Hardness and algorithms for variants of line graphs of directed graphs

Mourad Baïou[1], Laurent Beaudou *[1], Zhentao Li[2], and Vincent Limouzy **[1]

[1] Limos - Cnrs and Univ. Clermont II
[2] Lip - Cnrs and ENS Lyon

Abstract. Given a directed graph $D = (V, A)$ we define its intersection graph $I(D) = (A, E)$ to be the graph having A as a node-set and two nodes of $I(D)$ are adjacent if their corresponding arcs share a common node that is the tail of at least one of these arcs. We call them facility location graphs since they arise from the classical uncapacitated facility location problem. In this paper we show that facility location graphs are hard to recognize but they are easy to recognize when the underlying graph is triangle-free. We also determine the complexity of the vertex coloring, the stable set and the facility location problem for triangle-free facility location graphs.

1 Introduction

In this paper we study the following class of intersection graphs. Given a directed graph $D = (V, A)$, we denote by $I(D) = (A, E)$ the *intersection graph of* D defined as follows:

- the node-set of $I(D)$ is the arc-set of D,
- two nodes $a = (u, v)$ and $b = (w, t)$ of $I(D)$ are adjacent if one of the following holds: (1) $u = w$, (2) $v = w$, (3) $t = u$, (4) $(u, v) = (t, w)$ (see Figure 1(a)).

We focus on two aspects: the recognition of these intersection graphs and some combinatorial optimization problem in this class. De Simone and Mannino [1] considered the recognition problem and provided a characterization of these graphs based on the structure of the (directed) neighbourhood of a vertex. Unfortunately this characterization does not yield a polynomial time recognition algorithm.

Intersection graphs we consider arise from the *uncapacitated facility location problem* (UFLP) defined as follows. We are given a directed graph $D = (V, A)$, costs $f(v)$ of opening a facility at node v and cost $c(u, v)$ of assigning v to u (for each $(u, v) \in A$). We wish to select a subset of facilities to open and an assignment of each remaining node to a selected facility so as to minimize the cost of opening the selected facilities plus the cost of arcs used for assignments.

* Supported by Région Auvergne project M2F
** Supported by the ANR junior project DORSO

This problem can be formulated as a linear integer program equivalent to maximal clique formulation of the *maximum stable set problem* associated with $I(D)$, where the weight of each node (u, v) of $I(D)$ is $f(u) - c(u, v)$. This correspondence is well known in the literature (see [2,3,1]). We may consider several combinatorial optimization problems on directed graphs that may be reduced to the maximum stable set problem on an undirected graph. For example, Chvátal and Ebenegger [4] reduce the max cut problem in a directed graph $D = (V, A)$ to the maximum stable set problem in the following intersection graph called the *line graph of a directed graph*: we assign a node to each arc $a \in A$ and two nodes are adjacent if the head of one (corresponding) arc is the tail of the other. They prove that recognizing such graphs is NP-complete. Balas [5] considered the asymmetric assignment problem. He defined an intersection graph of a directed graph D where nodes are arcs of D and two nodes are adjacent if the two corresponding arcs have the same tail, the same head or the same extremities without being parallel. Balas uses this correspondence to develop new facets for the asymmetric assignment polytope.

Fig. 1. (a)The adjacency of two nodes a and b in $I(D)$ (b) A graph which is not a FL graph

We may generalize the notion of line graphs to directed graphs in many ways. The simplest involves deciding

1. if arcs that share a head are adjacent,
2. if arcs that share a tail are adjacent, and
3. if two arcs are adjacent when the head of one arc is the tail of the other.

It is not too difficult to show the recognition problem is easy if we choose non-adjacency for (3).

So suppose arcs of type (3) are adjacent. Choosing adjacency for (1) and (2) gives the line graphs of the underlying undirected graph, and these are easy to recognize [6]. Choosing non-adjacency for both (1) and (2) leads to the line graphs defined by Chvátal and Ebenegger and it is NP-complete to recognize them [4]. And picking exactly one of (1) and (2) to be adjacent and non-adjacency for the other leads to the same class of graphs (as we can simply reverse all arcs of a digraph before taking its line graph) and we wish to determine the complexity of recognizing this very last class.

Finally, note that since the stable set problem in our class is equivalent to the facility location problem, one may use tools developed for facility location

problem to solve the stable set problem on these graphs. It is well known that in practice the facility location problem may be solved efficiently via several approaches: polyhedra, approximation algorithms and heuristics.

This paper is organized as follows. Section 2 contains some basic definitions and notations. Other definitions and notations will be given when needed. In Section 3 we show that the subclass of triangle-free facility location graphs are recognizable in linear time and, in contrast, in Section 4, we show that facility location graphs are hard to recognize. Section 5 is devoted to some combinatorial optimization problem in facility location graphs. In particular we show that the maximum stable set problem remains NP-complete in triangle-free facility location graphs but the vertex coloring problem is solvable in polynomial time in this class. We also discuss the facility location problem and show it is NP-complete in some restricted class of graphs. We omit some of the more routine but tedious proofs in this extended abstract but make available a complete preprint [7].

2 Definitions and notations

Let G be an undirected graph, we say that G is a *facility location* (FL) graph if there exists a directed graph D such that $G = I(D)$. D is the *preimage* of G.

Let $D = (V, A)$ be a directed graph. For an arc $a = (u, v) \in A$, we say $t(a) = u$ is the *tail* of a and $h(a) = v$ is the *head* of a. A *sink* is a node which is a tail of no arc in A. A *branch* is an arc a where $h(a)$ is not the head or tail of any other arc.

An undirected graph G is triangle-free if it does not contain a clique of size 3. A *wheel* W_n is a graph obtained from a cycle C_n by adding a vertex adjacent to all vertices of the cycle.

3 Recognizing triangle-free facility location graphs

In Section 4, we will show recognizing FL graphs is NP-complete. Our reduction constructs a graph with many cliques of size 3 but none of size 4. Hence it is natural to ask if the recognition problem remains difficult for triangle-free FL graphs. In this section, we show triangle-free FL graphs can be recognized in polynomial time.

In subsection 3.1 we examine the structure of general FL graphs. In subsection 3.2, we prove the main result of this section.

3.1 Structural properties of facility location graphs

Here, we state some basic properties of FL graphs.

Remark 1 *The preimage of any cycle either contains two arcs with the same tail or is a directed cycle.*

We will use the next three propositions as reduction rules in a recognition algorithm. Each rule allows us to find some structure in the graph and recurse on a simpler graph until no rules apply. If such a simpler graph is also triangle-free then it has a very specific form that is easy to recognize.

Remark 2 *If u is a degree one vertex adjacent to a degree two vertex in an undirected graph G then G is a FL graph if and only if $G - u$ is a FL graph.*

Lemma 1. *If G is a FL graph, then there exists a digraph D such that $G = I(D)$ and every sink node in D has exactly one entering arc.*

Lemma 2. *If u and v are adjacent degree two vertices in G with no common neighbours in an undirected graph G then G is a FL graph if and only if $G - uv$ is a FL graph.*

3.2 Application to triangle-free facility location graphs

We now characterizes FL graphs on which no rules from the previous section apply. The *outdegree* of a vertex is the number of arcs leaving that vertex.

Lemma 3. *Let G be a connected triangle-free graph with no degree two vertex adjacent to degree ≤ 2 vertices. If $G = I(D)$ then vertices of D have outdegree at most two. Furthermore, vertices with outdegree exactly 2 have a sink as one of their outneighbours.*

Theorem 3 *Let G be a triangle-free graph. Let G' be the graph obtained from G by removing all edges between degree two vertices. Then G is a FL graph if and only if G' has at most one cycle per connected component.*

Proof. By Lemma 2 (and since G is triangle-free so no adjacent vertices share a common neighbour), we only need to show that G' is a FL graph if and only if G' has at most one cycle per component. Note that G' is also triangle-free.

Necessity. Let G' be a triangle-free facility location graph. By Lemma 1, there is a D where every sink has indegree one with $G' = I(D)$. By Remark 1 and Lemma 3, the preimage of every cycle in G' is a directed cycle.

Suppose that a connected component of G' contains two cycles C_1 and C_2. Since both their preimages are directed cycles, the preimages of any vertex in both C_1 and C_2 is a common arc in both directed cycles and this leads to a triangle in G' (by taking a common arc and two differing arcs following it, one in each cycle). Thus, C_1 and C_2 are vertex disjoint. Since C_1 and C_2 belong to the same connected component, there is a path P in G' from some $u \in C_1$ to $v \in C_2$. Then the image of the second vertex of P points towards the image of C_1 and the image of the second to last vertex of P points towards the image of C_2. So the image of P is not a directed path. But then some two consecutive vertices in P have an image that share a common tail, a contradiction to Lemma 3.

Sufficiency. Consider a connected component of G. Suppose that it consists of a tree. Let us construct a directed graph D with $G = I(D)$. Pick any node r as a root. Let $r = (u_0, v_0)$. Let $r_1, \ldots, r_k$ be the children of r in G, we set $r_i = (v_i, u_0)$ for $i = 1, \ldots, k$. Now each node r_i play the role of r and we repeat this step. This procedure ends with a directed graph D such that $G = I(D)$.

Suppose that there is a cycle C. This cycle must be chordless. Let C' be a directed cycle where each arc in C' correspond to a node in C. The rest of this component consist of disjoint trees each intersect C in one node. If this node is chosen to be the root of the tree, then the procedure above may be applied to get a directed graph D such that $G = I(D)$. □

We are now ready to describe our recognition algorithm, which is the main result of this section.

Theorem 4 *Given an undirected triangle-free graph $G = (V, E)$, we may decide whether or not G is a facility location graph in $O(|E|)$.*

Proof. In $O(|E|)$ we may remove all the edges $e = bc$ with both b and c of degree two. Then we apply a breadth-first search in $O(|E|)$. If a node is encountered more than twice or there are two nodes that were encountered twice, then there are two cycles. Otherwise G is a facility location graph. □

4 Recognizing facility location graphs is NP-complete

The main result of this section is the following theorem.

Theorem 5 *Recognizing facility location graphs is* NP*-complete.*

We will reduce the problem 3-SAT to the recognition of FL graphs. We assume we are given an instance of 3-SAT. That is, variables $x_1, \ldots, x_n$ and a Boolean formula $F = C_1 \wedge \cdots \wedge C_m$, where each clause $C_j = \lambda_{j_1} \vee \lambda_{j_2} \vee \lambda_{j_3}$, for $j = 1 \ldots, m$. We construct an undirected graph G_F from F and we show that F is satisfiable if and only if G_F is a facility location graph.

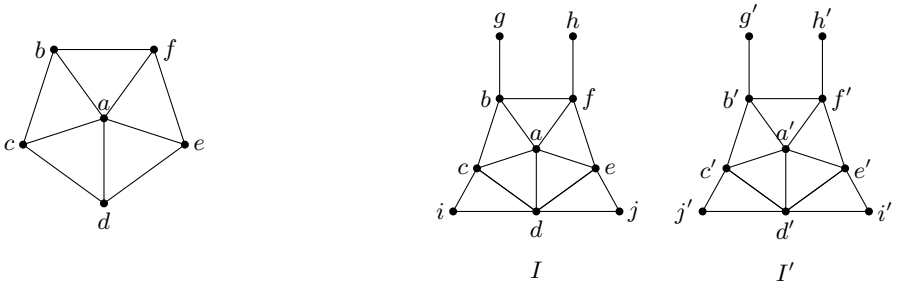
We build G_F using gadgets for variables and clauses. Values for variables are stored, replicated and negated through the "branches" of the variable gadgets. These branches are then connected to the clauses gadgets of clauses that contain these variables (and their negation).

More precisely, the construction of G_F follows three steps: (1) for each variable x_i, we construct a graph called GAD_i^1 (GAD stands for gadget), (2) for each clause C_j, another gadget called GAD_j^2 is constructed and (3) we connect the graphs GAD_i^1 and GAD_j^2 to produce G_F. Each graph GAD_i^1 contains $2m$ branches where each branch express the fact that the variable x_i (or $\bar{x}_i$) is present in the clause C_j, $j = 1, \ldots, m$. Each graph GAD_j^2 contains exactly three branches where each branch expresses the literals of this clause λ_{j_1}, λ_{j_2} and λ_{j_3}.

The three following subsections are devoted to the construction of the graphs GAD_i^1, GAD_j^2 and G_F.

4.1 Variable gadgets

Our variable gadgets is built by identifying vertices in many copies of a graph I with only two preimages. We think of one preimage of I as the Boolean value "true" and the other as "false". I (see Figure 2(b)) is constructed from the wheel W_5 (see Figure 2(a)) by adding four vertices.



(a) The wheel W_5.

(b) I and I' are extensions of W_5

Fig. 2.

We make m copies of I for each variable and combine these copies so that the associated Boolean values are all equal. To ease our analysis, we first examine two copies of I identified on the vertex j in both copies (see Figure 3). Then the preimage of the two copies of I are forced to be the same. We call this graph Inv, the *inverter*, and will use in later construction.

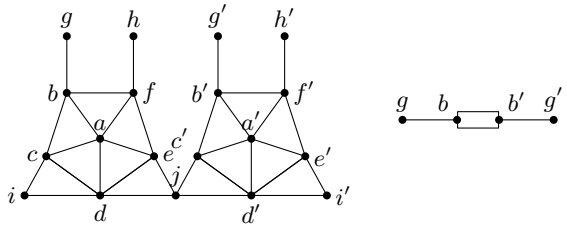


Fig. 3. The graph Inv and its abbreviation.

Now if we take m copies of I and identify the node labelled j in a copy with the node labelled i in the next copy (for the first $m-1$ copies, which have successor) then the resulting gadget (see Figure 4) forces all copies of I to have the same preimage. In fact, the preimages of vertices labelled i, j and d in all copies for a directed path.

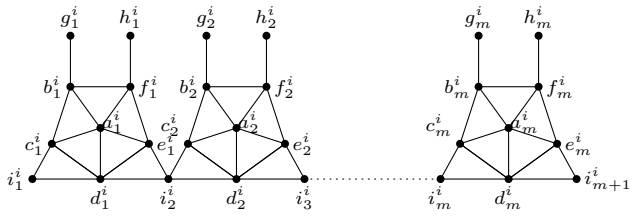


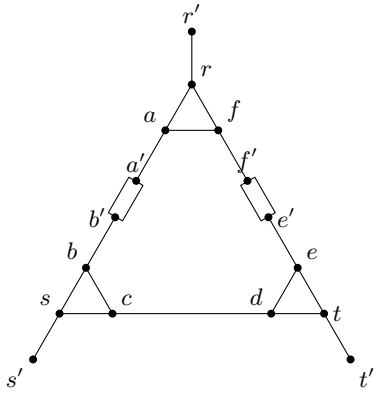
Fig. 4. Graph for every variable x_i, $\textsc{Gad}_i^1$.

Lemma 4. *For each directed graph D with $I(D) = \textsc{Gad}_i^1$ exactly one of the following two assumptions holds:*

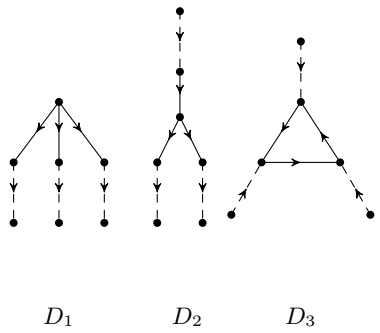
(i) $h(b_j^i) = t(g_j^i)$ *and* $t(f_j^i) = h(h_j^i)$ *for each* $j = 1, \ldots, m$,
(ii) $t(b_j^i) = h(g_j^i)$ *and* $h(f_j^i) = t(h_j^i)$ *for each* $j = 1, \ldots, m$.

4.2 Clause gadgets

We now describe how to build clause gadgets (see Figure 5(a) for one such gadget and the labelling we use). They are built from three nets (triangles with three degree 1 vertices, one adjacent to each vertex of the triangle) and two copies of Inv (of the previous section) by identifying some edges.



(a) Graph Gad_j^2 for clause C_j.



(b) The solid line are the nodes of the triangle Δ and the dashed lines are its pending nodes.

Fig. 5.

We begin with the following remark.

Remark 6 *There are only three possible preimages for a triangle of a net: those shown in Figure 5(b).*

Lemma 5. *No preimage of* GAD_j^2 *has* $h(r'_j) = t(r_j)$, $h(s'_j) = t(s_j)$ *and* $h(t'_j) = t(t_j)$.

However, if we pick any proper subset of the above three contraints then there is a preimage of GAD_j^2 *where those constraints are satisfied and no other constraints are satisfied.*

4.3 Reduction from 3-SAT

As discussed at the beginning of this section, we build a graph G_F from a Boolean formula F. Build a copy of GAD_i^1 for each Boolean variable x_i and a copy of GAD_j^2 for each clause C_j. We combine some of these disjoint gadgets as follows.

We connect GAD_i^1 and GAD_j^2 through their branches if x_i or $\bar{x}_i$ appears in C_j. Specifically, for each clause $C_j = \lambda_{j_1} \vee \lambda_{j_2} \vee \lambda_{j_3}$ we identify the following vertices:

if $\lambda_{j_1} = x_{j_1}$, we identify r_j with $g_j^{j_1}$ and r'_j with $b_j^{j_1}$,

if $\lambda_{j_1} = \bar{x}_{j_1}$, we identify r_j with $h_j^{j_1}$ and r'_j with $f_j^{j_1}$.

We proceed in the same way for remaining literals λ_{j_2} and λ_{j_3}.

4.4 Proof of Theorem 5

Since the problem 3-SAT is NP-complete, it is sufficient to prove that a Boolean formula F is true if and only if the graph G_F we build is a facility location graph.

Suppose G_F is a facility location graph with preimage D. Then we claim F evaluates to true with the following assignment.

$$x_i = \begin{cases} \text{true} & \text{if the arc } g_1^i \text{ enters the arc } b_1^i \text{ in } D, \\ \text{false} & \text{otherwise} \end{cases}$$

We say an arc a *enters* an arc b if $h(a) = t(b)$.

Notice that from Lemma 4 whenever the arc g_1^i enters the arc b_1^i, then g_j^i enters the arc b_j^i for each $j = 1, \ldots, m$. Let C_j be any clause of F. From Lemma 5 (i), we must have that r_j enters r'_j, or s_j enters s'_j or that t_j enters t'_j in any directed graph whose intersection graph is GAD_j^2. We may assume that r_j enters r'_j. By the definition of G_F the branch $r_j r'_j$ is identified with $g_j^i b_j^i$ when x_i is present in C_j and in this case $x_i = 1$ and so $C_j = 1$. Otherwise the branch $r_j r'_j$ is identified with $h_j^i f_j^i$ when $\bar{x}_i$ is present in C_j. So the arc h_j^i enters the arc f_j^i and from Lemma 4 we have that the arc b_j^i enters the arc g_j^i and by definition we have $x_i = 0$, which implies that $C_j = 1$.

Now assume that there is an assignment of the variables x_i, $i = 1, \ldots, n$ for which F evaluates to true. We build a preimage of G_F as follows. By Lemma 4, we can (independently) build preimages for each GAD_i^1 so that g_j^i enters b_j^i if x_i is true and b_j^i enters g_j^i if x_i is false. Now given a clause $C_j = \lambda_{j_1} \vee \lambda_{j_2} \vee \lambda_{j_3}$, from

Lemma 5 there is no preimage only when the assumption (i) of Lemma 5 is not satisfied. But one can check that this may happen only when all of $\lambda_{j_1}, \lambda_{j_2}, \lambda_{j_3}$ are false, which is not possible.

5 Consequences and related problems

5.1 The vertex coloring problem

A *vertex coloring* of a graph is an assignment of colors to the nodes of the graph such that no two adjacent nodes receive the same color. The minimum number needed for a such coloring is called the *chromatic number* and denoted by $\chi(G)$. It is well known that finding $\chi(G)$ is NP-complete for triangle-free graphs. A direct consequence of the previous section shows that when G is a triangle-free facility location graph, it is 2-degenerate, and therefore $\chi(G) \leq 3$.

Theorem 7 *If G is a triangle-free facility location graph, then $\chi(G) \leq 3$. Moreover, $\chi(G)$ may be computed in $O(|E|)$.*

A natural question arises: whether or not coloring facility locations graphs is polynomial. Unfortunately this problem is NP-complete by a reduction from the edge coloring problem (i.e., the vertex coloring problem for line graphs).

Theorem 8 *Coloring facility locations graphs is* NP*-complete.*

Proof. Given an input graph G that we wish to k edge color (a task that is NP-complete [8]), we build an auxiliary digraph D obtained from vertices $V(G)$ with no arcs by adding a vertex x_{uv} for each edge $uv \in E(G)$ with entering arcs from u and v and $k-1$ leaving arcs to $k-1$ new vertices. Now any k vertex coloring of $I(D)$ forces vertices corresponding to both entering arcs of x_{uv} to be colored the same as vertices of leaving arcs from x_{uv} take up $k-1$ colors. It is easy to see that a k vertex coloring of $I(D)$ leads to a k edge coloring of G by giving $e \in E$ the color of arcs entering x_e. Similarly, a k edge coloring of G gives a k vertex coloring of $I(D)$. □

5.2 The stable set problem

Given an undirected graph $G = (V, E)$, a subset of nodes $S \subseteq V$ of an undirected graph is called a *stable set* if there is no edge between any two nodes of S. The *maximum stable set problem* is to find a stable set of maximum size. This size is usually called the *stability number* and denoted by $\alpha(G)$. If we associate a weight $w(v)$ to each vertex $v \in V$, then the *maximum weighted stable set problem* is to find a stable set S with $\sum_{v \in S} w(v)$ maximum.

The maximum stable set problem is NP-complete for triangle-free graph. Poljak [9] showed this by building an auxiliary graph SUB_G from an input graph $G = (V, E)$ by replacing any edge $e = uv$ in E by a path $uu', u'u'', u''v$. Now SUB_G is triangle-free and $\alpha(\text{SUB}_G) = \alpha(G) + |E|$. By Theorem 3, SUB_G is also a facility location graph since the removal of the edges $u'u''$ yields a graph where each connected component is a star. Hence, we obtain the following result,

Theorem 9 *The maximum stable set problem is* NP*-complete in triangle-free facility location graphs.*

Since any triangle-free facility location graph can be colored with 3 colors in $O(|E|)$ Theorem 7, we can get a 3-approximation algorithm for the maximum (weighted) stable set problem. Indeed, we may assume the input graph G has only positive weights and pick the color class S of maximum (total) weight. Now S has weight at least a third of the weight of all of G which is at least a third of the weight of largest stable set in G.

5.3 The facility location problem

Recall that the uncapacitated facility location problem (UFLP) associated with a directed graph D is equivalent to the maximum weighted stable set problem for $I(D)$. Therefore, from Theorem 9 we have the following corollary.

Corollary 10 *The uncapacitated facility location problem remains* NP*-complete even when the input digraph does not contain the four graphs of Figure 6(a) as subgraphs.*

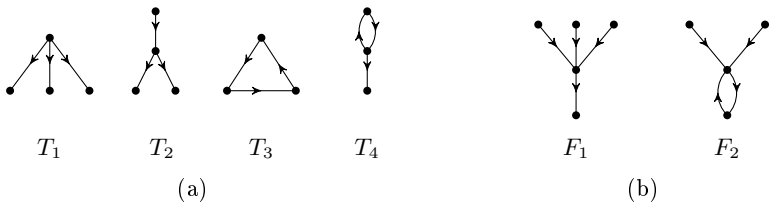


Fig. 6. (a)Graphs T_1, T_2, T_3 and T_4 (b)Graphs F_1 and F_2

Mohar proved the following result in [10].

Theorem 11 *[10] The maximum stable set problem in 2-connected cubic planar graphs is* NP*-complete.*

This results allows us to strengthen Corollary 10 to the following theorem.

Theorem 12 *The uncapacitated facility location problem is* NP*-complete for graphs that do not contain any of T_1, T_2, T_3, T_4, F_1 and F_2 as a subgraph.*

Proof. Let $G = (V, E)$ be an undirected 2-connected cubic planar graph. From G define the subdivision of it, SUB_G, as in the previous subsection, that is each edge $e = uv \in E$ is replaced by path of size three. Now we construct a directed graph D containing none of the graphs T_1, T_2, T_3, T_4, F_1 and F_2 as a subgraph and such that $I(D) = \text{SUB}_G$. Thus from Theorem 11 the maximum weighted

stable set problem is NP-complete in 2-connected cubic planar graphs, and by equivalence we have that UFLP is also NP-complete in graphs satisfying the theorem's hypothesis. Now let us give the construction of D.

From Petersen's theorem the graph G contains a perfect matching M. Let G' be the graph obtained by removing M. Each component of G' is a chordless cycle. Let $C = v_0, v_1, \ldots, v_p$ be one of these cycles. In SUB_G this cycle corresponds to a cycle $C' = v_0, v_1, v_2, \ldots, v_{3p}, v_{3p+1}, v_{3p+2}$. Let us construct a directed graph D with $I(D) = \text{SUB}_G$. Each cycle C' of SUB_G may be defined in D by the directed cycle where the arc v_i enters the arc v_{i+1} for each $i = 0, \ldots, 3p+1$, and the arc v_{3p+2} enters the arc v_0 (an arc a enters an arc b means that the head of a coincide with the tail of b). To complete the definition of D we need to consider all the edges of M and their subdivisions. Let $e = uv \in M$ and u_1, u_2, u_3, u_4 the corresponding path in SUB_G. Complete the construction of D by creating for every such edge e two arcs u_2 and u_3 having the same tail where u_2 enters the arc u_1 and u_3 enters the arc u_4.

References

1. De Simone, C., Mannino, C.: Easy instances of the plant location problem. Technical Report R. 427, IASI, CNR (1996)
2. Avella, P., Sassano, A.: On the p-median polytope. Mathematical Programming **89** (2001) 395–411
3. Cornuejols, G., Thizy, J.M.: Some facets of the simple plant location polytope. Math. Program. **23** (1982) 50–74
4. Chvátal, V., Ebenegger, C.: A note on line digraphs and the directed max-cut problem. Discrete Applied Mathematics **29** (1990) 165 – 170
5. Balas, E.: The asymmetric assignment problem and some new facets of the traveling salesman polytope on a directed graph. SIAM Journal on Discrete Mathematics **2** (1989) 425–451
6. Beineke, L.W.: Characterizations of derived graphs. Journal of Combinatorial Theory **9** (1970) 129 – 135
7. Baïou, M., Beaudou, L., Li, Z., Limouzy, V.: On a class of intersection graphs. `http://arxiv.org/abs/1306.2498` (2013)
8. Holyer, I.: The NP-completeness of edge-coloring. SIAM Journal on Computing **10** (1981) 718–720
9. Poljak, S.: A note on stable sets and colorings of graphs. Commentationes Mathematicae Universitatis Carolinae **15** (1974) 307 – 309
10. Mohar, B.: Face covers and the genus problem for apex graphs. J. Comb. Theory, Ser. B **82** (2001) 102–117