-cycle, and no initialization is required for Algorithm 6.

As the elements of all other sets are singletons, while the elements of $L(v_{2m+1})$ are ordered pairs, some definitions are required. Care is required here, as we are defining what we mean when saying that two sets containing very different objects are "equal". Define $L(v_j) = L(v_{2m+1})$ if and only if for each $x \in L(v_j)$, there exists $(y, z) \in L(v_{2m+1})$ such that either $y = x$ or $z = x$ (or both). Define $L(v_j) - L(v_{2m+1}) = \{x : x \in L(v_j)$ and for all $(y, z) \in L(v_{2m+1})$, $y \neq x$ and $z \neq x\}$. Define $L(v_{2m+1}) - M'(v_0) = \{(y, z) : (y, z) \in L(v_{2m+1})$, $y \notin M'(v_0)$ and $z \notin M'(v_0)\}$. In Algorithm 1, define $m(v_{2m+1}) \in L(u)$ if and only if $m(v_{2m+1}) = (y, z)$, where $y \in L(u)$ or $z \in L(u)$ (or both).

Lemma 3.5 *The pairing of the elements of $L(v_{2m+1})$ and $L(v_{2m+2})$ in Algorithm 5 occurs only if every $x \in L(v_0) \cup L(v_{2m}) \cup L(v_{2m+1}) \cup L(v_{2m+2})$ is contained in exactly two or in exactly four of $L(v_0)$, $L(v_{2m})$, $L(v_{2m+1})$, $L(v_{2m+2})$.*

Proof. If the elements of $L(v_{2m+1})$ and $L(v_{2m+2})$ are paired in Algorithm 5, then the following conditions are satisfied.

1. If $x \in L(v_0)$ then x is in at least two of $L(v_1)$, $L(v_{2m+1})$, $L(v_{2m+2})$;
2. If $x \in L(v_{2m})$ then x is in at least two of $L(v_{2m-1})$, $L(v_{2m+1})$, $L(v_{2m+2})$;
3. If $x \in L(v_0) \cap L(v_{2m+1}) \cap L(v_{2m+2})$, then $x \in L(v_{2m})$;
4. If $x \in L(v_{2m}) \cap L(v_{2m+1}) \cap L(v_{2m+2})$, then $x \in L(v_0)$;
5. If $x \in L(v_0) \cap L(v_{2m}) \cap L(v_{2m+1})$, then $x \in L(v_{2m+2})$;

6. If $x \in L(v_0) \cap L(v_{2m}) \cap L(v_{2m+2})$, then $x \in L(v_{2m+1})$.

By conditions 3 through 6, if x is in any three of $L(v_0)$, $L(v_{2m})$, $L(v_{2m+1})$, $L(v_{2m+2})$, then x is in all four of them. By conditions 1 and 2, if $x \in L(v_0)$ or $x \in L(v_{2m})$, then $x \in L(v_{2m+1})$ or $x \in L(v_{2m+2})$. Thus it suffices to show that if $x \in L(v_{2m+1})$ or $x \in L(v_{2m+2})$, then $x \in L(v_0)$, or $x \in L(v_{2m})$.

$$
\begin{aligned}
|L(v_0)| &= |L(v_0) \cap L(v_{2m+1})| + |L(v_0) \cap L(v_{2m+2})| - \\
&\quad |L(v_0) \cap L(v_{2m+1}) \cap L(v_{2m+2}) \cap L(v_{2m})| \\
|L(v_{2m})| &= |L(v_{2m}) \cap L(v_{2m+1})| + |L(v_{2m}) \cap L(v_{2m+2})| - \\
&\quad |L(v_{2m}) \cap L(v_{2m+1}) \cap L(v_{2m+2}) \cap L(v_0)| \\
|L(v_{2m+1})| &= |L(v_{2m+1}) - L(v_0) - L(v_{2m})| + |L(v_{2m+1}) \cap L(v_0)| + \\
&\quad |L(v_{2m+1}) \cap L(v_{2m})| - |L(v_{2m+1}) \cap L(v_0) \cap L(v_{2m}) \cap L(v_{2m+2})| \\
|L(v_{2m+2})| &= |L(v_{2m+2}) - L(v_0) - L(v_{2m})| + |L(v_{2m+2}) \cap L(v_0)| + \\
&\quad |L(v_{2m+2}) \cap L(v_{2m})| - |L(v_{2m+2}) \cap L(v_0) \cap L(v_{2m}) \cap L(v_{2m+1})|
\end{aligned}
$$

As $|L(v_0)| = |L(v_{2m})| = |L(v_{2m+1})| = |L(v_{2m+2})|$, $0 = |L(v_{2m+1})| + |L(v_{2m+2})| - |L(v_0)| - |L(v_{2m})| = |L(v_{2m+1}) - L(v_0) - L(v_{2m})| + |L(v_{2m+2}) - L(v_0) - L(v_{2m})|$.

Therefore $|L(v_{2m+1}) - L(v_0) - L(v_{2m})| = 0$ and $|L(v_{2m+2}) - L(v_0) - L(v_{2m})| = 0$, so any $x \in L(v_0) \cup L(v_{2m}) \cup L(v_{2m+1}) \cup L(v_{2m+2})$ is contained in exactly two or in exactly four of $L(v_0)$, $L(v_{2m})$, $L(v_{2m+1})$, $L(v_{2m+2})$. ∎

Lemma 3.6 *The pairing of the elements of $L(v_{2m+1})$ and $L(v_{2m+2})$ in Algorithm 5 occurs only if $|L(v_0) \cap L(v_{2m+1})| = |L(v_{2m}) \cap L(v_{2m+2})|$ and $|L(v_0) \cap L(v_{2m+2})| = |L(v_{2m}) \cap L(v_{2m+1})|$.*

Proof. $0 = |L(v_{2m+1})| + |L(v_0)| - |L(v_{2m+2})| - |L(v_{2m})| = 2|L(v_0) \cap L(v_{2m+1})| - 2|L(v_{2m}) \cap L(v_{2m+2})|$. Thus $|L(v_0) \cap L(v_{2m+1})| = |L(v_{2m}) \cap L(v_{2m+2})|$. Similarly for $|L(v_0) \cap L(v_{2m+2})| = |L(v_{2m}) \cap L(v_{2m+1})|$. ∎

Lemma 3.7 *The elements of $L(v_{2m+1})$ can be paired with elements of $L(v_{2m+2})$ in Algorithm 5.*

Proof. Suppose the elements of $L(v_{2m+1})$ and $L(v_{2m+2})$ are paired in Algorithm 5. If $x \in L(v_{2m+1}) \cap L(v_{2m+2})$ then (x, x) is paired at line (5.25). Otherwise $x \in L(v_{2m+1})$ and $x \notin L(v_{2m+2})$. If $x \in L(v_{2m+1}) \cap L(v_0)$ then by Lemma 3.6, there exists $y \in L(v_{2m+2}) - L(v_0)$ and (x, y) is paired at line (5.26). Otherwise $x \in L(v_{2m+1})$, $x \notin L(v_{2m+2})$ and $x \notin L(v_0)$, so $x \in L(v_{2m+1}) \cap L(v_{2m})$. By Lemma 3.6, there exists $y \in L(v_{2m+2}) - L(v_{2m})$ and (x, y) is paired at line (5.27). Thus the elements of $L(v_{2m+1})$ can be paired with the elements of $L(v_{2m+2})$. ∎

Lemma 3.8 *Suppose there is a call to Algorithm 1 where $m(r) = x$. If there exists a path $r = u_1, u_2, \ldots, u_\ell$ in T where $x \in u_i$ for $i = 1$ to ℓ, then $m(u_\ell) = x$ during the call to Algorithm 1.*

Proof. The proof is by induction on ℓ. Suppose that a call is made to Algorithm 1 where $m(r) = x$. If $\ell = 1$, then $m(r) = m(u_1) = x$ during the call to Algorithm 1.

Suppose that if there exists a path $r = u_1, u_2, \ldots, u_\ell$ in T where $x \in u_i$ for $i = 1$ to ℓ, then $m(u_{\ell-1}) = x$ during the call to Algorithm 1. Now $m(u_{\ell-1}) = x \in L(u_\ell)$, so $m(u_\ell) := x$ at line (1.3). ∎

Theorem 3.9 *For all $m \geq 1$, $k \geq 1$, $\theta_{2,2,2m}$ is $(2k : k)$-choosable.*

Proof. It will be shown that the sets $M(v)$ for each $v \in V(\theta_{2,2,2m})$ are a $(2k : k)$-choice for $\theta_{2,2,2m}$. After Algorithm 1 is called $i + i'$ times, $|L(v_j)| = 2k - i - i'$ for all v_j. For $j = 0, 2, \ldots, 2m$, $|M(v_j)| = i$, and for $j = 1, 3, \ldots, 2m+1, 2m+2$, $|M(v_j)| = i'$. If $i = k$, then $|L(v_j)| = k - i'$ for each v_j, and for $j = 0, 2, \ldots, 2m$, $|M(v_j)| = k$. Therefore $|M(v_j)| = |M(v_j) \cup L(v_j)| = i' + (k - i') = k$ for $j = 1, 3, \ldots, 2m+1, 2m+2$. If $i' = k$, then $|L(v_j)| = k - i$ for each v_j, and for $j = 1, 3, \ldots, 2m+1, 2m+2$, $|M(v_j)| = k$. Therefore, $|M(v_j)| = |M(v_j) \cup L(v_j)| = i + (k - i) = k$ for $j = 0, 2, \ldots, 2m$. If $i = k$ or $i' = k$, then $M(v_j) \subset L(v_j)$ and $|M(v_j)| = k$ for each v_j. If $i < k$ and $i' < k$, then the elements of $L(v_{2m+1})$ and $L(v_{2m+2})$ are paired.

By Lemma 3.1, if $m(v) = x$ is added to $M(v)$ during a call to Algorithm 1, then for any u which is adjacent to v in T, x will not be in $L(u)$ after the call, and x will not be added to $M(u)$ during the call. Thus it suffices to consider choices for vertices which are adjacent in $\theta_{2,2,2m}$, but which are not adjacent in T.

Suppose there is a call to Algorithm 1 where $T = \theta_{2,2,2m} - (v_0, v_s) - (v_0, v_t)$ (line (5.1)). By line (5.2), $m(v_0) = x$ and x is added to $M(v_0)$ at line (1.1). As $x \in L(v_0) - L(v_s) - L(v_t)$, $x \notin L(v_s)$ and $x \notin L(v_t)$. Similarly if there is a call to Algorithm 1 where $T = \theta_{2,2,2m} - (v_{2m}, v_s) - (v_{2m}, v_t)$ (line (5.3)).

Suppose there is a call made to Algorithm 1 where $T = \theta_{2,2,2m} - (v_{2m}, v_{2m+1}) - (v_{2m}, v_{2m+2})$ (line (5.4)). By line (5.5), $m(v_{2m+1}) = x$ and x is added to $M(v_{2m+1})$ at line (1.1). As $x \in (L(v_0) \cap L(v_{2m+1}) \cap L(v_{2m+2})) - L(v_{2m})$, $x \notin L(v_{2m})$, but $x \in L(v_0)$ and $x \in L(v_{2m+2})$. The next choice will be $m(v_0)$, and the choice after that will be $m(v_{2m+2})$. As $m(v_{2m+1}) = x \in L(v_0)$, $m(v_0) := x$ at line (1.3). As $m(v_0) = x \in L(v_{2m+2})$, $m(v_{2m+2}) := x$ at line (1.3). As $d(v_{2m+1}, v_{2m+2})$ is even, x will be added to $M(v_{2m+2})$ at line (1.6). Similarly if there is a call to Algorithm 1 where $T = \theta_{2,2,2m} - (v_0, v_{2m+1}) - (v_0, v_{2m+2})$ (line (5.6)).

Suppose there is a call to Algorithm 1 where $T = \theta_{2,2,2m} - (v_t, v_{t-1}) - (v_{2m}, v_{2m+2})$ (line (5.8)). By line (5.9), $m(v_{2m}) = x$ and x is added to $M(v_{2m})$ at line (1.1). As $x \in (L(v_0) \cap L(v_{2m}) \cap L(v_{2m+1})) - L(v_{2m+2})$, $x \notin L(v_{2m+2})$. There exists a path $r = v_{2m}, v_{2m-1}, \ldots, v_{t-1}, v_t$ in T where

x is in $L(v_i)$ for each v_i in the path. So by Lemma 3.8, $m(v_t) = x$ during the call to Algorithm 1. As $d(v_{2m}, v_t)$ is even, x is added to $M(v_t)$ at line (1.6). As the minimum t was chosen at line (5.7), $x \notin L(v_{t-1})$. Similarly if there is a call to Algorithm 1 where $T = \theta_{2,2,2m} - (v_t, v_{t-1}) - (v_{2m}, v_{2m+1})$ (line (5.17)).

Suppose there is a call made to Algorithm 1 where $T = \theta_{2,2,2m} - (v_{2m}, v_{2m+2}) - (v_0, v_{2m+2}) - (v_t, v_{t-1})$ (line (5.10)). By line (5.11), $m(v_t) = x$ and x is added to $M(v_t)$ at line (1.1). As the minimum t was chosen at line (5.7), $x \notin L(v_{t-1})$. There exists a path $r = v_t, v_{t+1}, \ldots, v_{2m}, v_{2m+1}, v_0$ in T where x is in $L(v_i)$ for each v_i in the path. So by Lemma 3.8, $m(v_{2m}) = x$ and $m(v_0) = x$ during the call to Algorithm 1. As $x \in (L(v_0) \cap L(v_{2m}) \cap L(v_{2m+1})) - L(v_{2m+2})$, $x \notin L(v_{2m+2})$. Similarly if there is a call to Algorithm 1 where $T = \theta_{2,2,2m} - (v_{2m}, v_{2m+1}) - (v_0, v_{2m+1}) - (v_t, v_{t-1})$ (line (5.18)).

Suppose there is a call made to Algorithm 1 where $T = \theta_{2,2,2m} - (v_{2m}, v_{2m+2}) - (v_0, v_{2m+2}) - (v_u, v_{u-1})$ (line (5.13)). By line (5.14), $m(v_u) = y$ and y is added to $M(v_u)$ at line (1.1). As the minimum u was chosen at line (5.12), $y \notin L(v_{u-1})$. There exists a path $r = v_u, v_{u+1}, \ldots, v_{2m}$ in T where y is in $L(v_i)$ for each v_i in the path. So by Lemma 3.8, $m(v_{2m}) = y$ during the call to Algorithm 1. As $y \in L(v_{2m+2}) \cap L(v_0)$, $y \notin L(v_0)$. Suppose that u is even. Then since $d(v_u, v_{2m})$ is even, y is added to $M(v_{2m})$ at line (1.6). As $y \notin L(v_0)$, some $m(v_0) = z \in L(v_0)$ is added to $M(v_0)$ at line (1.6). As u is even, y is not added to $M(v_{2m+2})$ at line (5.15), and y is deleted from $L(v_{2m+2})$ at line (5.16). Suppose that u is odd. Then y is added to $M(v_{2m+2})$ at line (5.15). At line (1.5), y is deleted from $L(v_{2m})$, and as $d(v_u, v_{2m})$ is odd, y is not added to $M(v_{2m})$ at line (1.6). Similarly if there is a call to Algorithm 1 where $T = \theta_{2,2,2m} - (v_{2m}, v_{2m+1}) - (v_0, v_{2m+1}) - (v_u, v_{u-1})$ (line (5.19)). ∎

An example demonstrating the application of Algorithm 5 to a $\theta_{2,2,4}$ graph follows. Suppose sets $L(v_0) = \{1, 2, 3, 4\}$, $L(v_1) = \{1, 2, 5, 6\}$, $L(v_2) = \{1, 3, 4, 5\}$, $L(v_3) = \{4, 5, 6, 7\}$, $L(v_4) = \{5, 6, 7, 8\}$, $L(v_5) = \{1, 2, 4, 8\}$ and $L(v_6) = \{3, 5, 6, 7\}$ are assigned to the vertices of $\theta_{2,2,4}$.

The algorithm begins with a search for an element of $L(v_0)$ which is not in the colour sets of at least two of the vertices adjacent to v_0. Elements $1, 2 \in L(v_0)$ do not satisfy this criterion, but element $3 \in L(v_0)$ does. It is contained in neither $L(v_1)$ nor $L(v_5)$. At line (5.1), tree T is obtained by removing edges (v_0, v_1) and (v_0, v_5). Vertex v_0 is chosen as the root of T, and at line (5.2), $m(v_0) = 3$ is chosen from its colour set. At line (5.21), Algorithm 1 is called, beginning a breadth-first search on T, where an element will be chosen from each colour set.

Algorithm 1 begins by adding $m(v_0) = 3$ to choice set $M(v_0)$, and deleting it from set $L(v_0)$. Then the neighbour v_6 of v_0 is considered.

Because $m(v_0) = 3 \in L(v_6)$, $m(v_6) = 3$ is chosen at line (1.3). Element 3 is deleted from $L(v_6)$ at line (1.5). The neighbour v_4 of v_6 is considered next. As $m(v_6) = 3 \notin L(v_4)$, an element of $L(v_4)$ which is not in $L(v_6)$ must be chosen at line (1.4). Element $m(v_4) = 8$ satisfies this criterion. It is deleted from $L(v_4)$ at line (1.5), and because v_4 is at an even distance from the root v_0, element 8 is added to choice set $M(v_4)$ at line (1.6). Continuing in this fashion, $m(v_5) = 8$ and $m(v_3) = 4$ are chosen, and are deleted from sets $L(v_5)$ and $L(v_3)$, respectively. Next, $m(v_2) = 4$ is chosen, deleted from $L(v_2)$ and added to $M(v_2)$. Finally, $m(v_1) = 2$ is chosen and deleted from $L(v_1)$, completing this call to Algorithm 1.

As root v_0 was chosen, i is incremented at line (5.22). Because $i < 2$, this iteration of Algorithm 5 is completed, and there is a return to the beginning of the loop.

Again, there is a search for an element in $L(v_0)$ which is not in the sets of at least two of its adjacent vertices. As element 2 was deleted from $L(v_1)$, this element is in neither $L(v_1)$ nor $L(v_6)$, thus satisfying the criterion. At line (5.1), tree T is obtained by removing edges (v_0, v_1) and (v_0, v_6) from the $\theta_{2,2,4}$ graph. The root of T becomes vertex v_0, $m(v_0) = 2$ is chosen, and Algorithm 1 is called.

First, $m(v_0) = 2$ is added to $M(v_0)$ and is deleted from $L(v_0)$. Now $m(v_5) = 2$ is chosen and deleted from $L(v_5)$. Then, $m(v_4) = 5$ is chosen, deleted from $L(v_4)$ and added to $M(v_4)$. Next, $m(v_6) = 5$ and $m(v_3) = 5$ are chosen, and are deleted from sets $L(v_6)$ and $L(v_3)$, respectively. Then, $m(v_2) = 5$ is chosen, deleted from $L(v_2)$ and added to $M(v_2)$. Finally, $m(v_1) = 5$ is chosen and deleted from $L(v_1)$, completing this call to Algorithm 1.

Once again, i is incremented at line (5.22). Now $i = 2$, which means that the choice sets for the vertices at an even distance from v_0 have been completed. They are $M(v_0) = \{2, 3\}$, $M(v_2) = \{4, 5\}$ and $M(v_4) = \{5, 8\}$. At line (5.23), the choice sets for the remaining vertices are completed by adding their remaining elements to their respective choice sets. Set $L(v_1)$ had elements 2 and 5 deleted, and has elements 1 and 6 remaining, so choice set $M(v_1)$ becomes $\{1, 6\}$. Similarly, $M(v_3) = \{6, 7\}$, $M(v_5) = \{1, 4\}$ and $M(v_6) = \{6, 7\}$. This completes the choices, and Algorithm 5.

Theorem 3.10 *Any 2-choosable graph is $(2k : k)$-choosable for any $k \geq 1$.*

Proof. By the results in this section, and the characterization of 2-choosable graphs given by Theorem 2.2, the core of any graph which is 2-choosable is $(2k : k)$-choosable for all $k \geq 1$. By Lemma 2.3, any graph whose core is $(2k : k)$-choosable is $(2k : k)$-choosable. ∎

References

[1] P. Erdös, A.L. Rubin and H. Taylor, *Choosability in graphs*, Proc. West Coast Conf. on Combinatorics, Graph Theory and Computing, Congressus Numerantium **26** (1979) 125-157.

[2] S. Ferneyhough, **Choice Numbers for Unions of Graphs**, M.Sc. Thesis, Department of Mathematics and Statistics, University of Victoria, Victoria, B.C., Canada (1996).

[3] S. Gutner, M.Sc. Thesis, Tel Aviv University (1992).

[4] S. Gutner and M. Tarsi, *Some results on* $(a : b)$*-choosability* (1995), manuscript.

[5] Zs. Tuza and M. Voigt, *Every 2-choosable graph is* $(2m, m)$*-choosable*, Journal of Graph Theory, **22** (1996) 245-252.