Coloring Artemis graphs

Benjamin Lévêque[*]   Frédéric Maffray[†]   Bruce Reed[‡]

Nicolas Trotignon[§]

April 19, 2005

Abstract

We consider the class $\mathcal{A}$ of graphs that contain no odd hole, no antihole, and no "prism" (a graph consisting of two disjoint triangles with three disjoint paths between them). We show that the coloring algorithm found by the second and fourth author can be implemented in time $O(n^2m)$ for any graph in $\mathcal{A}$ with n vertices and m edges, thereby improving on the complexity proposed in the original paper.

1   Introduction

We denote by $\chi(G)$ the chromatic number of a graph G and by $\omega(G)$ the maximum clique size in G. An *even pair* in a graph G is a pair $\{x, y\}$ of non-adjacent vertices having the property that every chordless path between them has even length (number of edges). Given two vertices x, y in a graph G, the operation of *contracting* them means removing x and y and adding one vertex with edges to every vertex of $G \setminus \{x, y\}$ that is adjacent in G to at least one of x, y; and we denote by G/xy the graph that results from this operation. Fonlupt and Uhry [3] proved that *if $\{x, y\}$ is an even pair in a graph G, then $\chi(G/xy) = \chi(G)$ and $\omega(G/xy) = \omega(G)$.* In particular, given a $\chi(G/xy)$-coloring c of the vertices of G/xy, one can easily obtain a $\chi(G)$-coloring of the vertices of G by assigning to x and y the color assigned by c to the contracted vertex and keeping the color of every vertex different from x, y. This idea is the basis of a conceptually simple coloring algorithm: as long as the graph has an even pair, contract any such pair; when there is no even pair find a coloring c of the contracted graph and, applying the above procedure repeatedly, derive from c a coloring of the original

graph. In this perspective, a graph G is called *even-contractile* [1] if it can be turned into a clique by a sequence of contractions of even pairs, and the graph is called *perfectly contractile* if every induced subgraph of G is even-contractile. We propose here a fast implementation of the above algorithm for a class of perfectly contractile graphs studied in [7].

A *hole* is a chordless cycle with at least four vertices and an *antihole* is the complement of a hole. A *prism* is a graph that consists of two vertex-disjoint triangles (cliques of size three) with three vertex-disjoint paths between them, and with no other edge than those in the two triangles and in the three paths. Let $\mathcal{A}$ be the class of graphs that contain no odd hole, no antihole of length at least 5, and no prism (such graphs have also been called "Artemis graphs" [2]). Maffray and Trotignon [7] proved Everett and Reed's conjecture [2, 8] that every graph in class $\mathcal{A}$ is perfectly contractile. The proof contains an algorithm which, given any graph G in class $\mathcal{A}$ with n vertices and m edges, finds an optimal coloring of the vertices of G in time $O(n^4 m)$. The point of this note is to show that this coloring algorithm can be implemented in time $O(n^2 m)$.

In a graph $G = (V, E)$, we say that a vertex u *sees* a vertex v when u, v are adjacent, else we say that u *misses* v. For any $X \subseteq V$, the subgraph induced by X is denoted by $G[X]$, and $N(X)$ denotes the set of vertices of $V \setminus X$ that see at least one vertex of X. A vertex of $V \setminus X$ is called X-*complete* if it sees every vertex of X; and $C(X)$ denotes the set of X-complete vertices of $V \setminus X$. The complementary graph of G is denoted by $\overline{G}$. The length of a path is the number of its edges. An edge between two vertices that are not consecutive along the path is a *chord*, and a path that has no chord is *chordless*. A vertex is *simplicial* if its neighbours are pairwise adjacent.

2 The method

We recall the method from [7]. An even pair $\{a, b\}$ in a graph G is called *special* if the graph G/ab contains no prism.

Lemma 2.1 ([2, 7]) *If G is in class $\mathcal{A}$ and $\{a, b\}$ is a special even pair of G, then G/ab is in class $\mathcal{A}$.*

The proof from [7] consists in finding a special even pair and contracting it. Since Lemma 2.1 ensures that the contracted graph is still in $\mathcal{A}$, the algorithm can be iterated until the graph is a clique. Actually we will stop when the graph is a dsjoint union of cliques, which can be colored optimally by the greedy method. Let us now recall how a special even pair is found when the graph is not a disjoint union of cliques.

A non-empty subset $T \subseteq V$ is called *interesting* if $\overline{G}[T]$ is connected (in short we will say that T is co-connected) and $G[C(T)]$ is not a clique (so $|C(T)| \geq 2$ since we view the empty set as a clique). An interesting set is *maximal* if it is not strictly included in another interesting set. A T-*outer path* is a chordless path whose two endvertices are in $C(T)$ and whose interior vertices are all in $V \setminus (T \cup C(T))$. A T-outer path P is *minimal* if there is no T-outer path whose

interior is strictly contained in the interior of P. The search for a special even pair considers three cases: (1) when the graph has no interesting set; (2) when a maximal interesting set T of G has no T-outer path; (3) when a maximal interesting set T of G has a T-outer path. These three cases correspond to the following three lemmas.

Lemma 2.2 ([7]) *For any graph G the following conditions are equivalent:*
(1) G has no interesting set,
(2) Every vertex of G is simplicial,
(3) G is a disjoint union of cliques.
Moreover, if G is not a disjoint union of cliques then every non-simplicial vertex forms an interesting set.

Lemma 2.3 ([7]) *Let G be a graph in $\mathcal{A}$ that contains an interesting set, and let T be any maximal interesting set in G. If T has no T-outer path, then every special even pair of the subgraph $G[C(T)]$ is a special even pair of G.*

When a maximal interesting set T has a T-outer path, we let $\alpha z_1 \cdots z_p \beta$ be a minimal T-outer path and we define sets:

$$
\begin{aligned}
A &= \{v \in C(T) \mid vz_1 \in E, vz_i \notin E \ (i = 2, \ldots, p)\}, \\
B &= \{v \in C(T) \mid vz_p \in E, vz_i \notin E \ (i = 1, \ldots, p-1)\}.
\end{aligned}
$$

Define a relation $<_A$ on A by setting $u <_A u'$ if and only if $u, u' \in A$ and there exists an odd chordless path from u to a vertex of B such that u' is the second vertex of that path (where u is the first vertex). Likewise define a relation $<_B$ on B by setting $v <_B v'$ if and only if $v, v' \in B$ and there exists an odd chordless path from v to a vertex of A such that v' is the second vertex of that path.

Lemma 2.4 ([7]) *When A, B and $<_A, <_B$ are defined as above they satisfy:*
(1) The sets A and B are non-empty cliques with no edge between them.
(2) If $P = uu' \cdots v'v$ is a chordless odd path with $u \in A$ and $v \in B$, then either $u' \in A$ or $v' \in B$ holds.
(3) The relation $<_A$ is a strict partial order on A. The relation $<_B$ is a strict partial order on B.
(4) If a is any maximal vertex of $<_A$ and b is any maximal vertex of $<_B$, then $\{a, b\}$ is a special even pair of G.

Lemma 2.5 *Let T be a maximal interesting set in a graph G and a, b be any two non-adjacent vertices in $C(T)$. Let $C'(T)$ be the set of T-complete vertices in G/ab. If $C'(T)$ is not a clique then T is a maximal interesting set in G/ab.*

The proof is easy and we omit it.

We find a special even pair as follows: first an algorithm finds a maximal interesting set T in G. Then a second algorithm finds a special even pair in $C(T)$, on the basis of Lemmas 2.3 and 2.4, and contracts it; this second algorithm is iterated as long as the set $C(T)$ is not a clique, which is possible by Lemmas 2.1

and 2.5. When the set $C(T)$ becomes a clique, the first algorithm is called again to find another maximal interesting set. Since the contraction of an even pair reduces the number of vertices by 1, there will be at most n contractions. So the total complexity is n times the complexity of finding a special even pair. We will see in the next sections that a special even pair can be found in time $O(nm)$, so the total complexity of the coloring algorithm is $O(n^2m)$.

3 Finding a maximal interesting set

Lemma 3.1 *Let T be an interesting set in a graph G. If there is a vertex $u \in V \setminus (T \cup C(T))$ such that $N(u) \cap C(T)$ is not a clique then $T \cup \{u\}$ is an interesting set. If there is no such vertex then T is a maximal interesting set.*

The proof is easy and we omit it.

> **Algorithm Find_Interesting**
>
> *Input:* A graph G.
>
> *Output:* Either a maximal interesting set T of G or the answer "G is a disjoint union of cliques".
>
> *Method:*
>
> *Step 1: Looking for a non-simplicial vertex t.*
>
> Compute the components of G. If every vertex has degree equal to the size of its component minus 1, return the answer "G is a disjoint union of cliques" and stop. Else, consider a vertex u whose degree is strictly less than the size of its component minus 1. Perform a breadth-first search from u, let v be any vertex at distance 2 from u, and let t be the parent of v in the search.
>
> *Step 2: Building T from t.*
>
> Set $T := \{t\}$, $C := N(t)$, $U := V \setminus (T \cup C)$, $Z := \emptyset$.
>
> While there exists a vertex $u \in U$ do:
> If $N(u) \cap C$ is a clique, move u from U to Z.
> If $N(u) \cap C$ is not a clique, move u from U to T and move every vertex of $C \setminus N(u)$ from C to U.
>
> Return the set T and stop.

Lemma 3.2 *Algorithm Find_Interesting is correct.*

Proof. Clearly, Step 1 of the algorithm is correct. At the beginning of Step 2 the set T is interesting and C is equal to the set of T-complete vertices and is not a clique. The definition of Step 2 implies that these properties remain true throughout, and Lemma 3.1 ensures that when Step 2 terminates the set T is a maximal interesting set. $\qquad\square$

Lemma 3.3 *The complexity of Algorithm Find_Interesting is $O(\max\{n + m, m(n-k)\})$ where k is the number of vertices in $C(T)$ for the output set T (if no set T is output, we consider $k = n$ and the complexity is $O(n + m)$).*

Proof. Step 1 takes time $O(n+m)$ steps. In Step 2, a vertex can only move from C to U or from U to Z or to T. So the sets T and Z can only increase and the sets U and C can only decrease. Deciding whether $N(u) \cap C$ is a clique takes time $O(m)$, and updating C takes time $O(\deg(t))$ since C can only decrease from its initial value $N(t)$. Thus, each iteration of the while loop takes time $O(m)$. A vertex plays the role of u at most once, and the k vertices that are in C when the algorithm stops have never played such a role. So there are at most $n - k$ iterations of the while loop. $\qquad\square$

4 Looking for an outer path

Lemma 4.1 *An interesting set T has an outer path if and only if there exists a component R of $V \setminus (T \cup C(T))$ such that $N(R) \cap C(T)$ is not a clique.*

The proof is easy and we omit it.

Lemma 4.2 ([7]) *Let G be a graph in $\mathcal{A}$ that contains an interesting set, and let T be any maximal interesting set in G. Then every T-outer path has length even and at least 4.*

Given two disjoints subsets $X, Y \subseteq V$ of a graph G, we call *breadth first search (BFS) from X to Y in G* any breadth first search such that (a) the vertices of X form the root level, and (b) the vertices of Y may only appear as leaves in the search tree. Points (a) and (b) can be implemented as in the usual form of BFS by using a queue from which we get the next vertex to be scanned, the only modification being that we put all vertices of X in the queue at the start of the search and we never put any vertex of Y in the queue.

> **Algorithm Find_Outer_Path**
>
> *Input:* A graph G and a maximal interesting set T of G.
>
> *Output:* Either a minimal T-outer path or the answer "G has no T-outer path".
>
> *Method:* Initially all vertices of $V(G) \setminus (T \cup C(T))$ are unmarked.
> **while** there is an unmarked vertex r in $V(G) \setminus (T \cup C(T))$ do:
> Start a Breadth-First Search S from r to $C(T)$ in $G \setminus T$, mark each vertex of $V(S) \setminus C(T)$, and maintain the set $M = V(S) \cap C(T)$.
> When a vertex x is added to M, **if** the new M is not a clique, do:
> Let $M_x := M \cap N(x)$. Perform a BFS from x in the subgraph $G[S \setminus M_x]$. Let y be the first vertex of $M \setminus M_x$ that is reached by this search, and let x-v-$\cdots$-w-y be the path from x to y given by this search. Return this path and stop. **endif**
> **endwhile**
> Return the answer "G has no T-outer path" and stop.

Lemma 4.3 *Algorithm Find_Outer_Path is correct.*

Proof. Let R be the component of $V(G) \setminus (T \cup C(T))$ that contains r. The search from r potentially reaches all vertices of R and of $N(R) \cap C(T)$ and puts the latter into M. If $N(R) \cap C(T)$ is a clique, the search will mark all vertices of R and continue with a new vertex r from another component of $V(G) \setminus (T \cup C(T))$, if any. Lemma 4.1 ensures that the algorithm will correctly return the answer "G has no T-outer path" if and only if G has no outer path. There remains to show that when the algorithm returns a path, it is a minimal T-outer path. So let us examine the situation in this case. For some component R of $V(G) \setminus (T \cup C(T))$ the set $N(R) \cap C(T)$ is not a clique, and the search from a vertex $r \in R$ finds the vertex x on the first time M is no longer clique. The set $M \setminus M_x$ is not empty. The only neighbours of x in $S \setminus M$ are either its parent in the search or vertices that are still in the queue, for otherwise x would have been added to S earlier. So every vertex of $S \setminus M$ that is not adjacent to x has been scanned before the neighbours of x in $S \setminus M$.

The search from x in $G[S \setminus M_x]$ will potentially reach all vertices of $M \setminus M_x$. So the vertex y and the path $x\text{-}v\text{-}\cdots\text{-}w\text{-}y$ exist. Let us rewrite this path as $P = x\text{-}z_1\text{-}\cdots\text{-}z_p\text{-}y$, with $z_1 = v$ and $z_p = w$, and write $Z = \{z_1, \ldots, z_p\}$. To show that P is a T-outer path, suppose on the contrary that some element z_i of Z is in $C(T)$ and let i be the smallest such integer ($1 \leq i \leq p$). We have $i \geq 2$ because $z_1 = v$ which is in R; but then z_i contradicts the definition of y. So all of $z_1, \ldots, z_p$ are in R, which means that P is a T-outer path. To prove the minimality of P, suppose on the contrary that there exists a T-outer path $x'\text{-}z_i\text{-}\cdots\text{-}z_j\text{-}y'$ with $1 \leq i \leq j \leq p$ and $j - i < p - 1$. By Lemma 4.2, $j - i$ is even and at least 2, so $j > 2$. Vertex y' is in $M \setminus \{x\}$ since z_j has been added to S before z_1. If $i > 1$ then x' too is in $M \setminus \{x\}$, since z_i has been added to S before z_1; but then $M \setminus \{x\}$ is not a clique, which contradicts the definition of x. So $i = 1$. If x is adjacent to y' then $x\text{-}z_1\text{-}\cdots\text{-}z_j\text{-}y'\text{-}x$ is a hole of odd length $j - 1 + 3 \geq 5$, a contradiction. So x is not adjacent to y', but then $x\text{-}z_1\text{-}\cdots\text{-}z_j\text{-}y'$ is a chordless path and y' contradicts the definition of y. So P is a minimal T-outer path. $\qquad\square$

Lemma 4.4 *The complexity of Algorithm Find_Outer_Path is $O(lm)$, where l is the number of components of $G \setminus (T \cup C(T))$.*

Proof. The search from a vertex r reaches all the vertices of the component R of $V(G) \setminus (T \cup C(T))$ that contains r and the vertices of $N(R) \cap C(T)$, and only them. Moreover these vertices are scanned only once during this search. The search from x reaches vertices of R a second time. Thus vertices of R are scanned at most twice. In order to check whether M is a clique, we use a counter for each vertex u of $C(T)$, which counts the number of neighbours of u in M. Whenever a new vertex u is added to M, we check if the counter of u is equal to $|M|$, and we scan u to increase by 1 the counter of its neighbours in $C(T)$. Thus the complexity for one component R is $O(m(R))$, where $m(R)$ is the number of edges in the subgraph induced by $R \cup (N(R) \cap C(T))$. However, a vertex u

of $C(T)$ may be scanned several times, depending on the number of sets of the type $N(R) \cap C(T)$ that contain it. So the total complexity is $O(lm)$. $\square$

When the set T has no T-outer path, Lemma 2.3 says that we need to continue the search recursively in the subgraph $G[C(T)]$. Thus we may have to find a maximal interesting set T_1 of G, then (putting $C_1 = C(T_1)$) find a maximal interesting set T_2 of $G[C_1]$, then (putting $C_2 = C(T_2) \cap C_1$) find a maximal interesting set T_3 of $G[C_2]$, up to (putting $C_{q-1} = C(T_{q-1}) \cap C_{q-2}$) a maximal interesting set T_q of $G[C_{q-1}]$ such that either there is a T_q-outer path in $G[C_{q-1}]$ or $G[C(T_q) \cap C_{q-1}]$ is a disjoint union of cliques. Let us analyze the complexity of this procedure.

For $i = 1, \ldots, q$, put $n_i = |C_i|$ and $m_i = |E(G[C_i])|$, and put $n_0 = n$ and $m_0 = m$. By Lemma 3.3, the total complexity of finding interesting sets over all the recursive calls is $O(\Sigma_{i=1}^{q-1} m_{i-1}(n_{i-1} - n_i) + \max\{n_{q-1} + m_{q-1}, m_{q-1}(n_{q-1} - n_q)\}) = O(\max\{n + m, mn\})$.

For $i = 1, \ldots, q$, let l_i be the number of components of $G[C_{i-1} \setminus (T_i \cup C(T_i))]$. Observe that all these components (over all $i = 1, \ldots, q$) are pairwise disjoint, so $l_1 + \cdots + l_q \le n$. By Lemma 4.4, the total complexity, over all recursive calls, of finding outer paths is $O(\Sigma_{i=1}^{q} l_i m) = O(nm)$.

So the total complexity of this recursive procedure is $O(nm)$.

5 Finding a special even pair

Algorithm Find_Even_Pair

Input: A graph G, a maximal interesting set T and the minimal T-outer path x-v-$\cdots$-w-y given by Algorithm Find_Outer_Path.

Output: A special even pair of G

Method:
1. Set $A := (N(v) \cap C(T)) \setminus N(y)$ and $B := (N(w) \cap C(T)) \setminus N(x)$.
2. Perform a BFS from B to $N(A)$ in $G \setminus (T \cup A)$ and call K the set of vertices of $N(A)$ that are reached by this search.
3. Perform a BFS from A to $N(B)$ in $G \setminus (T \cup B)$ and call L the set of vertices of $N(B)$ that are reached by this search.
4. Let a be a vertex of A that sees all of K.
5. Let b be a vertex of B that sees all of L.
6. Return the pair $\{a, b\}$.

Lemma 5.1 *The preceding algorithm returns a pair of vertices $\{a, b\}$ that is a special even pair of G.*

Proof. Let us rewrite the path x-v-$\cdots$-w-y as $P = x$-z_1-$\cdots$-z_p-y, with $z_1 = v$ and $z_p = w$, and write $Z = \{z_1, \ldots, z_p\}$. Define sets $A' = \{u \in C(T) \mid uz_1 \in E(G), uz_i \notin E(G)(i = 2, \ldots, n)\}$ and $B' = \{u \in C(T) \mid uz_n \in E(G), uz_i \notin E(G)(i = 1, \ldots, n-1)\}$. These are the sets mentioned in Lemma 2.4. We claim that the sets A, B defined in the algorithm satisfy $A = A'$ and $B = B'$. First

observe that $x \in A'$ and $y \in B'$ and that there is no edge $a'b'$ with $a' \in A'$ and $b' \in B'$, for otherwise $Z \cup \{a', b'\}$ would induce an odd hole of length $p + 2 \geq 5$. This implies $A' \subseteq A$ and $B' \subseteq B$. Now let a be any vertex of A. Suppose that a has a neighbour z_i in Z with $i \geq 2$, and let i be the largest such integer. By the definition of A, vertices a and y are non-neighbours. Then a-z_i-$\cdots$-z_p-y is a T-outer path, which contradicts the minimality of P. So a has no neighbour in $Z \setminus \{z_1\}$. So $A \subseteq A'$. Similarly $B \subseteq B'$. So $A = A'$ and $B = B'$ as claimed.

There remains to show that lines 2–5 of the algorithm correctly produce maximal elements of $(A, <_A)$ and $(B, <_B)$. Let K be as defined by the algorithm. Let a^* be a maximal vertex for the relation $<_A$. Suppose a^* is not adjacent to a vertex u of K. Let $a' \in A \setminus \{a^*\}$ be a neighbour of u (a' exists by the definition of K), and let $Q = q_1$-$\cdots$-q_k be the chordless path from $q_1 \in B$ to $u = q_k$ given by the search tree of line 2 of the algorithm. Since A is a clique, a^* and a' are adjacent. Suppose a vertex of A is adjacent to a vertex q_i with $1 \leq i \leq k - 1$. Then q_i is a vertex of $N(A)$ and the search should not have been continued from q_i; but this contradicts the existence of q_{i+1}. So a' and a^* are not adjacent to any of $q_1, \ldots, q_{k-1}$. Let $Q' = q_1$-$\cdots$-q_k-a' and $Q^* = q_1$-$\cdots$-q_k-a'-a^*. Then Q' and Q^* are chordless paths. Lemma 2.4 implies that Q' has even length since none of its interior vertices are in $A \cup B$. It follows that Q^* is odd, which implies that $a^* <_A a'$, which contradicts the choice of a^*. This proves that every maximal vertex of $<_A$ is adjacent to all of K, and consequently that the vertex a of the algorithm exists. Conversely, let us prove that any such a is maximal for the relation $<_A$. Suppose the contrary. Then, by the definition of $<_A$, there exists an odd chordless path $Q'' = a$-a''-q-$\cdots$-b'' from a to a vertex $b'' \in B$ with $a'' \in A$. Then q is in $N(A)$ and on a chordless path from B, so q has been reached by the BFS defined on line 2, so q is in K. But then a is adjacent to q, which contradicts the fact that Q'' is chordless. So a is maximal for the relation $<_A$. The proof is similar for B: a vertex of B is maximal for the relation $<_B$ if and only if it is adjacent to all of L. Now Lemma 2.4 implies that the pair $\{a, b\}$ returned by the algorithm is a special even pair of G and the proof of correctness is complete. $\square$

Lemma 5.2 *The complexity of Algorithm Find_Even_Pair is $O(m)$.*

Proof. Determining the sets A and B takes time $O(d(v) + d(y))$ and $O(d(w) + d(x))$ respectively. Performing the breadth-first search from B to $N(A)$ and determining the set K takes time $O(m)$, and similarly for determining the set L. Moreover, each time a vertex is put into K we add $+1$ to a counter associated to each of its neighbours in A. And we do similarly for L and B. So finding vertices a and b takes time $O(|A|)$ and $O(|B|)$ respectively. $\square$

6 Analogy between interesting sets and handles

Recall that a graph is *weakly chordal* if G and its complementary graph contain no hole of length at least 5. A *handle* [4, 5] in a graph $G = (V, E)$ is a subset $H \subset V$, of size at least 2, such that $G[H]$ is connected, some component $J \neq H$

of $G \setminus N(H)$ satisfies $N(J) = N(H)$, and each vertex of $N(H)$ sees at least one vertex of each edge of $G[H]$. Any such J is called a *cohandle* of H. Hayward, Spinrad and Sritharan [6] use handles to obtain a recognition algorithm for weakly chordal graphs with complexity $O(m^2)$ and a coloring algorithm for those graphs with complexity $O(n^3)$. We observe that there is an analogy between handles and interesting sets.

Lemma 6.1 *Let H be a handle of G and J a co-handle of H. Then J is an interesting set of $\overline{G}$.*

Proof. By the definition of a handle, $G[J]$ is connected. Moreover, in the graph $\overline{G}$ we have $H \subseteq C(J)$, and H is connected and $|H| \geq 2$. So J is an interesting set in $\overline{G}$. □

Lemma 6.2 *Let T be a maximal interesting set in G, and let H be a co-connected component of $G[C(T)]$ of size at least 2. Then H is a handle of $\overline{G}$ and T is a co-handle of H.*

Proof. Since $C(T)$ is not a clique in G, there exists a component H of $\overline{G}[C(T)]$ of size at least 2. Let $X = N_{\overline{G}}(H)$. Since T is connected in $\overline{G}$, there is a component T' of $\overline{G} \setminus X$ that contains T. If $T \neq T'$, there is a vertex $u \in V \setminus (X \cup T)$ such that (in $\overline{G}$) u has a neighbour in T. Then (in G) $T \cup \{u\}$ is an interesting set because $T \cup \{u\}$ is co-connected and $H \subseteq C(T \cup \{u\})$. This contradicts the maximality of T. So T is a component of $\overline{G} \setminus X$. Now let $Y = N_{\overline{G}}(T)$. Clearly $Y \subseteq X$. In G every vertex x of X has a non-neighbour in H and thus is not in $T \cup C(T)$, and so $x \in Y$. Therefore $Y = X$. Finally, suppose that (in $\overline{G}$) some vertex $x \in X$ misses both vertices of an edge of $\overline{G}(H)$. Then (in G) the set $T \cup \{x\}$ is an interesting set strictly larger than T, a contradiction. So H is a handle and T is a co-handle of H in $\overline{G}$. □

The preceding two results show that any maximal interesting set of G gives a handle of $\overline{G}$, but a handle of $\overline{G}$ gives only an interesting set of G, which is not necessarily maximal. This suggests the following new definition which will strengthen the correspondence. A *generalized handle* is a subset $H \subset V$ that contains at least one edge, such that some component $J \neq H$ of $G \setminus N(H)$ satisfies $N(J) = N(H)$, and every vertex of $N(H)$ sees at least one vertex of each edge of $G[H]$. Any such J is called a *generalized co-handle* of H. This new definition of handles still enables us to use ideas from [6], where the hypothesis of connectedness of H does not seem to be necessary. A handle is a particular type of generalized handle, and the algorithm *find-handle* of [6] can be modified as follows:

Algorithm Find_Generalized_Handle

Search for a vertex v and an edge e such that v misses e
If no such v, e exist **then** return "no handle" and stop **endif**
$J \leftarrow$ component of $G \setminus N(e)$ containing v
$H \leftarrow V \setminus (J \cup N(J))$

9

while some v in $N(H)$ misses some e in H **do:**
$J \leftarrow$ component of $G \setminus N(e)$ containing v
$H \leftarrow V \setminus (J \cup N(J))$
endwhile
return (H, J)

Lemma 6.3 *The co-handle produced by this algorithm is a maximal interesting set in $\overline{G}$.*

Proof. By Lemma 6.1, J is an interesting set of $\overline{G}$. Suppose that J is not maximal. So there exists $j \notin J$ such that $J' = J \cup \{j\}$ is an interesting set of $\overline{G}$. Since $G[J']$ is connected, j is in $N(J)$. However, $N(J) = N(H)$ so j sees at least one vertex of each edge of H. Thus $V \setminus (J' \cup N(J')) \subset V \setminus (J \cap N(J)) = H$ and so $V \setminus (J' \cup N(J'))$ is a stable set and J' is not an interesting set, a contradiction. $\square$

Lemma 6.3 points to an alternative way to find a maximal interesting set. However, algorithms to find a handle so far [6] have not broken the complexity barrier that would make them better than the one we presented in Section 3.

References

[1] M.E. Bertschi, Perfectly contractile graphs. *J. Comb. Th. B* 50 (1990), 222–230.

[2] H. Everett, C.M.H. de Figueiredo, C. Linhares Sales, F. Maffray, O. Porto, B.A. Reed. Even pairs. In: *Perfect Graphs*, J.L. Ramírez-Alfonsín and B.A. Reed, eds., Wiley Interscience (2001), 67–92.

[3] J. Fonlupt, J.P. Uhry. Transformations which preserve perfectness and h-perfectness of graphs. *Ann. Disc. Math.* 16 (1982), 83–85.

[4] R. Hayward. Meyniel weakly triangulated graphs I: co-perfect orderability. Discrete Applied Mathematics 73 (1997), 199–210.

[5] R. Hayward. Meyniel weakly triangulated graphs II: A theorem of Dirac. Discrete Applied Mathematics 78 (1997), 283–289.

[6] R.B. Hayward, J.P. Spinrad, R. Sritharan. Weakly chordal graph algorithms via handles. Proc. 11th annual ACM-SIAM Symp. on Discrete Algorithms, 2000, 42–49.

[7] F. Maffray, N. Trotignon. A class of perfectly contractile graphs. Leibniz Research Report 67, Grenoble, France. Submitted for publication. http://www-leibniz.imag.fr/NEWLEIBNIZ/LesCahiers/2002/Cahier67.

[8] B.A. Reed. Problem session on parity problems (Public communication). *DIMACS Workshop on Perfect Graphs*, Princeton Univ., New Jersey, 1993.

[9] N. Trotignon. Graphes parfaits: structure et algorithmes. Doctoral Thesis, University Joseph Fourier, Grenoble, France, 2004.