Recognizing line-polar bipartite graphs in time $O(n)$

Tınaz Ekim$^{a,1}$, Jing Huang$^{b,*2}$

$^a$ Bogazici University, Industrial Engineering Department, 34342, Bebek-Istanbul, Turkey
$^b$ Department of Mathematics and Statistics, University of Victoria, P.O. Box 3060 STN CSC, Victoria, B.C., V8W 3R4, Canada

A graph is polar if the vertex set can be partitioned into $A$ and $B$ such that $A$ induces a complete multipartite graph and $B$ induces a disjoint union of cliques (i.e., the complement of a complete multipartite graph). Polar graphs naturally generalize several classes of graphs such as bipartite graphs, cobipartite graphs and split graphs. Recognizing polar graphs is an NP-complete problem in general, and thus it is of interest to restrict the problem to special classes of graphs. Cographs and chordal graphs are among those whose polarity can be recognized in polynomial time. The line-graphs of bipartite graphs are another class of graphs whose polarity has been characterized recently in terms of forbidden subgraphs, but no polynomial time algorithm is given. In this paper, we present an $O(n)$ algorithm which decides whether the line-graph of an input bipartite graph is polar and constructs a polar partition when one exists.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

We follow the standard terminology and notation from [9] and consider only simple graphs (i.e., those containing no loops or multiple edges). In particular, we use $P_k$ and $C_k$ to denote the path and cycle with $k$ vertices, respectively. The degree of a vertex $x$ in a graph is denoted by $d(x)$.

A graph $G$ is polar if the vertex set can be partitioned into $A$ and $B$ such that the subgraph induced by $A$ is a complete multipartite graph and the subgraph induced by $B$ is a disjoint union of cliques. Such a partition $(A, B)$ is called a polar partition of $G$. Polar graphs are a common generalization of several classes of graphs. For instance, every bipartite graph is polar as a bipartition is a polar partition of the graph. Split graphs form another subclass of polar graphs. These are the graphs which admit polar partitions $(A, B)$ where $A$ induces an independent set and $B$ induces a clique; see [7].

Foldes and Hammer [7] proved that a graph is split if and only if it does not contain $2K_2$, $C_4$ or $C_5$ as an induced subgraph. Consequently, split graphs can be recognized in polynomial time. In general, to determine whether a graph is polar is an NP-complete problem; see [1]. Ekim et al. [6] studied the polarity among cographs (i.e., graphs containing no induced $P_4$). They showed that there are only finitely many forbidden subgraphs for polar cographs and hence the polarity of cographs is recognizable in polynomial time. In contrast, polar chordal graphs (i.e., graphs with no induced cycles of length $\geq 4$) cannot be characterized in terms of finitely many forbidden subgraphs. Nevertheless, a polynomial time recognition algorithm has been devised for polar chordal graphs; see [5]. The polarity of permutation graphs has also been studied in [4].

Given a graph $G = (V, E)$, the line graph $L(G)$ of $G$ has vertex set $E$ and two vertices are adjacent in $L(G)$ if and only if the two corresponding edges are adjacent (i.e., having a common endvertex) in $G$. Call a graph $G$ line-polar if $L(G)$ is polar and call...
a polar partition of $L(G)$ a line-polar partition of $G$. Thus a line-polar partition of $G$ is a partition of the edge set of $G$. Although bipartite graphs are all polar, the characterization of line-polar bipartite graphs in [8] demonstrates several infinite families of forbidden subgraphs.

Polar partitions of certain classes of graphs take only special forms. In particular, a polar partition $(A, B)$ of a graph $G$ is called monopolar if $A$ is an independent set. Graphs that admit monopolar partitions are called monopolar. Similarly, a graph $G$ is called line-monopolar if $G$ has a line-polar partition $(S, T)$ where $S$ is a matching of $G$; such $(S, T)$ is called a line-monopolar partition of $G$.

For a graph $G = (V, E)$ and a set $R \subseteq E$, we shall use $R$ (when no confusion arises) to denote the subgraph of $G$ induced by $R$ (that is, having a vertex set consisting of all end vertices of edges in $R$ and edge set $R$). For instance, we say that $R$ is a star if $R$ consists of the edges in a star and similarly that $R$ is a disjoint union of stars if $R$ consists of the edges in a vertex-disjoint union of stars. In general, the subgraph induced by $R$ is not necessarily an induced subgraph of $G$ (e.g., any three edges in $C_4$ induce a $P_4$).

**Proposition 1.1** ([8], Let $G$ be a bipartite graph. Suppose that $(S, T)$ is a line-polar partition of $G$. Then $S$ is either a matching, or a star, or a $P_4$, or a $C_4$, and $T$ is a disjoint union of stars (or equivalently, $T$ contains no $P_4$).

It is easy to see that in polynomial time one can decide whether a graph $G$ has a line-polar partition $(S, T)$ where $S$ is either a star, or a $P_4$, or a $C_4$, and $T$ is a disjoint union of stars in $G$. For bipartite graphs, we will show that this can be done in time $O(n^2)$, where $n$ is the number of vertices of the input bipartite graph. Thus the existence of a polynomial time recognition algorithm for the line-polarity of bipartite graphs depends on a polynomial time recognition algorithm for the line-monopolarity of bipartite graphs.

We present an $O(n)$ algorithm for recognizing line-monopolar bipartite graphs. Hence the line-polarity of bipartite graphs is recognizable in time $O(n)$. Our algorithm makes use of the structural properties of line-monopolar bipartite graphs developed in [8]. It works directly on the input bipartite graph and returns a line-polar partition when the graph is line-polar.

We note that an $O(n \log n)$ recognition algorithm has been devised recently in [3] for the monopolarity of claw-free graphs and hence for the line-monopolarity of bipartite graphs since every line-graph is claw-free. However, our algorithm is much simpler and more efficient than the one given in [3].

Throughout the paper, when we say $G$ contains $H$, it means that $H$ is a (not necessarily induced) subgraph of $G$, and to emphasize the case when $H$ is an induced subgraph of $G$, we say that $G$ contains $H$ as an induced subgraph. It is a simple fact that if $G$ contains $H$ then $L(G)$ contains $L(H)$ as an induced subgraph.

2. Algorithm

Let $G = (V, E)$ be the (input) bipartite graph with $n$ vertices. Suppose that $(S, T)$ is a line-polar partition of $G$. Then $T$ is a disjoint union of stars and hence at most $n - 1$ edges. By Proposition 1.1, $S$ is either a matching with at most $n/2$ edges, or a star with at most $n - 1$ edges, or a $P_4$ with three edges, or a $C_4$ with four edges. It follows that the number of edges in $G$ is at most $2n - 2$. We can check in time $O(n)$ whether $G$ has at most $2n - 2$ edges. When $G$ has more than $2n - 2$ edges, we simply report that $G$ is not line-polar. So we may assume that $G$ has at most $2n - 2$ edges and express the complexity of our algorithm purely in terms of the number $n$.

To check if $G$ has a line-polar partition $(S, T)$ where $S$ is a star, we compute the components of $G$. If there are two or more components, each of which contains two or more vertices of degree $\geq 2$, then $G$ has no desired partition. On the other hand, if no component has two or more vertices of degree $\geq 2$, then $G$ has a desired partition $(S, T)$ with $S = \emptyset$. So we may assume that $F$ is the only component which contains two or more vertices of degree $\geq 2$. We can determine whether $F$ is a tree by comparing the number of edges with the number of vertices in $F$. In the case when $F$ is a tree, we find a longest path $P : v_1v_2\cdots v_r$ in $F$. If $r \geq 7$, then $G$ does not have a desired partition. Otherwise $r \leq 7$ and a desired partition $(S, T)$ (if one exists) has the star $S$ centered at one of the three vertices $v_3, v_5,$ and $v_6$. Thus we check whether $G - v_i$ is a disjoint union of stars for $i = 3, 4, 5$. In the case when $F$ is not a tree, find any cycle $C$ in $F$. If $C$ has six or more vertices, then $G$ has no desired partition. So $C$ has exactly four vertices, and a desired partition $(S, T)$ (if one exists) has the star $S$ centered at one of the four vertices of $C$. We check whether $G - x$ is a disjoint union of stars for each of the four vertices $x$ of $C$. Since $G$ has at most $2n - 2$ edges, all these can be done in time $O(n)$.

**Lemma 2.1.** Suppose that $G$ has a line-polar partition $(S, T)$ such that $S$ is a $P_4$. Then either $S$ can be chosen to be such a $P_4$: $abcd$ where each of $a$, $b$, $c$, $d$ is of degree $\geq 2$ or $G$ admits a line-polar partition $(S', T')$ such that $S'$ is a star.

**Proof.** Let $(S, T)$ be a line-polar partition of $G$ where $S$ is a $P_4$: $abcd$. If the degrees of $a, d$ are at least 2, then we are done. So assume that at least one of $a, d$ is of degree 1. Suppose that the degree of $a$ is 1 and the degree of $d$ is at least 2. Note that $b$ is adjacent to at most two vertices of degree $\geq 2$ (including $c$). If there is a vertex $u$ not in the $P_4$ which is of degree $\geq 2$ and adjacent to $b$, then we replace $S$ by the new $P_4$: $ubcd$ whose four vertices are of degree at least 2 and obtain a new line-polar partition of $G$. On the other hand, if such a $u$ does not exist, then we replace $S$ by the star centered at $c$ and obtain a line-polar partition of $G$. A similar argument applies when the degree of $d$ is 1 and the degree of $a$ is at least 2. Suppose now that both $a, d$ are of degree 1. Again if there is a vertex $u$ not in the $P_4$ which is of degree $\geq 2$ and adjacent to $b$ and there is a vertex $v$ not in the $P_4$ which of degree at least 2 and adjacent to $c$, then we replace $S$ by the new $P_4 : ubcv$ and obtain a line-polar partition $$(S', T')$$ such that $S'$ is a star.
Lemma 2.1. Let $u$, $v$ be such that $d(u) = d(v) = 3$. If such a $u$ or such a $v$ does not exist, then we replace $S$ by the star centered at $c$ or the star centered at $b$ and obtain a line-polar partition of $G$. \hfill $\blacksquare$

In view of Lemma 2.1, to check whether $G$ has a line-polar partition $(S, T)$ such that $S$ is either a $C_4$: $abcda$ or a $P_4$: $abcd$, we may focus on the case when all four vertices $a$, $b$, $c$, $d$ are of degree at least 2. We compute the components of $G$ and check whether there is a unique component which contains two or more vertices of degree $\geq 2$; otherwise $G$ does not have a desired partition. So let $F$ be the only component of $G$ which contains at least two vertices of degree $\geq 2$. If the number of vertices of degree $\geq 2$ in $F$ is less than four or greater than eight, then we simply report that $G$ does not have a desired partition. The reason that $F$ can have at most eight vertices of degree $\geq 2$ is that every vertex in $F - \{a, b, c, d\}$ of degree $\geq 2$ must have a neighbour in $\{a, b, c, d\}$ and no two of them have a common neighbour in $\{a, b, c, d\}$. Therefore, to check whether $G$ has a desired partition, we examine each set of four vertices of degree $\geq 2$ in $F$ (at most $\left(\binom{8}{2}\right) = 70$ such sets) to see if it induces either a $C_4$ or a $P_4$ and can be used as the set $S$ in a line-polar partition $(S, T)$ of $G$. All these can be done in time $O(n)$.

Summarizing, we have the following.

Proposition 2.2. Given a bipartite graph $G$, one can decide in time $O(n)$ if $G$ has a line-polar partition $(S, T)$ such that $S$ is either a star, or a $C_4$, or a $P_4$, and find such a partition if it exists. \hfill $\blacksquare$

By Proposition 2.2, it remains to determine whether $G$ is line-monopolar.

Proposition 2.3 ([8]). If a bipartite graph $G$ contains any graph in Fig. 1 as a subgraph (not necessarily induced), then $G$ is not line-monopolar. \hfill $\blacksquare$

In fact, it is shown in [8] that if a bipartite graph is line-polar but not line-monopolar, then it must contain a graph in Fig. 1 as a subgraph.

Our algorithm will try to construct a line-monopolar partition $(S, T)$ of $G$. Initially, both $S$ and $T$ are empty. Edges of $G$ will be added one by one either to $S$ or to $T$ in such a way that if $G$ is line-monopolar then there is a line-monopolar partition $(S^*, T^*)$ of $G$ with $S \subseteq S^*$ and $T \subseteq T^*$. Such a pair $(S, T)$ will be called valid. If a valid pair $(S, T)$ contains a conflict, that is, either a $P_3$ in $S$, or a $P_4$ in $T$, then $G$ is not line-monopolar. Thus our algorithm will also check possible conflicts in $(S, T)$: if a conflict is found, then it stops and returns ‘$G$ is not line-monopolar’; otherwise, it will continue adding edges to $S$ or $T$ until $(S, T)$ contains all edges of $G$, in which case it is a line-monopolar partition of $G$.

The following two propositions explain how we may begin to construct $(S, T)$.

Proposition 2.4. If $uv$ is an edge of $G$ such that $d(u) \geq 3$ and $d(v) \geq 3$, then $uv \in S$ for every line-monopolar partition $(S, T)$ of $G$.

Proof. Let $u_1$, $u_2$ be two neighbours of $u$ distinct from $v$ and let $v_1$, $v_2$ be two neighbours of $v$ distinct from $u$. Since $S$ is a matching in $G$, at least one of $uu_1$, $uu_2$ is not in $S$ and hence in $T$. Similarly, at least one of $vv_1$, $vv_2$ is in $T$. Hence $T$ cannot contain $uv$, as otherwise $T$ contains a $P_4$. Therefore $S$ must contain $uv$. \hfill $\blacksquare$

Our algorithm consists of three parts. The first part is called Preprocessing. Preprocessing will add those edges to $S$ suggested by Proposition 2.4. That is, it examines each edge $uv$, and adds it to $S$ if both $d(u) \geq 3$ and $d(v) \geq 3$.

Proposition 2.5. Suppose that $(S, T)$ is obtained by Preprocessing. If $(S, T)$ contains a conflict, then $G$ is not line-monopolar. If $G$ is line-monopolar, then any line-monopolar partition $(S^*, T^*)$ of $G$ satisfies $S^* \subseteq S^*$ and $T \subseteq T^*$.

Proof. When Preprocessing is complete, although some edges may be added to $S$, the set $T$ remains empty. If $(S, T)$ contains a conflict at this point, then $G$ contains Fig. 1(i), (ii) or (v) as a subgraph, and hence is not line-monopolar, by Proposition 2.3. On the other hand, by Proposition 2.4, any line-monopolar partition $(S^*, T^*)$ of $G$ satisfies $S^* \subseteq S^*$ and $T \subseteq T^*$, i.e., the pair $(S, T)$ is valid. \hfill $\blacksquare$

![Fig. 1. Some obstructions for being line-monopolar.](image-url)
We may assume that there is no conflict in $(S, T)$ when Preprocessing is complete. Our algorithm proceeds with the second part, called Propagation. Propagation consists of Propagation $(S)$ and Propagation $(T)$. In Propagation $(S)$, we process edges which are newly added in $S$. To process an edge $uv$ of $S$, we add to $T$ all edges which are incident with either $u$ or $v$ but not in $S \cup T$.

In Propagation $(T)$, we process edges which are newly added in $T$. Let $vw$ be such an edge in $T$. To process $vw$, we do the following. If either $d(v) \geq 3$ or $v$ is incident with another edge $vw$ in $T$, then add to $T$ edges which are incident with $w$ but not in $S \cup T$. (We will show that there can be at most one such edge or else $G$ is not line-monopolar.) If $d(v) = 2$ and $v$ is incident with an edge $uv$ in $S$, then add each $uw \not\in S \cup T$ to $T$ if $d(x) \geq 3$.

Note that Propagation $(S)$ only adds edges to $T$ and Propagation $(T)$ only adds edges to $S$. A justification of Propagation is explained in the following proposition.

**Proposition 2.6.** Let $(S', T')$ be the pair obtained from $(S, T)$ by applying Propagation $(S)$ and Propagation $(T)$. If $(S, T)$ is valid, then so is $(S', T')$.

**Proof.** Suppose that $G$ is line-monopolar. Since $(S, T)$ is valid, there is a line-monopolar partition $(S^*, T^*)$ with $S \subseteq S^*$ and $T \subseteq T^*$. Since $(S^*, T^*)$ is a line-monopolar partition of $G$, for every edge $uv \in S$, all edges of $G$ sharing a vertex with $uv$ must be in $T^*$. Thus edges added to $T$ by Propagation $(S)$ are all in $T^*$.

Let $uv$ be an edge in $T$. Suppose that either $d(v) \geq 3$ or $v$ is incident with another edge in $T$. Then $T^*$ contains at least two edges incident with $v$. So $d(w) \leq 2$ and the only possible edge incident with $w$ but not in $T$ has to be in $S^*$. Suppose that $v$ is incident with an edge in $S$ (and hence in $S^*$). If $uw$ is an edge not in $S \cup T$ and $d(x) \geq 3$, then $uw$ must be in $S^*$ (as otherwise $T^*$ contains $P_4$ induced by $vw$, $uw$ and another edge incident with $x$, a contradiction). Hence all edges added to $S$ by Propagation $(T)$ are in $S^*$.

The proof of Proposition 2.6 shows that the same line-monopolar partition $(S^*, T^*)$ which satisfies $S \subseteq S^*$ and $T \subseteq T^*$ also satisfies $S' \subseteq S^*$ and $T' \subseteq T^*$.

Proposition causes some edges to be added to $S$ and $T$. Thus we can recursively apply Propagation to newly added edges until all edges in $S \cup T$ are processed. By Propositions 2.5 and 2.6 and the above remarks, we conclude the following.

**Proposition 2.7.** Suppose that $(S, T)$ is the pair obtained after recursive applications of Propagation. If $(S, T)$ contains a conflict, then $G$ is not line-monopolar. On the other hand, if $(S^*, T^*)$ is a line-monopolar partition of $G$, then $S \subseteq S^*$ and $T \subseteq T^*$. □

Suppose that $(S, T)$ is obtained when all edges in $S \cup T$ are processed by Propagation. Again we may assume that $(S, T)$ contains no conflict, as otherwise $G$ is not line-monopolar according to Proposition 2.7. If $S \cup T$ contains all edges of $G$, then $(S, T)$ is a line-monopolar partition of $G$. So we may further assume that $S \cup T$ does not contain all edges of $G$. Our algorithm then proceeds with the last part, called Finalizing. Before describing what Finalizing does, let us take a look at the subgraph $H$ induced by the edges not in $S \cup T$. In the following, $d(x)$ is always the degree of $x$ in $G$, regardless of whether or not $x$ is in $H$.

**Proposition 2.8.** No vertex of $H$ is incident with any edge in $S$. If $w \in V(H)$ is incident with an edge $vw$ in $T$, then $v \not\in V(H)$, $d(v) = 2$, and the other edge of $G$ incident with $v$ is in $S$.

**Proof.** For any edge $xy$ in $S$, Propagation $(S)$ will add all edges incident with either $x$ or $y$ to $T$ and thus neither $x$ nor $y$ can be a vertex of $H$.

Suppose that $vw$ is an edge of $T$ with $w \in V(H)$. Then $vw$ is added by Propagation $(S)$, which means that either $v$ or $w$ is incident with an edge in $S$. Since no vertex of $H$ is incident with any edge in $S$, $v \not\in V(H)$ and $w$ is incident with an edge $uw$ in $S$. If $d(v) \geq 3$, then Propagation $(T)$ would have added edges incident with $w$ other than $uw$ to $S$, and hence $w$ is not a vertex of $H$, a contradiction. So $d(v) = 2$. □

An ear in $H$ is either a cycle $W$: $w_0w_1 \cdots w_k$ with $w_0 = w_k$ and $d(w_i) = 2$ for all $1 \leq i \leq k - 1$ (and $d(w_0) = d(w_k) \geq 2$) or a path $P$: $w_0w_1 \cdots w_k$ with $k$ as large as possible such that $d(w_i) = 2$ for all $1 \leq i \leq k - 1$. (Note that, for $i = 0$, $k$, $d(w_i) \geq 1$, and when $d(w_0) = 2$ the edge incident with $w_i$ not in $H$ is in $T$ according to Proposition 2.8.) It is easy to see that $H$ can be decomposed into ears. In Finalizing, we do the following for each ear $W$ of $H$. If $d(w_0) \leq 2$ and $d(w_k) \leq 2$, then add $w_0w_1$ to $S$ for each even $i$ and add $w_iw_{i+1}$ to $T$ for each odd $i$. Suppose that $d(w_0) \geq 3$. If either $d(w_k) \leq 2$ or $d(w_k) \geq 3$ and $k$ is odd, then add $w_iw_{i+1}$ to $S$ for each odd $i$ and add $w_iw_{i+1}$ to $T$ for each even $i$. If $d(w_k) \geq 3$ and $k \geq 6$ is even, then add $w_iw_{i+1}$ to $S$ for $i = 1$ and each even $i \geq 4$ and add $w_iw_{i+1}$ to $T$ for $i = 0, 2$ and each odd $i \geq 3$. Denote by $(S', T')$ the resulting pair. We remark that the only possible edges in $H$ which are not in $S' \cup T'$ are either in a cycle $w_0w_1w_2w_3w_0$ with $d(w_0) \geq 3$ and $d(w_1) = d(w_2) = d(w_3) = 2$ or in a path $w_0w_1 \cdots w_k$ with $k = 2$ or $4$, $d(w_0) \geq 3$, $d(w_k) \geq 3$ and $d(w_1) = \cdots = d(w_{k-1}) = 2$.

**Proposition 2.9.** If $(S, T)$ is valid, then so is $(S', T')$.

**Proof.** Suppose that $G$ is line-monopolar. Since $(S, T)$ is valid, there is a line-monopolar partition $(S^*, T^*)$ with $S \subseteq S^*$ and $T \subseteq T^*$. Let $S^{**} = (S^* \setminus T^*) \cup (S \cap T^*)$ and $T^{**} = (T^* \setminus S^*) \cup (T \cap S^*)$. Clearly, $S' \subseteq S^{**}$, $T' \subseteq T^{**}$, and $(S^*, T^{**})$ is a partition of $E(G)$. We show that $(S^{**}, T^{**})$ does not contain a conflict. Suppose that $S^{**}$ contains a pair of edges, say $e, f$,
sharing an endvertex. Then at least one of them, say e, is in \( S' \cap T^* \). Since \( T^* \) contains no edge of \( S, e \in S' - S \). In particular, e is in an ear \( W : w_0w_1 \cdots w_k \) of \( H \) as defined above. Since \( S' \) does not contain consecutive edges of \( W, e \) is either the first edge or the last edge of \( W \). By symmetry, we may assume that e is the last edge \( w_{k-1}w_k \). The definition of \( S' \) implies that \( d(w_k) \leq 2 \) and \( f \) is not in \( W \). By Proposition 2.8, \( f \) cannot be in \( S \), and hence is in \( H \). Therefore \( d(w_k) = 2 \) and \( f \) is the other edge incident with \( w_k \), which contradicts the choice of \( W \).

Suppose that \( T^{**} \) contains three edges forming a \( P_4 \). Then at least one of the three edges is in \( T' \cap S^* \), and hence is in \( T' - T \). In particular, this edge is an edge, say \( w_1w_{i+1} \), in an ear \( W : w_0w_1 \cdots w_k \) of \( H \). By the definition of \( T' - T, i \geq 1 \). Since \( W \) does not contain three consecutive edges in \( T' \), we must have \( i = k - 1 \). Observe that \( d(w_{k-1}) = 2 \) and \( w_{k-2}w_{k-1} \) is in \( S' \). So, if \( w_{k-1} \) is an edge of \( T^{**} \) adjacent to \( w_{k-1}w_k \), then \( w_{k-1} \) is not in \( H \) and by Proposition 2.8, \( d(v) = 2 \) and the other edge of \( G \) incident with \( v \) is in \( H \). Hence \( w_k \) cannot form a \( P_4 \) in \( T^{**} \), a contradiction. Therefore \( (S^{**}, T^{**}) \) is a line-monopolar partition of \( G \). \( \square \)

It follows from Proposition 2.9 that \( (S, T) \) contains no conflict and then neither does \( (S', T') \), and in particular, \( (S', T') \) is a line-monopolar partition of \( G \) when \( S' \cup T' \) contains all edges of \( G \). So we may assume that \( S' \cup T' \) does not contain all edges of \( G \). Let \( H^+ \) be the subgraph induced by the edges of \( H \) not in \( S' \cup T' \). From the above remark, \( H^+ \) consists of either cycles \( w_0w_1w_2w_3w_0 \) with \( d(w_0) \geq 3 \) and \( d(w_1) = d(w_2) = d(w_3) = 2 \) or paths \( w_0 \cdots w_k \) with \( k = 2 \) or 4, \( d(w_0) \geq 3, d(w_k) \geq 3 \) and \( d(w_i) = \cdots = d(w_{k-1}) = 2 \). Note that these paths may form cycles of any even length \( \geq 4 \) in \( H^+ \).

**Proposition 2.10.** Suppose that \( (S', T') \) contains no conflict. If some connected component of \( H^+ \) contains two cycles, then \( G \) is not line-monopolar; otherwise \( G \) is line-monopolar.

**Proof.** Let \( u_0u_1 \cdots u_a \) be either a cycle (when \( u_0 = u_a \)) or a path in \( H^+ \) where \( d(u_0) \geq 3 \) and \( a \) is even. It follows from the above remark that \( d(u_i) = 2 \) for each odd \( i \) and either \( d(u_i) \geq 3 \) or \( d(u_{i+2}) \geq 3 \) for each even \( i \). We claim that if \( (S, T) \) is a line-monopolar partition of \( G \) then \( S \) must contain either \( u_0u_1 \) or \( u_0u_{a-1}u_a \). Indeed, if \( S \) does not contain \( u_0u_1 \), then \( S \) has to contain \( u_1u_2 \), as otherwise \( T \) would contain \( u_0u_1, u_1u_2 \) and at least one of two other edges incident with \( u_0 \) forming a \( P_4 \), a contradiction. So \( u_1u_2 \) is in \( S \), and hence \( u_1u_3 \) is in \( T \). Since either \( d(u_2) \geq 3 \) or \( d(u_4) \geq 3, S \) must contain \( u_3u_4 \), as otherwise \( u_2u_3, u_3u_4 \) together with an edge incident with either \( u_3 \) or \( u_4 \) forming \( P_4 \) in \( T \). Continuing this way, we see that in fact \( S \) contains \( u_0u_1 \) for every odd \( i \) and in particular \( u_0u_{a-1}u_a \).

Suppose that \( (S, T) \) is a line-monopolar partition of \( G \) and some component of \( H^+ \) contains two cycles \( u_0u_1 \cdots u_{a-1}u_0 \) and \( v_0v_1 \cdots v_{b-1}v_0 \) and follows from the rules of Preprocessing that both cycles are induced cycles in \( G \). Assume first that the two cycles are vertex disjoint. Let \( w_0w_1 \cdots w_t \) be a shortest path in \( H^+ \) connecting the two cycles. Without loss of generality, assume \( w_0 = u_0 \) and \( w_t = v_0 \). Then \( d(w_0) \geq 3 \) and \( d(v_0) \geq 3 \). Note that \( a, b, c \) are all even. From the above discussion, \( S \) contains one of \( u_0u_1, v_0v_1, u_0u_1v_0, v_0v_1u_0 \), and one of \( u_0v_1, v_0u_1, u_0v_1u_0, v_0u_1v_0, u_0v_1u_0v_1, v_0u_1v_0u_1 \). But this is impossible, as \( S \) is a matching. Assume next that the two cycles share exactly one vertex, say \( w_0 = v_0 \). Again, from the above, \( S \) contains one of \( u_0u_1, u_0u_1 \), one of \( v_0v_1, v_0v_1u_0 \) and one of \( v_0v_1u_0v_1, v_0v_1u_0v_1u_0 \), which is not possible either. Finally, assume that the two cycles share two or more vertices. Then some two vertices are joined by three internal vertex-disjoint paths in \( H^+ \). Let \( x_0 \cdots x_9, y_0y_1 \cdots y_9, z_0z_1 \cdots z_9 \) be three internal vertex-disjoint paths where \( x_0 = y_0 = z_0 \) and \( x_9 = y_9 = z_9 \). Then \( d(x_0) \geq 3 \) and \( d(x_9) \geq 3 \). It follows that \( a, b, c \geq 2 \) are all even. Then \( S \) contains one of \( x_0x_1, x_0y_0, y_9y_1, y_9y_1y_9, y_9y_1y_9y_1, z_0z_1, z_0z_1z_0, z_0z_1z_0z_1 \), which is once again not possible. Therefore if \( H^+ \) contains two cycles then \( G \) is not line-monopolar.

So suppose that no component of \( H^+ \) contains no cycle. We obtain \( (S^*, T^*) \) from \( (S', T') \) by applying the following procedure. When a component of \( H^+ \) contains no cycle, arbitrarily choose a vertex \( w_0 \) with \( d(w_0) \geq 3 \), and for each path \( w_0w_1w_2 \cdots w_k \) in the component, add \( w_1w_1+1 \) to \( S' \) for each odd \( i \) and to \( T' \) for each even \( i \). When a component contains a (unique) cycle, we first add edges alternatively to \( S' \) and \( T' \) along the cycle, and then for each vertex \( w_0 \) in the cycle with \( d(w_0) \geq 3 \), and a path \( w_0w_1w_2 \cdots w_k \) in the component, add \( w_1w_1+1 \) to \( S' \) for each odd \( i \) and to \( T' \) for each even \( i \). Then it is easy to verify that \( (S^*, T^*) \) is a line-monopolar partition of \( G \). \( \square \)

So, if Finalizing, we also check whether a connected component of \( H^+ \) contains two cycles. If some connected component of \( H^+ \) contains two cycles then return \( G \) is not line-monopolar. Otherwise we obtain a line-monopolar partition of \( G \) as described in the proof of Proposition 2.10.

**Algorithm 2.11.** Input: A bipartite graph \( G \).

Output: Either a line-monopolar partition \( (S, T) \) of \( G \) or report that \( G \) is not line-monopolar.

1. **Do** Preprocessing.
2. **Do** Propagation until all edges in \( S \cup T \) are processed.
3. **Check** possible conflicts in \( (S, T) \): if there is a conflict, then return ‘\( G \) is not line-monopolar’.
4. **Do** Finalizing: if some connected component of \( H^+ \) contains two cycles, then return ‘\( G \) is not line-monopolar’; otherwise return a line-monopolar partition \( (S, T) \).

**Proposition 2.12.** Given a bipartite graph \( G \) with \( n \) vertices and \( O(n) \) edges, **Algorithm 2.11** decides in time \( O(n) \) whether \( G \) is line-monopolar and returns a line-monopolar partition in the case when \( G \) is a line-monopolar graph.
Algorithm 2.11 follows from Propositions 2.4–2.10. To implement Algorithm 2.11, we compute, for each vertex $v$, the set $N(v)$ of neighbours of $v$ and the set $N_{\geq 3}(v)$ of neighbours of $v$ of degree $\geq 3$. Also for each vertex $v$ we compute the set $N_S(v)$ (resp. $N_T(v)$) of neighbours $u$ of $v$ for which $uv$ is in $S$ (resp. $T$) and maintain the numbers $d_S(v) = |N_S(v)|$, $d_T(v) = |N_T(v)|$. We also maintain a stack $L_S$ of unprocessed edges from $S$ and a stack $L_T$ of unprocessed edges from $T$.

To perform Preprocessing, we pick a vertex $v$ with $d(v) \geq 3$ and add edge $uv$ to $S$ and to $L_S$ for each $u \in N_{\geq 3}(v)$. We update the sets $N_S(u)$, $N_T(u)$ and the numbers $d_S(u)$, $d_T(u)$. The running time is proportional to the number $O(n)$ of edges of $G$.

To perform Propagation $(S)$, we pick an edge $uv \in L_S$ and add edges $xu$, $yv$ to $T$ and to $L_T$ for each $x \in N(u)$ and $y \in N(v)$. We update the sets $N_T(u)$, $N_T(v)$, $N_S(y)$, $N_T(y)$, $N_L$ and the numbers $d_T(u)$, $d_T(v)$, $d_T(x)$, $d_T(y)$. To perform Propagation $(T)$, we pick an edge $vw \in L_T$. When $d(v) \geq 3$, or $d_T(v) \geq 2$, or $d_S(v) \geq 1$, we add to $S$ and to $L_S$ edges $ux$ where $x \in N(u)$. We also update the sets $N_S(w)$, $N_S(x)$, $L_T$ and the numbers $d_S(w)$, $d_S(x)$. Since every edge is examined once, the running time of Step 2 is the number $O(n)$ of edges of $G$.

To detect possible conflicts in $(S, T)$, we check if $d_S(v) \geq 2$ for any vertex $v$, and for each vertex $v$ with $d_T(v) \geq 2$ we check if $d_T(u) \geq 2$ for any vertex $u \in N_T(v)$. The running time of this is proportional to the number $O(n)$ of edges of $G$.

Finally, Step 4 can also be implemented to run in time $O(n)$. First of all the ear decomposition of $H$ can be computed in time $O(n)$ since $G$ has $O(n)$ edges. Partitioning the edges of ears (as defined before Proposition 2.9) takes time $O(n)$. To check if any component of $H^+$ contains two cycles, we compute the components of $H^+$ and check if any of them has the number of edges greater than the number of vertices. This can be done in time $O(n)$.

Combining Propositions 2.2 and 2.12, we have the following.

**Theorem 2.13.** Given a bipartite graph $G$, there is an $O(n)$ algorithm which decides whether $G$ is a line-polar graph and, when it is, returns a line-polar partition of $G$. □

**Note** (Added in June 2010). An $O(n)$ recognition algorithm for general line-polar graphs has recently been devised by Churchley and the second author [2].

**Acknowledgements**

We are grateful to the referees for their invaluable comments which improved the implementation of the algorithm.

**References**