Theoretical Computer Science 410 (2009) 2234–2240

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

# Coloring Artemis graphs

Benjamin Lévêque [a], Frédéric Maffray [b,*], Bruce Reed [c], Nicolas Trotignon [d]

[a] Laboratoire G-SCOP, 46 avenue Félix Viallet, 38031 Grenoble Cedex, France
[b] C.N.R.S., Laboratoire G-SCOP, 46 avenue Félix Viallet, 38031 Grenoble Cedex, France
[c] School of Computer Science, McGill University, 3480 University, Montreal, Quebec, Canada H3A 2A7
[d] Centre d'Économie de la Sorbonne, Université de Paris 1 Panthéon-Sorbonne, 106-112, Boulevard de l'Hôpital, 75647 Paris cedex 13, France

ARTICLE INFO

*Article history:*
Received 9 May 2007
Received in revised form 4 September 2008
Accepted 14 February 2009
Communicated by E. Pergola

*Keywords:*
Graph
Coloring
Perfect graph
Even pair
Algorithm

ABSTRACT

We consider the class 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 give an algorithm that can optimally color the vertices of these graphs in time $O(n^2m)$.



## 1. Introduction

A graph $G$ is an *Artemis graph* [6,21] if it contains no odd hole, no antihole of length at least five, and no prism; where a *hole* is a chordless cycle with at least four vertices, an *antihole* is the complement of a hole with at least five vertices, and a *prism* is a graph that consists of two vertex-disjoint triangles (cliques of size three) and three vertex-disjoint paths between them, with no other edge than those in the two triangles and in the three paths. The class $\mathcal{A}$ of Artemis graphs is contained in the class of perfect graphs defined by Berge [1,3]. Class $\mathcal{A}$ contains several classical families of perfect graphs, in particular *Meyniel graphs* [18,15,6], *perfectly orderable graphs* [5,6], *weakly chordal graphs* [9,6], and a few other classes (see [6]). We focus here on the question of finding an optimal coloring of the vertices of a graph in a given class. For a graph $G$, we denote by $\chi(G)$ the chromatic number of $G$ (i.e., the minimum number of colors in a coloring of the vertices of $G$) and by $\omega(G)$ the maximum clique size in $G$.

It is possible to color every perfect graph optimally and in polynomial time, thanks to the algorithm of Grötschel, Lovász, and Schrijver [8]. But that algorithm is based on the ellipsoid method and is generally considered impractical. So the quest for a simple and efficient algorithm to color optimally the vertices of every perfect graph remains a meaningful subject. There exist efficient algorithms to find the chromatic number of graphs in the classes mentioned above. Let $G$ be a graph with $n$ vertices and $m$ edges. Then the chromatic number of $G$ can be computed in time $O(n^2)$ if $G$ is a Meyniel graph [22], in time $O(n^3)$ if $G$ is a weakly chordal graph [13], and in time $O(n^2)$ if $G$ is a perfectly ordered graph (i.e., $G$ is perfectly orderable and a perfect ordering is given; however, finding such an ordering is NP-hard [19]). Here we show how to find an optimal coloring of $G$ in time $O(n^2m)$ if $G$ is in $\mathcal{A}$. Also it is the only known polynomial and combinatorial algorithm that can optimally color perfectly orderable graphs when the perfect order is not given.

* Corresponding author.
*E-mail addresses:* benjamin.leveque@g-scop.inpg.fr (B. Lévêque), frederic.maffray@g-scop.inpg.fr (F. Maffray), breed@cs.mcgill.ca (B. Reed), nicolas.trotignon@univ-paris1.fr (N. Trotignon).

0304-3975/$ – see front matter 
doi:10.1016/j.tcs.2009.02.012

An *even pair* in $G$ is a pair $\{x, y\}$ of non-adjacent vertices such 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 $x, y$; we denote by $G/xy$ the graph that results from this operation. The following lemma is due to Fonlupt and Uhry [7], see [6].

**Lemma 1.1** (*[6,7]*). *Let $G$ be any graph that contains an even pair $\{x, y\}$. Then, $\omega(G/xy) = \omega(G)$ and $\chi(G) = \chi(G/xy)$.* □

We will not repeat here the whole proof of this lemma, but two facts can be mentioned.
*Fact* (a): If $c$ is any coloring of $G/xy$, then there is a coloring of $G$, using the same number of colors, where $x$ and $y$ receive the same color (the color assigned by $c$ to the contracted vertex), all other colors remain unchanged.
*Fact* (b): If $C$ is a clique in $G/xy$, then either the vertex obtained by contracting $x, y$ is not in $C$, and then $C$ is also a clique in $G$, or the contracted vertex is in $C$, and replacing it by one of $x$ or $y$ gives a clique in $G$ of the same size as $C$.

Fact (a) 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* [2] if it can be turned into a clique by a sequence of contractions of even pairs, and a graph is called *perfectly contractile* if every induced subgraph of $G$ is even-contractile. Everett and Reed [6,21] conjectured that every graph in $\mathcal{A}$ is perfectly contractile. Maffray and Trotignon [16] proved that conjecture by showing that every graph $G$ in $\mathcal{A}$ admits an even pair $\{x, y\}$ such that $G/xy$ is in $\mathcal{A}$. From their proof they derive an algorithm that finds an optimal coloring of the vertices of any graph $G$ in $\mathcal{A}$ in time $O(n^6)$. Here we show that this algorithm can be implemented to run in time $O(n^4)$.

Because of Fact (b), if you can contract a graph into a clique by a sequence of even pair contractions then you can obtain a clique of maximum size of the original graph by decontracting the even pairs.

## 2. The method

Our main algorithm follows the method from [16,23], which focused on proving the above-mentioned conjecture of Everett and Reed. (Incidentally, ideas from [16] have been used recently by Chudnovsky and Seymour [4] to find a substantial shortcut in the proof of the strong perfect graph theorem [3].) Here each step has been simplified to improve the total complexity compared with that in [16].

An even pair $\{a, b\}$ in a graph $G$ is called *special* if the graph $G/ab$ contains no prism.

**Lemma 2.1** (*[6,16]*). *If $G$ is in $\mathcal{A}$ and $\{a, b\}$ is a special even pair of $G$, then $G/ab$ is in $\mathcal{A}$.*

The algorithm from [16] 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. Let us now recall how a special even pair is found.

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 are adjacent to at least one vertex of $X$. A vertex of $V \setminus X$ is called *$X$-complete* if it is adjacent to 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.

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 of length at least two whose two end vertices 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 a proper subset of 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** (*[16]*). *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 (i.e., a graph whose components are cliques).*
*Moreover, if $G$ is not a disjoint union of cliques then every non-simplicial vertex forms an interesting set.*

**Lemma 2.3** (*[16]*). *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:

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

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** ([16]). *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 partial order on $A$. The relation $<_B$ is a 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 $G$.*

The proof of the next lemma is easy and we omit it.

**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 $G/ab$.*

We find a special even pair as follows. On the basis of Lemma 2.2, 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.

It is important to remark that all operations performed by the algorithms below can be done on any input graph, so that the complete algorithm always ends up with some pair of non-adjacent vertices for any input graph that is not a clique. Indeed, we use the fact that the input graph is Artemis only in the proof of correctness, never in the description of the algorithm nor in the complexity analysis. In general the pair produced by the algorithm is not necessarily an even pair, but it is an even pair whenever the input graph is Artemis. This remark is essential to Section 7.

## 3. Looking for a maximal interesting set

The proof of the next lemma is easy and we omit it.

**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.*

**Algorithm Find_Interesting_Set**
*Input:* An Artemis graph $G$.
*Output:* Either a maximal interesting set $T$ of $G$ or the answer "$G$ is a disjoint union of cliques".

1. Compute the components of $G$.
2. **If** every vertex has degree equal to the size of its component minus 1, **then** return the answer "$G$ is a disjoint union of cliques" and stop. **endif**
3. 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.
4. Set $T := \{t\}$, $C := N(t)$, $U := V \setminus (T \cup C)$, $Z := \emptyset$.
5. **while** there exists a vertex $u \in U$ **do**:
   **If** $N(u) \cap C$ is a clique, **then** move $u$ from $U$ to $Z$, **else** move $u$ from $U$ to $T$ and move every vertex of $C \setminus N(u)$ from $C$ to $U$. **endif**
   **endwhile**
6. Return the set $T$ and stop.

**Lemma 3.2.** *Algorithm Find_Interesting_Set is correct.*

**Proof.** Clearly, Step 2 of the algorithm is correct, and if the algorithm reaches Step 3, then there exists a vertex at distance 2 from $u$, and vertex $t$ found in Step 3 is not simplicial (because $u$, $v$ are non-adjacent neighbours of $t$). At the beginning of Step 4 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 5 implies that these properties remain true throughout, and Lemma 3.1 ensures that when Step 5 terminates the set $T$ is a maximal interesting set. □

**Lemma 3.3.** *The complexity of Algorithm Find_Interesting_Set 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.** Steps 1 to 4 take time $O(n + m)$ steps. In Step 5, 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 $U \cup C$ 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. □

## 4. Looking for an outer path

The proof of the next lemma is easy and we omit it.

**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.*

**Lemma 4.2** ([16]). *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:* An Artemis 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".

1. All vertices of $V(G) \setminus (T \cup C(T))$ are unmarked.
2. **While** there is an unmarked vertex $r$ in $V(G) \setminus (T \cup C(T))$
   **do**:
   Start a Breadth-First Search from $r$ to $C(T)$ in $G \setminus T$. Call $Q$ the queue of this search and $S$ the set of vertices reached by the search. Mark each vertex of $S \setminus C(T)$, and maintain the set $M = S \cap C(T)$. Whenever a vertex $x$ is added to $M$, **if** the new $M$ is not a clique, **then** do:
   Let $M_x := M \cap N(x)$. Perform a BFS from $x$ in the subgraph $G[S \setminus (Q \cup M_x)]$. 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**
3. Return the answer "there is 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 "there is no $T$-outer path" if and only if $G$ has no $T$-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 $S \setminus (Q \cup M)$ has been scanned before the parent of $x$.

The search from $x$ in $G[S \setminus (Q \cup M_x)]$ will potentially reach all vertices of $M \setminus M_x$. So the vertex $y$ and the path $x$-$v$-$\cdots$-$w$-$y$ exist. Let us rewrite this path as $P = x$-$z_1$-$\cdots$-$z_p$-$y$, with $z_1 = v$ and $z_p = w$, and write $Z = \{z_1, \ldots, z_p\}$. 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 $x'$-$z_i$-$\cdots$-$z_j$-$y'$ 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 $x$-$z_1$-$\cdots$-$z_j$-$y'$-$x$ is a hole of odd length $j - 1 + 3 \geq 5$, a contradiction. So $x$ is not adjacent to $y'$, but then $x$-$z_1$-$\cdots$-$z_j$-$y'$ is a chordless path and $y'$ contradicts the definition of $y$. So $P$ is a minimal $T$-outer path. □

**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 $R \cup C(T)$. However, a vertex $u$ of $C(T)$ may be scanned several times since it may belong to $N(R) \cap C(T)$ for several $R$'s. So the total complexity is $O(lm)$. □

## 5. Looking for a special even pair

We assume for now that we have a maximal interesting set $T$ and a minimal $T$-outer path given by the algorithms presented in the preceding sections.

**Algorithm Find_Even_Pair**

*Input:* An Artemis 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 $G$

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 is adjacent to all of $K$, and let $b$ be a vertex of $B$ that is adjacent to all of $L$.
5. Return the pair $\{a, b\}$.

**Lemma 5.1.** *Algorithm Find_Even_Pair is correct.*

**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 $A' = \{u \in C(T) \mid uz_1 \in E(G), uz_i \notin E(G) (i = 2, \ldots, p)\}$ and $B' = \{u \in C(T) \mid uz_p \in E(G), uz_i \notin E(G) (i = 1, \ldots, p-1)\}$. 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. □

**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. □

## 6. The algorithm

**Algorithm Contract_Artemis**
*Input:* An Artemis graph $G$.
*Output:* A clique obtained from $G$ by a sequence of even pair contractions.
**While** $G$ is not a disjoint union of cliques, do:

1. Run Find_Interesting_Set on $G$, and call $T$ the output.
2. **While** $C(T)$ is not a clique, **do**:
   – Run Find_Outer_Path on $T$.
   – **If** the output is a path $P$, **then** run Find_Even_Pair on $T$ and $P$ and contract the output pair.
   **Else** run Contract_Artemis on $C(T)$. **endif**
   **endwhile**

**endwhile**
Now $G$ is not a disjoint union of cliques. Contract them into the largest of them.

To color an Artemis graph $G$, we apply the contracting algorithm. Note that each vertex of the output clique represents a stable set in $G$, and these stable sets form a partition of $V(G)$. This partition is the coloring produced by the coloring algorithm.

**Lemma 6.1.** *Algorithm Contract_Artemis is correct.*

**Proof.** Suppose that $G$ is not a disjoint union of cliques. By Lemma 2.2, $G$ has an interesting set, and by Lemma 3.2, Algorithm Find_Interesting_Set will find a maximal interesting set. By Lemma 4.3, Algorithm Find_Outer_Path will find an outer path if any exists. When there is a $T$-outer path, Lemma 5.1 ensures that Algorithm Find_Even_Pair will find a special even pair, and Lemma 2.1 ensures that the contracted graph is still in class $\mathcal{A}$. When there is no $T$-outer path, Lemma 2.3 ensures that we may continue the search in the subgraph induced by $C(T)$. □

**Lemma 6.2.** *The complexity of Algorithm Contract_Artemis is $O(n^2m)$.*

**Proof.** Let us examine the complexity of Step 2. When there is no $T$-outer path, we continue the search for an even pair 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. 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 $\ell_i$ 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 $\ell_1 + \cdots + \ell_q \leq n$. By Lemma 4.4, the total complexity, over all recursive calls, of finding outer paths is $O(\Sigma_{i=1}^{q} \ell_i m) = O(nm)$. So the total complexity of Step 2 is $O(nm)$. Since Step 2 always results in the contraction of an even pair, there are at most $n$ instances of Step 2. Thus the total complexity of the algorithm is $O(n^2 m)$. □

## 7. Maximum clique and robustness

For every algorithm that colors a graph $G$ by contracting even pairs until a clique $K$ is obtained, it is easy to also obtain a clique of maximum size of the graph $G$. Indeed, as observed in the last paragraph of the Introduction, we can "decontract" the even pairs in reverse order, starting from $|K|$, until we obtain a clique of $G$ of size $|K|$. Since this size is the number of colors used by the algorithm, we have a simple certificate of optimality of the algorithm.

It is easy to see that all operations performed in Algorithm Contract_Artemis can be performed on any input (not necessarily Artemis) graph. Thus the algorithm will output a clique $K$, which in general is not necessarily obtained by even pair contractions. We can decontract this clique as in the paragraph above. If we obtain a clique of $G$ of size $|K|$, then as above, this proves that the coloring is optimal (it might be that the input graph is not Artemis). Else, we can conclude that the graph is not Artemis. This means that the coloring algorithm is a "robust" algorithm [20] in the sense that, for every graph $G$, it will either return a coloring and a clique of the same size or the answer "$G$ is not in $\mathcal{A}$". The fact that we do not have to test beforehand if $G$ is in $\mathcal{A}$ is interesting because the only known algorithm [17] that decides if a graph is in $\mathcal{A}$ has complexity $O(n^9)$.

## 8. Analogy between interesting sets and handles

Recall that a graph is *weakly chordal* if the graph and its complementary graph contain no hole of length at least 5. Clearly, a weakly chordal graph is an Artemis graph. The method presented here to color Artemis graph can be seen as a generalization of the method of Hayward, Spinrad and Sritharan [12,13] to color weakly chordal graphs. The difference is that, in a weakly chordal graph, a maximal interesting set has no outer path (see [16, Lemma 2.8]). So, if we apply our algorithms on a weakly chordal graph, we never call algorithms *Find_Outer_Path* and *Find_Even_Pair*, and the recursive search for maximal interesting sets always leads to a disjoint union of cliques. The following lemmas explain the analogy.

A *handle* [10,11] 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 $N(H)$ is adjacent to at least one vertex of each edge of $G[H]$. Any such $J$ is called a *co-handle* of $H$. Hayward, Spinrad and Sritharan [12,13] 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 $O(n^3)$.

**Lemma 8.1.** *Let $G$ be any graph. Then:*

- *If $H$ is a handle of $G$ and $J$ a co-handle of $H$, then $J$ is an interesting set of $\overline{G}$.*
- *If $T$ is a maximal interesting set in $G$, and $H$ is 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.** First suppose that $H$ is a handle of $G$ and $J$ a co-handle of $H$. By the definition of a handle, $G[J]$ is connected. Moreover, $H$ is connected, $|H| \geq 2$, and in the graph $\overline{G}$ we have $H \subseteq C(J)$. So $J$ is an interesting set in $\overline{G}$, which proves the first item of the lemma.

Now we consider the second item. 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 $x \in X$ is not adjacent to any of the two 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 lemma shows 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 $N(H)$ is adjacent to 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 [12,13], 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 [12,13] can be modified as follows:

**Algorithm Find_Generalized_Handle**

1. Search for a vertex $v$ and an edge $e$ such that $v$ is not adjacent to $e$;
2. **If** no such $v$, $e$ exist **then** return "no handle" and stop **endif**
3. $J :=$ the component of $G \setminus N(e)$ that contains $v$ and $H := V \setminus (J \cup N(J))$;
4. **while** some $v$ in $N(H)$ is not adjacent to some $e$ in $H$ **do:**
   $J :=$ the component of $G \setminus N(e)$ that contains $v$ and $H := V \setminus (J \cup N(J))$
   **endwhile**
5. return $(H, J)$.

The proof of correctness of this algorithm is just like the proof of correctness of *find_handle* [10,11] and we omit it.

**Lemma 8.2.** *The co-handle produced by this algorithm is a maximal interesting set in* $\overline{G}$.

**Proof.** By Lemma 8.1, $J$ 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 $j$ is adjacent to at least one vertex of each edge of $H$. Thus $V \setminus (J' \cup N(J')) \subset V \setminus (J \cup N(J)) = H$ and so $V \setminus (J' \cup N(J'))$ is a stable set and $J'$ is not an interesting set, a contradiction. □

Lemma 8.2 points to an alternative way to find a maximal interesting set. However, algorithms to find a handle [12,13] so far have not broken the complexity barrier that would make them better than the one we presented in Section 3.

## 9. Conclusion

We have presented here a fast algorithm to color the vertices of an Artemis graph. This algorithm does not solve the weighted version of the coloring problem. Another open problem is to find a maximum stable set and a minimum partition into cliques for an Artemis graph, which would be useful for solving some precoloring extension problems as mentioned in [14].

## Acknowledgement

The first author was supported by Ecole Normale Supérieure de Lyon.

## References

[1] C. Berge, Les problèmes de coloration en théorie des graphes, Publ. Inst. Stat. Univ. Paris 9 (1960) 123–160.
[2] M.E. Bertschi, Perfectly contractile graphs, J. Combin. Theory Ser. B 50 (1990) 222–230.
[3] M. Chudnovsky, N. Robertson, P. Seymour, R. Thomas, The strong perfect graph theorem, Ann. Math. 164 (2006) 51–229.
[4] M. Chudnovsky, P. Seymour, Even pairs in Berge graphs, J. Combin. Theory Ser. B 99 (2009) 370–377.
[5] V. Chvátal,Chvátal, Perfectly ordered graphs, in: C. Berge (Ed.), Topics on Perfect Graphs, in: Ann. Disc. Math., vol. 21, North Holland, Amsterdam, 1984, pp. 63–68.
[6] H. Everett, C.M.H. de Figueiredo, C. Linhares Sales, F. Maffray, O. Porto, B.A. Reed, Even pairs, in: J.L. Ramírez-Alfonsín, B.A. Reed (Eds.), Perfect Graphs, Wiley Interscience, 2001, pp. 67–92.
[7] J. Fonlupt, J.P. Uhry, Transformations which preserve perfectness and $h$-perfectness of graphs, Ann. Discrete Math. 16 (1982) 83–85.
[8] M. Grötschel, L. Lovász, A. Schrijver, Polynomial algorithms for perfect graphs, in: Topics on Perfect Graphs, in: North-Holland Math. Stud., vol. 88, North-Holland, Amsterdam, New York, 1984, pp. 325–356.
[9] R.B Hayward, Weakly triangulated graphs, J. Combin. Theory Ser. B 39 (1985) 200–208.
[10] R.B. Hayward, Meyniel weakly triangulated graphs I: Co-perfect orderability, Discrete Appl. Math. 73 (1997) 199–210.
[11] R.B. Hayward, Meyniel weakly triangulated graphs II: A theorem of Dirac, Discrete Appl. Math. 78 (1997) 283–289.
[12] R.B. Hayward, J.P. Spinrad, R. Sritharan, Weakly chordal graph algorithms via handles, in: Proc.11th Annual ACM–SIAM Symp. on Discrete Algorithms, 2000, pp. 42–49.
[13] R.B. Hayward, J.P. Spinrad, R. Sritharan, Improved algorithms for weakly chordal graphs, ACM Trans. Algorithms 3 (2) (2007) article 14.
[14] V. Jost, B. Lévêque, F. Maffray, Precoloring extension of co-Meyniel graphs, Graphs Combin. 23 (2007) 291–301.
[15] I.A. Karapetian, S.E. Markossian, Perfect graphs, Akad. Nauk Armenii Dokl. 63 (1976) 292–296. (in Russian).
[16] F. Maffray, N. Trotignon, A class of perfectly contractile graphs, J. Combin. Theory Ser. B 96 (2006) 1–19.
[17] F. Maffray, N. Trotignon, Algorithms for perfectly contractile graphs, SIAM J. Discrete Math. 19 (2005) 553–574.
[18] H. Meyniel, On the perfect graph conjecture, Discrete Math. 16 (1976) 339–342.
[19] M. Middendorf, F. Pfeiffer, On the complexity of recognizing perfectly orderable graphs, Discrete Math. 80 (1990) 327–333.
[20] V. Raghavan, J. Spinrad, Robust algorithms for restricted domains, J. Algorithms 48 (2003) 160–172.
[21] B.A. Reed, Problem session on parity problems (Public communication), in: DIMACS Workshop on Perfect Graphs, Princeton Univ., New Jersey, 1993.
[22] F. Roussel, I. Rusu, An $O(n^2)$ algorithm to color Meyniel graphs, Discrete Math. 235 (2001) 107–123.
[23] N. Trotignon, Graphes parfaits: Structure et algorithmes. Doctoral Thesis, University Joseph Fourier, Grenoble, France, 2004.