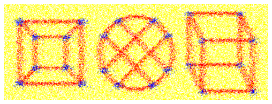
Minimizing the Hamming distance between a graph and a line-graph to discover the topology of an electrical network

Wilfried Ehounou, Dominique Barth, Arnaud de Moissac, Dimitri Watel, Marc-Antoine Weisser


Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 0, no. 0, pp. 0–0 (0)
DOI:

Minimizing the Hamming distance between a graph and a line-graph to discover the topology of an electrical network

Wilfried Ehounou[1,2] *Dominique Barth*[1] *Arnaud de Moissac*[2]
Dimitri Watel[3] *Marc-Antoine Weisser*[4]

[1]DAVID, University of Versailles-St Quentin, Versailles, France
[2]DCBrain, 55 Boulevard Vincent Auriol, Paris, France
[3]LRI, CentraleSupelec, Paris-Saclay University, France
[4]ENSIIE and SAMOVAR, Evry, France

Abstract

Our objective is to discover the topology of an energy distribution network modeled by a flow digraph from which we know the set of arcs without identification of their extremities. We also know possible correlations between these arcs from measurements in which errors could occur. To discover the network's topology from the obtained correlation matrix between arcs, we consider the graph whose incidence matrix is the correlation matrix with entries rounded to zero or one. Without errors in the correlations, this graph is the line-graph of the distribution network. We propose an algorithm that given the rounded correlation matrix M, finds the incidence matrix of a line-graph with minimum Hamming distance to M. We then evaluate the performance of this approach by simulation.

Keywords: Line-graph, Electrical Networks, NP-Completeness, Heuristic

Submitted:	Reviewed:	Revised:	Reviewed:	Revised:
December 2018	May 2019	July 2019	January 2020	January 2020
	Accepted:	Final:	Published:	
	February 2020	February 2020		
		Communicated by:		
		Evans William		

E-mail addresses: wilfried.ehounou@ens.uvsq.fr (Wilfried Ehounou) dominique.barth@uvsq.fr (Dominique Barth) arnaud@dcbrain.com (Arnaud de Moissac) dimitri.watel@ensiie.fr (Dimitri Watel) marc-antoine.weisser@centralesupelec.fr (Marc-Antoine Weisser)

1 Introduction

The maintenance of energy networks, especially electricity distribution networks, requires a good knowledge of the equipment used and their state, but also the topology and capabilities of networks [10]. This knowledge is not often perfect because of changes in the topology that have not been reported or inconsistencies in energy flow measurements from which the topology is deduced. This can have serious consequences on the resilience of equipment in such a network [11].
In order to correct these errors, the approach considered here is to predict a probable topology of the network, represented by a Directed Acyclic Graph (DAG), by correlating the measures regularly performed and from the *a priori* knowledge of the topology. This problem of prediction can be related to a problem of discovery of network topologies. This type of problem has been particularly studied in the context of telecommunication networks based on tomography or traffic analysis [5]. Those techniques cannot be used in the case of energy networks as the only available data are correlations between links of the networks obtained from electrical flow measurements. These correlations, if they are correct, induce the line-graph [3, 9] of the underlying undirected graph of the electrical network. In most cases, the errors or lack of measurements mean that the graph deduced from the correlations also contains errors (added and deleted edges) and that it is therefore not the expected line-graph of the topology of the electrical network (in most cases, it is no longer a line-graph). The objective is to correct this graph in order to obtain a line-graph as close as possible to the expected line-graph [6].

Various works have been devoted to determine if a graph is a line-graph and if yes, to discover a graph of which it is the line-graph (called root graph of the line-graph) [1, 4, 12, 14, 15]. In particular, initial works [12] show that a graph is a *line-graph* if and only if it admits a decomposition of its set of edges in maximum complete induced subgraphs such that each vertex belongs to at most two of these subgraphs. Line-graphs can also be characterized by a set of nine forbidden induced subgraphs [1]. The concept of root graph was introduced by Whitney [16] and he proved that if two connected line-graphs are isomorphic, then their root graphs are isomorphic except for the triangle graph. Finally, it is possible to compute the root graph $G = (V, E)$ in time $O(|E|)$ [12] and in time $O(log|V|)$ with a parallel algorithm using $O(|E|)$ processors [14].

The problem we are dealing with is, given an input graph, to determine the minimum number of edges to add or remove to obtain a line-graph. In the context of power grids, the assumption we want to verify is that if the number of correlation errors is small, then the line-graph obtained after adding or deleting edges is close to the one of the actual topology of the network. In this paper, we firstly focus on the complexity of the problem. When only edge deletion is allowed, the problem is NP-Complete [8, 17]. Determining whether the problem remains NP-Complete if we allow both edge addition and deletion is an open

problem. In this paper, we make a step closer to this result by proving that the problem is NP-Complete if we only allow edge addition. The second part of the paper is dedicated to the description of an heuristic algorithm to solve the problem on graphs with small degrees.

The paper is organized as follows. In Section 2, we define the problem and we deal with its complexity. Section 3 is dedicated to the definition and the analysis of the proposed algorithms, based on some line-graph properties. Finally, Section 4 presents an evaluation of these algorithms based on experiments on generated graphs.

2 Network modeling and problem formulation

In this paper, we consider the graph theory definitions and notations from [2]. We consider an electrical network modeled by a DAG $H = (V, A)$, in which each vertex with incoming (resp. outgoing) degree equal to zero is a source of energy (resp. final consumer of energy). We also consider a matrix μ_C, obtained from measurements on links, in which each row or column is associated with an arc of the DAG. Each element $\mu_C[i, j]$ is a value between 0 and 1 being the correlation between arc i and arc j in H. Ideally, if these two arcs have one extremity (indifferently initial or final) in common, the value is close to 1, otherwise it is close to 0. But as explained in the introduction, some measurement errors may appear.

Starting from μ_C and a chosen threshold value $\mathcal{S}$ between 0 and 1, we consider the matrix M of the same dimension as μ_C in which $M[i, j] = 1$ if and only if $\mu_C[i, j] \geq \mathcal{S}$, else $M[i, j] = 0$.

The line-graph of H is an undirected graph $L(H) = (V', E)$, where $V' = A$ and a pair $[a, a']$ of vertices of $L(H)$ is an edge if and only if a and a' have a common extremity node (indifferently initial or final) in H.

Let $G = (V, E)$ be the undirected graph whose M is the adjacency matrix, considering threshold $\mathcal{S}$. If there is no error in $\mu_C[i, j]$ and if the threshold $\mathcal{S}$ is well chosen, then G is the line-graph of H. We then focus on cases where some errors may occur in μ_C and consequently in M.

We say that an undirected graph is a line-graph if and only if it is the line-graph of a graph, called **root graph** of the line-graph. The following property gives a characterization of line-graphs. Recall that a clique in a graph $G = (V, E)$ is a subset of V inducing a complete subgraph of G [2].

Property 1 *[12] A graph $G = (V, E)$ is a* line-graph *if and only if there is a set $\mathcal{C}$ of cliques in V such that each vertex of V belongs to one or two cliques of $\mathcal{C}$ and each edge of E belongs to the complete subgraph induced by one and only one clique in $\mathcal{C}$.*

A **line-coverage** of a line-graph G is a set of cliques satisfying Property 1. In our context, if there exists a line-coverage of G then we assume that G is the exact line-graph of H (note that some errors in the adjacency matrix of a

line-graph could transform it into another line-graph, but we do not consider such cases here). Otherwise, we want to determine the line-graph L_G with the same set of vertices as G and minimizing a specific edit distance with G, defined as follows.

Definition 1 *Let G and G' be two graphs with the same set of vertices. The* **Hamming distance** *$dH(G, G')$ between G and G' is the Hamming distance between their two adjacency matrices, i.e., the number of elements having a different value in each of the two matrices.*

We consider the following problem.

Problem Proxi-Line
Data: A graph $G = (V, E)$, an integer k.
Question: Does there exist a line-graph $L_G = (V, L_E)$ with same set of vertices than G such that $dH(G, L_G) \leq k$?

We conjecture that Problem Proxi-Line is NP-complete. This problem generalizes one of the problems defined about hereditary graph properties on induced subgraphs (to be a line-graph is such a property) and proved to be NP-complete in [17]: given a graph G and an integer k, does there exist a line-graph L_G that is a spanning subgraph of G and such that $dH(G, L_G) \leq k$ (i.e., Problem Proxi-line in which only edge deletion is allowed). The NP-completeness proof given in [17] uses a polynomial reduction from the NP-complete Hamiltonian path problem. An integer programming solution for this problem has been given in [8]. We prove here the following complexity property concerning Problem Proxi-line in which only edge addition is allowed.

Theorem 1 *Given a connected graph G and an integer k, the problem of knowing if there exists a line-graph L_G such that G is a spanning subgraph of L_G and such that $dH(G, L_G) \leq k$ is NP-complete.*

Proof:

Determining whether a graph is a line-graph or not is a polynomial problem [12]. Consequently, the target problem is in NP.

We now show a reduction from 3-SAT: considering a CNF-formula φ with n variables $\{x_1, x_2, \ldots, x_n\}$ and m clauses $\{C_1, C_2, \ldots, C_m\}$ containing 3 literals each, is there a truth assignment of the variables satisfying φ?

Let $\mathcal{I} = (\{x_1, x_2, \ldots, x_n\}, \{C_1, C_2, \ldots, C_m\})$ be an instance of 3-SAT. We build an instance $\mathcal{J} = (G = (V, E), k)$ of Problem Proxi-Line restricted to edge addition only from $\mathcal{I}$.

We set $k = (4mn + nm(m + 3) + 12m + 6m)^2$. This reduction contains a gadget for each variable and a gadget for each clause.

For each variable x_i, we add a clique A_{x_i} with $4m$ nodes: $\{x_i^j, \bar{x_i}^j, x_i^{j\star}, \bar{x_i}^{j\star}, j \in [\![1; m]\!]\}$. For each clause C_j containing the literal x_i, we add a clique $B_{x_i}^j$ containing $m + 3$ nodes: the node x_i^j from the clique A_{x_i}, two nodes $o_{x_i}^{j1}$ and $o_{x_i}^{j2}$

and a set of m nodes $\{p_{x_i}^{jj'}, j' \in [\![1;m]\!]\}$. We similarly create a clique $B_{\bar{x_i}}^j$ for each clause C_j containing the literal $\bar{x}_i$. Finally, for each variable x_i and each couple of clauses C_j and $C_{j'}$ such that the first one contains x_i and the second one contains $\bar{x}_i$, we build a clique of size $k+1$ containing the two nodes $p_{x_i}^{jj'}$ from the clique $B_{x_i}^j$ and $p_{\bar{x_i}}^{j'j}$ from the clique $B_{\bar{x_i}}^{j'}$ and $k-1$ new nodes. This first gadget is illustrated on Figure 1.

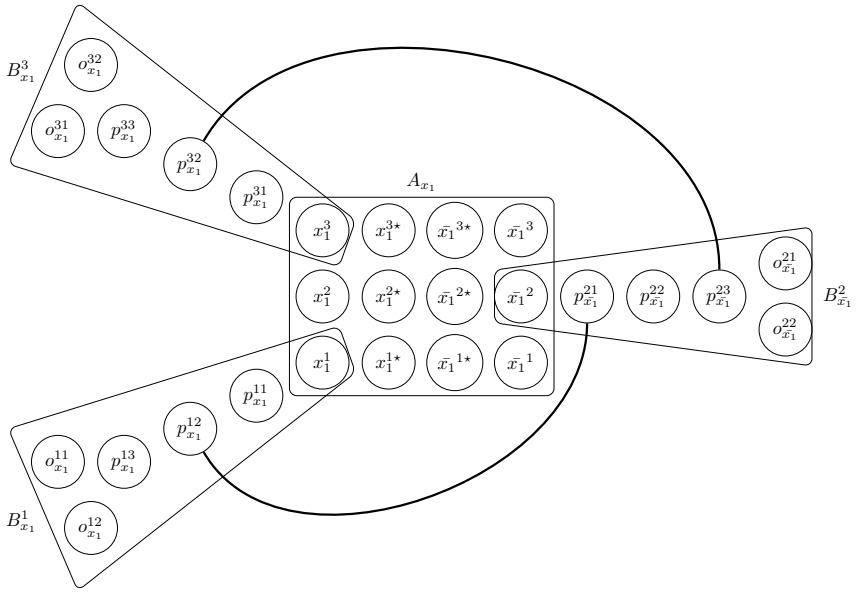


Figure 1: This graph is the gadget associated with the variable x_1 in the following 3-SAT instance: $\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\bar{x_1} \vee \bar{x_2} \vee x_3) \wedge (x_1 \vee \bar{x_2} \vee \bar{x_3})$. For readability, the edges of the cliques are not drawn. The thick curved edges are the cliques with $k+1$ nodes.

A second gadget is built for each clause $C_j = (l_1 \vee l_2 \vee l_3)$. We add 3 cliques C_j^{12}, C_j^{13} and C_j^{23}, each one contains four nodes: C_j^{12} contains the two nodes $o_{l_1}^{j1}$ from $B_{l_1}^j$ and $o_{l_2}^{j2}$ from $B_{l_2}^j$, and two new nodes q_j^{121} and q_j^{122}. We similarly create the cliques C_j^{13} and C_j^{23}. We then add three cliques of size $k+1$, the first one contains q_j^{121} and q_j^{132}, the second one contains q_j^{131} and q_j^{232} and the third one contains q_j^{231} and q_j^{122}, each clique is completed with $k-1$ new nodes.

We then add a single clique D_j with 6 nodes: the nodes l_1^j, l_2^j and l_3^j from the cliques A_{l_i} or $A_{\bar{l_i}}$ and three new nodes r_1^j, r_2^j and r_3^j. We finally build three cliques of size $k+1$ containing $l_i^{j\star}$, r_i^j and $k-1$ new nodes for each $i \in \{1,2,3\}$. This second gadget is illustrated on Figure 2.

We first explain what should a solution of the instance $\mathcal{J}$ looks like. Only the nodes x_i^j (resp. $\bar{x_i}^j$) such that the corresponding literal is in the clause C_j

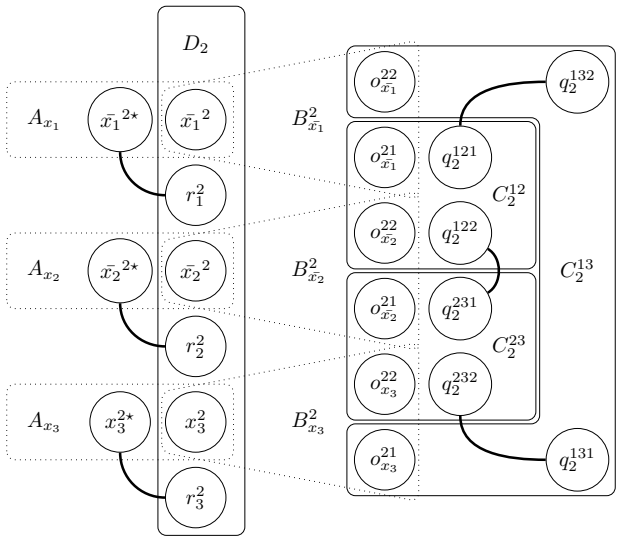


Figure 2: This graph is the gadget associated with the second clause C_2 in the following 3-SAT instance: $\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\bar{x_1} \vee \bar{x_2} \vee x_3) \wedge (x_1 \vee \bar{x_2} \vee \bar{x_3})$. For readability, the edges of the cliques are not drawn. The thick edges are cliques with $k+1$ nodes including the two nodes at the extremities.

are contained in three cliques in G: the cliques A_{x_i}, $B^j_{x_i}$ (resp. $B^j_{\bar{x_i}}$) and D_j. By Property 1, every node should be covered by at most two cliques. As we can only add edges to build L_G, the only way to satisfy the property is to merge at least two of the three cliques into a bigger clique by adding the missing edges. Indeed, not doing this would leave at least three disconnected neighbors of x_i^j (resp. $\bar{x_i}^j$), and thus no way to cover this node with at most two cliques. If C and C' are two cliques, we write $C \times C'$ the clique obtained by adding all the edges between the nodes of C and the nodes of C'.

However, assuming x_i is the first literal of C_j, the nodes $x_i^{j\star}$ from A_{x_i} and r_1^j from D_j belong to the same clique of size $k+1$. Let F be that clique. If we decide to merge the cliques A_{x_i} and D_j into a single clique $A_{x_i} \times D_j$, we would obtain a graph where the cliques F and $A_{x_i} \times D_j$ cover the same edge $(x_i^{j\star}, r_1^j)$. However, by Property 1, each edge should be covered by only one clique. Those cliques must then be merged too. But, if L_G contains the clique $F \times A_{x_i} \times D_j$, then $dH(G, L_G) \geq k+1$. As a consequence, the cliques A_{x_i} and D_j are not merged in a solution of $\mathcal{J}$. Thus, to obtain L_G, we can either merge $B^j_{x_i}$ with A_{x_i} or with D_j.

Similarly, it is not possible to merge a clique $B^j_{x_i}$ with a clique $B^{j'}_{\bar{x_i}}$ as those cliques intersect a same clique with $k+1$ nodes too. Consequently, we cannot merge $B^j_{x_i}$ with A_{x_i} and merge $B^{j'}_{\bar{x_i}}$ with A_{x_i}. It is then possible to decide which variable is true: for each variable x_i, we set x_i to true if and only if $B^j_{x_i}$ is merged with A_{x_i} for some j.

This is where the clauses D_j, C_j^{12}, C_j^{13} and C_j^{23} intervene. Let $C_j = (l_1 \vee l_2 \vee l_3)$ be a clause. We assume that we merge the three cliques $B_{l_1}^j$, $B_{l_2}^j$ and $B_{l_3}^j$ with D_j, which would mean that the clause is not satisfied. By doing this, we get a clique containing $o_{l_i}^{j1}$ and $o_{l_i}^{j2}$ for each $i \in [\![1;3]\!]$. All those couples of nodes are contained in the cliques C_j^{12}, C_j^{13} and C_j^{23}, therefore we also have to merge the super-clique $B_{l_1}^j \times B_{l_2}^j \times B_{l_3}^j \times D_j$ with those three cliques. As the cliques C_j^{12}, C_j^{13} and C_j^{23} intersect the same cliques of size $k+1$ with the nodes $q_j^{121}, q_j^{122}, \ldots, q_j^{232}$, this would lead to a graph L_G with $dH(G, L_G) \geq k+1$. Consequently, for every clause, it is not possible to merge the three cliques $B_{l_1}^j$, $B_{l_2}^j$ and $B_{l_3}^j$ with D_j

Therefore, if there exists a line-graph L_G such that G is a spanning subgraph of G and $dH(G, L_G) \leq k$, the two previous paragraphs prove we can build a truth assignment of the formula.

We now assume φ can be satisfied. For every true variable x_i, we merge the cliques $B_{x_i}^j$ with A_{x_i} for all j such that x_i appears in C_j and all cliques $B_{\bar{x_i}}^{j'}$ with $D_{j'}$. We similarly act with every false variable x_i by merging $B_{\bar{x_i}}^j$ with A_{x_i} and $B_{x_i}^{j'}$ with $D_{j'}$.

First, no two cliques $B_{x_i}^j$ and $B_{\bar{x_i}}^{j'}$ are merged with A_{x_i}. Secondly, for each clause $C_j = (l_1 \vee l_2 \vee l_3)$ the three cliques $B_{l_i}^j$, $B_{l_2}^j$ and $B_{l_3}^j$ are not all together merged with D_j. We are then not in the case mentioned above, there is no need to merge more than one of the cliques C_j^{12}, C_j^{13} or C_j^{23} with D_j. We then get a line-graph where no clique of size $k+1$ is merged with another clique. As there are at most $4mn + nm(m+3) + 12m + 6m = \sqrt{k}$ nodes in the rest of the graph, we add at most k edges. Thus L_G is a connected line-graph such that G is a spanning subgraph of L_G and $dH(G, L_G) \leq k$. This concludes the proof of the theorem.

QED

Knowing that Problem Proxiline is NP-complete when only the addition of edges or the deletion of edges is allowed does not imply that it remains NP-complete when the addition and deletion are simultaneously authorized, since these two operations could complement each other and thus make the problem polynomial. However, we conjecture that the problem remains NP-complete in this case.

3 Solving Proxi-Line problem

In this section, we describe and analyze two algorithms to be consecutively used to provide a line-graph L_G from an input graph G with $dH(G, L_G)$ as small as possible. These algorithms are in particular based on some specific line-graph properties we first focus on.

3.1 Line-graph properties

The first proposed algorithm consists in trying to obtain a line-coverage of the input graph G to determine a root graph, if G is a line-graph. Except for the complete graph K_3, it has been shown that each connected line-graph has only one root graph, up to an isomorphism [16]. Such an isomorphism corresponds to different possible line-coverages for a same line-graph. As a direct consequence of the proof given in [16], the line-graphs admitting different line-coverages are the ones given in Figure 3. Such a graph is here called an **ambiguity**.

Given a graph $G = (V, E)$, we denote $G[V']$ the subgraph of G induced by a subset V' of V (see [2]). A node u such that $\{u\} \cup \Gamma_G(u)$ (with $\Gamma_G(u)$ the neighborhood of u in G) can be covered by two cliques in two different ways is called an **ambiguity-anchor**. Note that in each ambiguity given in Figure 3 each vertex is an ambiguity-anchor. Some different line-coverages of some such line-graphs are given in Figure 4.

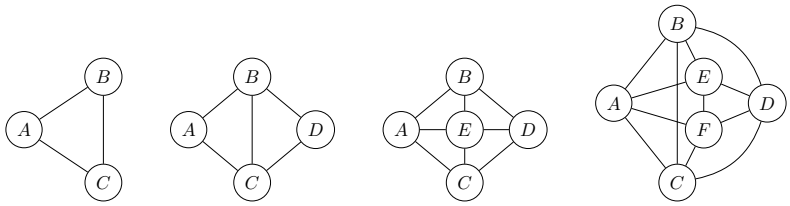


Figure 3: Graphs with two possible line-coverages.

If a vertex u in a line-graph G is an ambiguity-anchor, either G is isomorphic to an ambiguity in Figure 3, or only one of the two possible partitions of $\{u\} \cup \Gamma_G(u)$ in two cliques allows a line-coverage of G.

Let G be a graph and u a vertex in G. A partition of $\Gamma_G(u)$ in two cliques C_{u1}, C_{u2} is **consistent** if and only if each vertex v of C_{u1} (resp. C_{u2}) has at most one neighbor in C_{u2} (resp. C_{u1}). We can conclude the following property from these definitions and the results of [16] with reference to Figure 3.

Lemma 1 *Let G be a line-graph, u a vertex in G and a consistent partition of $\{u\} \cup \Gamma_G(v)$ in two cliques C_{u1}, C_{u2}. If the cardinality of one of these two cliques is greater than or equal to 4, then this consistent partition is unique.*

In all the following, we consider that G is not isomorphic to a graph of Figure 3 (this can easily be verified).

3.2 Heuristics to solve Proxi-Line problem on small degree graphs

In this section, we are interested in the minimization problem associated with Proxi-Line, i.e., given a graph G the determination of the minimum Hamming distance between G and a line-graph L_G with same vertex-set. We consider

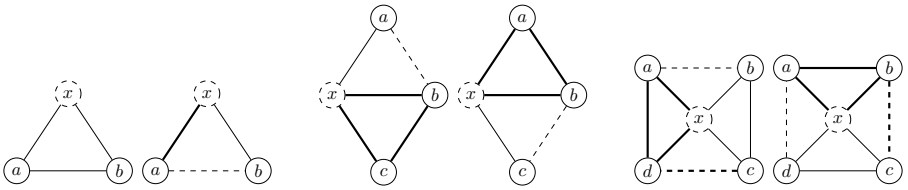


Figure 4: Ambiguities with possible line-coverages represented by solid, thick, dashed, and dashed thick cliques; nodes "x" are chosen ambiguity-anchors.

here a graph with small bounded degrees, which corresponds to most electrical distribution networks.

In order to deal with this problem, we consider an approach consisting in the execution of two consecutive algorithms: a covering algorithm, trying to obtain a line-coverage of the target graph, and, if it fails, a correction algorithm that adds and deletes edges in the target graph to obtain a line-graph. After the first algorithm proposed in [12] to determine if a given graph is a line-graph some other algorithms with linear complexity have also been defined (see [13] and references). In our context if G is not a line-graph, the goal of the covering algorithm is to cover as much as possible edges and vertices of G with cliques respecting Property 1, to minimize as far as possible the number of edges to be added or removed by the correction algorithm.

3.2.1 Covering algorithm

Given graph $G = (V, E)$, the following algorithm aims to determine a line-coverage if it exists, i.e., if G is a line-graph. If not, the algorithm will cover as much as possible vertices of G with one or two cliques, with each edge contained in at most one such clique.

In this algorithm, each vertex $v \in V$ is associated to an evolving status $Cliq(v)$ initialized to 0. This function gives the state of each vertex during the execution of the algorithm: $Cliq(v) = 0$ says that v has not been considered yet, $Cliq(v) = 1$ says that v is contained in one clique (i.e., this clique contains v and all its neighbors), $Cliq(v) = 2$ says that v is contained in two cliques (obtaining together all its neighbors), $Cliq(v) = 3$ says that v is contained in one clique and has still some neighbors not contained in cliques, and $Cliq(v) = -1$ says that the algorithm cannot cover v by only one or two cliques.

We execute Algorithm 1 on an input graph G not in those of Figure 3. An example of such an execution is given in Figure 5.

Note that, if G is a line-graph not isomorphic to an ambiguity, the vertex u chosen between Lines 3 and 11 of Algorithm 1 exists at each step of the execution. Then any such choice leads to a unique correct consistent partition, since there is only one possible line-coverage of G. We can indeed show by induction on the set of vertices that at each stage, there is a vertex

Algorithm 1 The covering algorithm

```
1: C ← ∅
2: loop
3: |   U_0 ← the set of vertices u such that Cliq(u) = 0 and u is not an ambiguity-anchor
4: |   U_3 ← the set of vertices u such that Cliq(u) = 3
5: |   if U_0 ≠ ∅ then
6: |   |   Choose any vertex u in U_0
7: |   else if U_3 ≠ ∅ then
8: |   |   Choose any vertex u in U_3
9: |   else
10: |  |   Set Cliq(u) ← −1 for every vertex u with Cliq(u) = 0
11: |  |   Quit the loop
12: |  if {u} ∪ Γ_G(u) can be covered by two cliques C_1 and C_2 with C_1 being maximal and C_1 and C_2 being consistent if C_2 is not empty then
13: |  |   if Cliq(u) = 3 and C_2 ≠ ∅ then
14: |  |   |   Cliq(u) ← −1
15: |  |   else
16: |  |   |   { Checking the neighbors of u in the two cliques }
17: |  |   |   for each w neighbor of u do
18: |  |   |   |   if there exists a neighbor of w not in C_1 ∪ C_2 then
19: |  |   |   |   |   if Cliq(w) = 3 then Cliq(w) ← −1
20: |  |   |   |   |   else if Cliq(w) = 0 then Cliq(w) ← 3
21: |  |   |   |   else
22: |  |   |   |   |   if Cliq(w) = 3 then Cliq(w) ← 2
23: |  |   |   |   |   else if Cliq(w) = 0 then Cliq(w) ← 1
24: |  |   |   { Updating the status of u }
25: |  |   |   if Cliq(u) = 0 and C_2 is empty then Cliq(u) ← 1
26: |  |   |   else Cliq(u) ← 2
27: |  |   |   { Removing the two cliques from G }
28: |  |   |    Remove each edge in E between two vertices in C_1 and two vertices in C_2
29: |  |   |   Add C_1 and C_2 (if not empty) to C
30: |  else
31: |  |   Cliq(u) ← −1
32: return C
```

- not contained in any clique and such that its remaining neighborhood can be covered by one or two new cliques ($Cliq(v) = 0$, see steps 1 and 2 in Figure 5),
- contained in a clique already in $\mathcal{C}$ and such that its remaining non covered neighborhood can be covered by a new clique ($Cliq(v) = 3$, see steps 3 and 4 in Figure 5).

Thus, if the initial graph G is a line-graph then the covering algorithm deter-

mines a line-coverage.

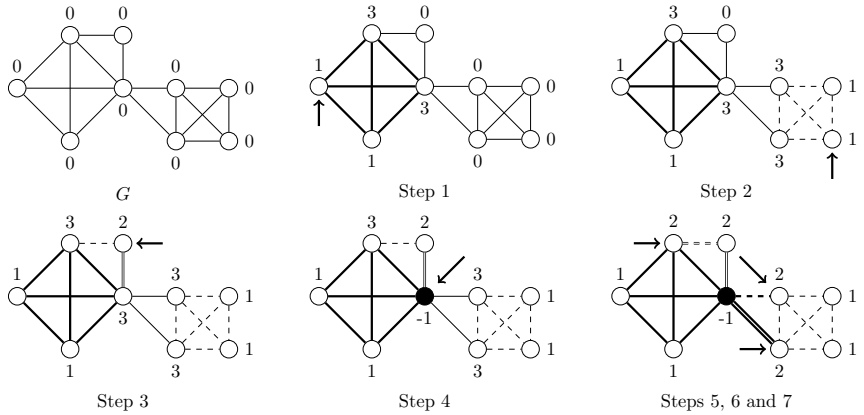


Figure 5: Steps of execution of the covering algorithm on a graph. For each vertex v the value of $Cliq(v)$ is given. At each step, the selected vertex v is designated by an arrow. Each dashed, thick and/or double edge corresponds to a clique C_1 or C_2 added to $\mathcal{C}$ in Algorithm 1

If the input graph G is not a line-graph, then at some step there is a vertex such that it is not possible to cover it by two cliques, or there are two ways to cover it by two cliques. In this last case, we arbitrarily choose one of these two coverings (this will only impact the correction algorithm). Note that we do the same if G is not connected, even if some connected components are isomorphic to a graph in Figure 3, since the final target line-graph to be obtained has to be connected.

Concerning the complexity, searching all maximal cliques in the subgraph is usually a complex problem, solved in [7] in time $O(n\Delta(G)3^{\Delta(G)/3})$, but in our context, we just need to determine for each vertex v in G if v can be contained in one or two cliques. This can be done with complexity $O(\Delta(G)^2)$ (i.e., by determining if the chromatic number of the complementary graph is 1 or 2). Thus the complexity of the covering algorithm is at worst $O(n \times \Delta(G)^2)$ with n the number of vertices. Recall that the algorithm of [12] has a complexity in $O(n \times \Delta(G))$, but if the graph is not a line-graph, it does not provide a partial line-coverage.

3.2.2 Correction algorithm

If the initial graph $G = (V, E)$ is not a line-graph, the covering algorithm provides a partial line-coverage $\mathcal{C}$. In this case, we denote by $\mathcal{Z}$ the set of vertices v such that $Cliq(v) = -1$ and then we execute the correction algorithm given in this section. In the example of Figure 5, even though there is a vertex v such that $Cliq(v) = -1$, each edge is covered by a clique; this is not always the case.

Thus if not, for each edge $[u, v]$ of G not covered by a clique in $\mathcal{C}$, we add the set $\{u, v\}$ in $\mathcal{C}$ (but u and v stay in $\mathcal{Z}$).

Let z be a vertex in $\mathcal{Z}$. For each such a vertex z the objective of the correction algorithm is to add and delete edges such that z can then be contained in one or two cliques, without loosing this same property for each vertex v not in $\mathcal{Z}$.

We consider first two types of cliques in $\mathcal{C}$ related to the correction of a vertex $z \in \mathcal{Z}$.

- A **local clique** of z is a clique in $\mathcal{C}$ containing z; we denote $\mathcal{C}(z)$ the set of such cliques.
- A **neighbored clique** of z is a clique $c \in \mathcal{C} - \mathcal{C}(z)$ such that there exists at least two local cliques c_1 and c_2 of $C(z)$ with $|c \cap c_1| = 1$ and $|c \cap c_2| = 1$. We denote by $\mathcal{N}(z)$ the set of neighbored cliques of z.

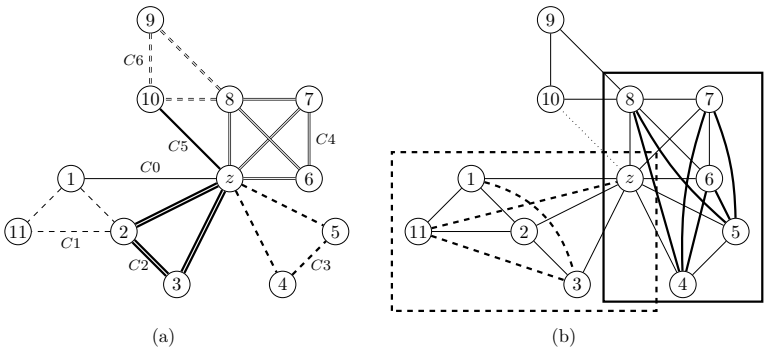


Figure 6: (a) An example of local and neighbored cliques of a node z and (b) an example of compression, with π_1 the dashed rectangle, π_2 the solid one and $\pi_s = \{10\}$. A correction is done by adding the thick (dashed and solid) edges and by removing the dotted edge.

In the example of Figure 6(a), the set $C(z)$ contains the cliques $C0$, $C2$, $C3$, $C4$ and $C5$; the set $\mathcal{N}(z)$ contains $C1$ and $C6$. Indeed, $C1$ has a common vertex with $C0$ and another with $C2$, and $C6$ has a common vertex with $C4$ and another with $C5$.

The purpose of the algorithm is to replace some cliques in $\mathcal{C}(z) \bigcup \mathcal{N}(z)$ by a unique clique (by adding the missing edges in the graph), or to remove the edges linking z to all the vertices in some local cliques. For this purpose, we define a property for a subset of cliques to determine whether or not those cliques can merged into one super-clique.

Definition 2 *A subset of cliques $S \subseteq \mathcal{C}(z) \cup \mathcal{N}(z)$ is* **extensible** *for z if and only if for each edge $[u, v]$ in the graph such that u and v are in two different cliques in S, set $\{u, v\}$ is a clique in $\mathcal{C}$ or $\{u, v\}$ is a subset of a clique in S.*

Applying an extension from such an extensible set of cliques S consists in adding the union of these cliques to $\mathcal{C}$, in removing from $\mathcal{C}$ any clique covered by this union, and in adding in G the missing edges between pairs of vertices in this union. We thus obtain a new clique in G. In the example of Figure 6(a) the subset $\{C4, C5\}$ is not extensible because the edge $[8, 10]$ is neither a clique of $\mathcal{C}$ (it is strictly included in $C6$) nor covered by $C4$ or $C5$, but the subset $\{C4, C5, C6\}$ is extensible.

The set $\mathcal{C}$ obtained after such an extension is still a partial line-coverage of the modified graph since by Definition 2, each edge of this graph is still covered by at most one clique. Moreover, each vertex initially contained in one or two cliques still has this property.

By using only extensible sets, we do not take advantage of edge deletions. We now define a second concept to correct a node using addition and deletion of edges at the same time.

Definition 3 *A* **compression** *of z as a triple (π_1, π_2, π_S) where*

- *π_1 and π_2 are the vertex sets of two extensible sets S_{π_1} and S_{π_2} in $\mathcal{C}(z) \cup \mathcal{N}(z)$*
- *π_1 and π_2 cannot simultaneously be equal to $\{z\}$ and $\pi_1 \cap \pi_2 = \{z\}$,*
- *π_S is the subset of neighbors of z not in $\pi_1 \cup \pi_2$,*
- *the set of edges $\{[u, z] \in E : u \in \pi_S\}$ is not disconnecting G after applying extensions of S_{π_1} and S_{π_2}.*

In the example of Figure 6(b), the considered compression is $\pi_1 = C0 \cup C1 \cup C2$, $\pi 2 = C3 \cup C4$ and $\pi_S = \{10\}$.

Note that there always exists such a compression. Let $\{CC_1, \ldots, CC_p\}$ be the p connected components in G/z, the graph obtained by removing z from G, and, for any $1 \leq i \leq p$, let $C_i \in \mathcal{C}(z)$ be a clique in CC_i. Then since there is no edge between two different cliques C_i and C_j, with $i \neq j$, the set $\{C_1, \ldots, C_p\}$ is extensible. Consider $\pi_1 = \bigcup_{1 \leq i \leq p} C_i$, $\pi_2 = \{z\}$ and π_S be the subset of vertices in $\Gamma_G(z)$ not in π_1. Then (π_1, π_2, π_S) is a compression. Indeed, consider $v \in \pi_S$ and let CC_v be the connected component of G/z containing v; then for each vertex v' in CC_v, there exists a chain between v and v' in G/z. Thus, removing the edge between z and v does not disconnect G.

Applying such a compression $T = (\pi_1, \pi_2, \pi_S)$ consists in applying extensions from S_{π_1} and S_{π_2} and deleting all edges $[z, x]$ with $x \in \pi_S$. More precisely, it consists in

1. Adding to E edges $[u, v]$ such that u and v are not adjacent vertices in π_1 (resp. π_2) ,
2. Removing from E all edges $[z, v]$ with $v \in \pi_S$,

3. Deleting all cliques covered by π_1 and the ones covered by π_2 in $\mathcal{C}$, and adding π_1 and π_2 in $\mathcal{C}$,

4. Assigning $Cliq(z)$ to 1 (if π_1 or $\pi_2 = \{z\}$, meaning that z is now contained in one clique in $\mathcal{C}$) or 2 (meaning that z is contained in two non empty cliques π_1 and π_2).

Note that, by definition, for each vertex $x \in \pi_S$, the vertices in the clique of $\mathcal{C}(z)$ containing z and x is a subset of $\{z\} \cup \pi_S$. Then after applying the compression T, each vertex in π_S initially contained in one or two cliques still has this property. In addition, the vertex z is contained in one or two cliques in $\mathcal{C}$: it is corrected. The **cost** $c(T)$ of a compression $T = (\pi_1, \pi_2, \pi_S)$ is the number of edges added and removed when applying extensions from S_{π_1} and S_{π_2} and deleting edges related to π_S. More precisely,

$$c(T) = |\{\{u,v\} \subseteq \pi_1 : \ [u,v] \notin E\}| \ + \ |\{\{u,v\} \subseteq \pi_2 : \ [u,v] \notin E\}| \ + \ |v \in \pi_S|$$

The compression given in Figure 6 (b) has cost 10, i.e., the number of thick (dashed or solid) edges plus the dotted removed edge $[10, z]$.

Thus, the correction algorithm is the following: while there exists a vertex z such that $Cliq(z) = -1$, choose a compression $T = (\pi_1, \pi_2, \pi_S)$ for z such that $c(T)$ is minimum, and apply it. Note that the correction of such a vertex z could also imply the correction of some other vertices z' such that $Cliq(z') = -1$, i.e., the optimal compression for them will have then cost zero.

In terms of complexity, the correction algorithm treats each vertex of the graph at most once. Furthermore $|\mathcal{C}(z)| \leq \Delta(G)$, $|\mathcal{N}(z)| \leq \Delta(G)/2$ and the cardinal of each of these cliques is less or equal to $\Delta(G)$, the maximum degree of a vertex in G. In the worst case, the algorithm must check if each subset of $\mathcal{N}(z) \cup \mathcal{C}(z)$ is extensible. Thus, the complexity of computing a minimum cost compression is less than $O(2^{\Delta(G)} \times \Delta(G)^4)$. Note that this upper bound is rarely reached in practice, as we will see in the next section; for example, the number of cliques $|\mathcal{C}(z)|$ and the average size of these cliques are inversely proportional, thus they can not simultaneously reach their respective upper bounds. Moreover, in theory, the maximum degree of the graph could increase after each correction, but here again in practice, this maximum degree does not really increase, in particular when the final obtained hamming distance with the initial graph is low. Thus, this algorithm is realistic for corrupted line-graphs of distribution network since corresponding graphs have usually a small degree.

4 Performance evaluation

We evaluate the performances of the covering and correction algorithms by considering randomly generated graphs. To do this, we first generate 500 undirected

connected graphs H with 25 vertices and a probability $\frac{1}{5}$ for an edge to occur between each pair of vertices. Note that if the obtained graph is not connected, we also randomly add some edges between the connected components to connect them. The finally obtained graphs have thus an average degree near to 5 (recall that we focus on small degree graphs). Given such a graph H, we consider $L(H)$ the line-graph of H, with $M_{L(H)}$ its adjacency matrix.

We randomly choose and modify k elements of $M_{L(H)}$, where $k \geq 0$ is a parameter of the experimentation. For each element, we first decide with probability $\frac{1}{2}$ if we pick a zero or a one in the matrix, then we pick it uniformly among all the possible elements (in other words, we choose on average $\frac{k}{2}$ zeros and $\frac{k}{2}$ ones in the matrix). We then replace the value x of each such chosen element with $1 - x$, which corresponds to the addition or removal of k edges in $L(H)$. Let G be the obtained graph; then $dH(L(H), G) = k$. Note that, if $k > 0$, G can be a line-graph but not the line-graph of H. We will evaluate the performances of each algorithm in terms of the number of corrections as a function of k.

For each generated graph H and each tested value of k, we consider 5 modified graphs G from $L(H)$. On each one we then execute consecutively the covering and correction algorithms. Note that at each execution step of the correction algorithm, the vertex z is chosen in $\mathcal{Z}$ in a uniform random manner (the choice of a vertex z minimizing the cost having proved experimentally less effective on average). Thus for each k we consider 500×5 executions of the algorithms.

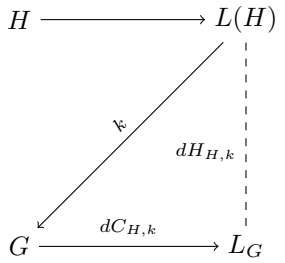


Figure 7: Graphs and process of the performance evaluation

Let L_G be the line-graph obtained from each G after the execution of the covering then correcting algorithms (see Figure 7). We consider $dC_{H,k}$ the average of the five measured distances $dH(G, L_G)$ corresponding to the five modified graphs G from each $L(H)$ and $dH_{H,k}$ the average of the five measured distances $dH(L(H), L_G)$, for each G and each k. Thus, ideally if $dC_{H,k} = k$

then $dH_{H,k} = 0$; note that if $dC_{H,k} < k$ for some G then it means that the algorithms have found a line-graph L_G different from $L(H)$ with less than k corrections of edges. To evaluate the correlation between these two parameters $dC_{H,k}$ and $dH_{H,k}$, we also consider a coefficient measuring the proportion of the k edges initially added or removed in L_G that the algorithm corrects compared to the total number of edges it adds or removes. More precisely,

$$corr_{H,k} = \frac{|dH_{H,k} - |dC_{H,k} - k||}{dC_{H,k} + k}$$

Thus, the more the correction algorithm corrects the k initial modifications in $M_{L(H)}$, the more the coefficient tends to zero. Indeed, if the correction algorithm does not correct any of these k initial modifications then the Hamming distance between L_G and $L(H)$ is $dC_{H,k} + k$ (recall that $dH_{H,k} \leq dC_{H,k} + k$). Likewise, the closer both the correction number $dC_{H,k}$ and the number of corrected initial modifications are to k, the closer $|dH_{H,k} - |dC_{H,k} - k||$ is to zero.

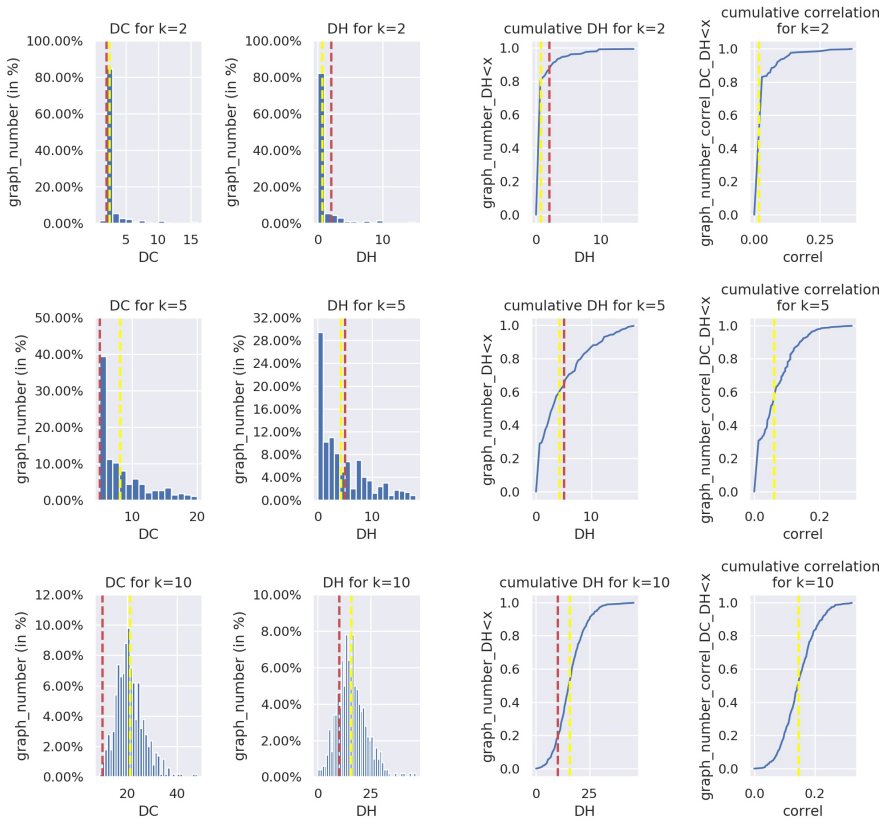


Figure 8: Distributions for $k \in \{2, 5, 10\}$.

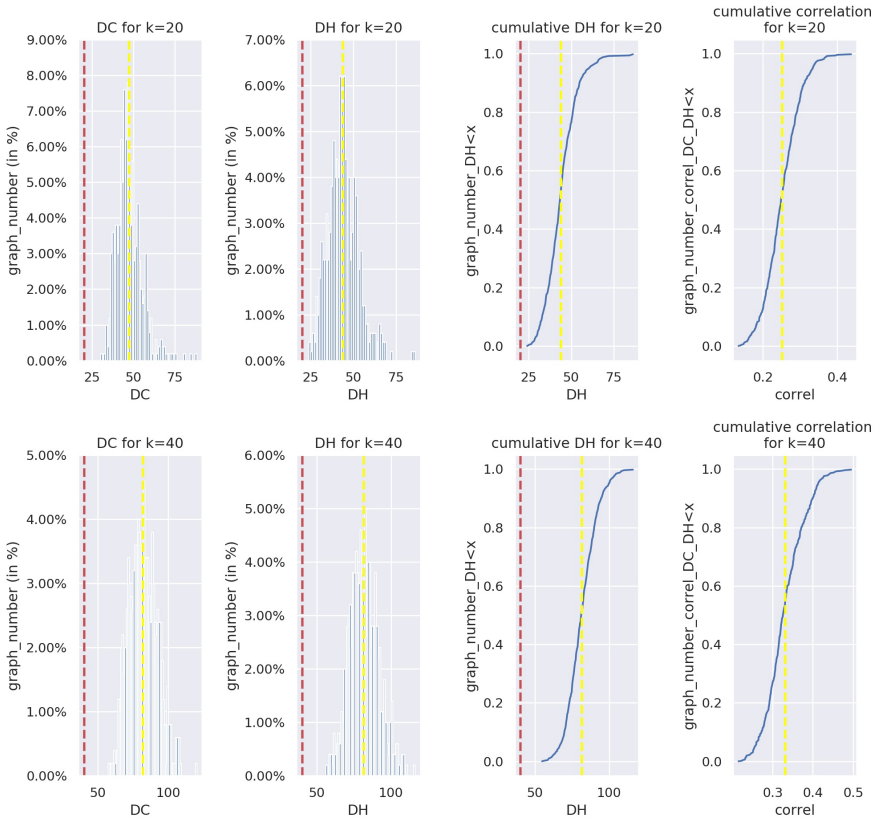


Figure 9: Distributions for $k \in \{20, 40\}$.

Figures 8 and 9 show the distributions of the values of $dC_{H,k}$, $dH_{H,k}$, cumulative $dH_{H,k}$ and $corr_{H,k}$ respectively, for $k \in \{2, 5, 10, 20, 40\}$. The dash red lines represent the values of k and the yellow dash lines the average of the considered values.

For $k = 1$ and for each tested graph H we have $dC_{H,1} = 1$ and $dH_{H,1} = 0$ (i.e., L_G is $L(H)$). When $k \geq 2$ the results given in Figure 8 show that for small values of k, the algorithms determine Hamming distances that are mostly equal or very close to the minimum value, correcting the k initial modifications. Indeed for $k < 10$ the results show a difference between $dC_{H,k}$ and $dH_{H,k}$ which shows that the corrections made by the algorithms tend to bring L_G closer to $L(H)$. This is no longer the case when k increases. This can be simply explained by the fact that the closest line-graph is not $L(H)$ anymore. The two averages of the values of $dC_{H,k}$ and $dH_{H,k}$ are similar (close to $2k$) and the shape of the cumulative correlation curves stabilizes. In fact, it can be seen that the larger k is, the lower the proportion of corrections made by the algorithms among the initial modifications of k are, but the link is almost linearly. Note that, if

$dH_{H,k}$ is the average value $2k$ and if none of the modified edges/non-edges were repaired, then $dC_{H,k} = k$ which means that the correction algorithm may have found a solution close to the optimal solution and that this optimal solution is not $L(H)$. We also found that the lower the average size of cliques given by the covering algorithm (between 2 and 3) is, the larger $\mathcal{Z}$ and the higher the average value of $dH_{H,k}$ are.

Moreover as also shows Figure 10(b), when k increases, the ratios between the average value of $dH_{H,k}$ and k is less than 3 for each $k \in \{1, 5, 10, 20, 40\}$, in particular less than 2 for $k = 40$ (remember that the number of edges of each generated line-graph is on average equal to 310, so that $k = 40$ corresponds to 10% of edge modifications). Similarly, if the average correlation increases only slowly over 0.3 when k increases.

These experiments show that for small degree graphs, the performances of the algorithms proposed are effective for the resolution of the Proxiline problem for small values of k and that these performances deteriorate slowly when k increases; these results thus show the relevance of a needed study of possible approximation properties of the Proxiline problem for bounded degree graphs.

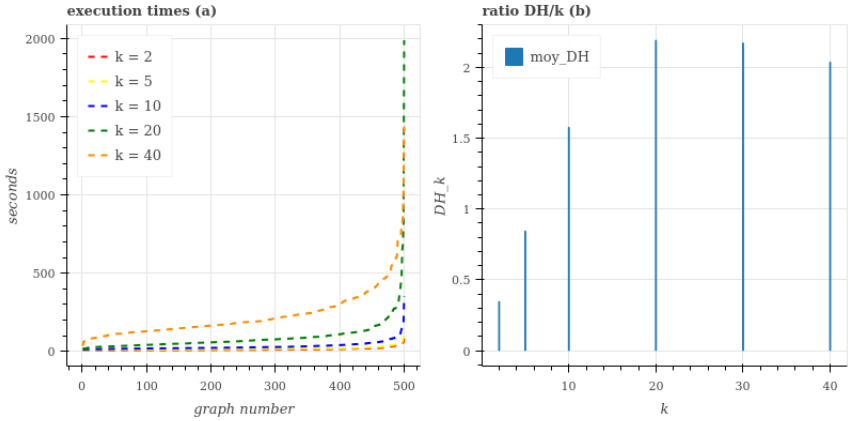


Figure 10: (a) Distribution of the 500 average execution times of the algorithms (one for each graph H) in increasing order, for the different values of k. (b) For each value of k, the average value of $dH_{G,k}$ divided by k.

Figure 10(a) also shows that the execution times of the algorithms are quite homogeneous and reasonable for most of the graphs tested, and that the average execution time increases slowly when k increases. Note that these experiments took about a day to run on a personal computer, which shows that the proposed algorithms can be used for realistic network topologies. Moreover, the longer the execution time that a graph requires, the greater the number of cliques covering it after execution of the correction algorithm is, and the smaller these cliques are. So unsurprisingly, the higher $dC_{H,k}$ and $dH_{H,k}$ are, the longer the running time is, as shown in Table 1 for six graphs H and $k = 2$. Note finally

that for only one graph among the 500 considered, the running time is exceptionally long (2000 seconds for $k = 20$); the topology of the graph causes the correction algorithm to correct a large number of small cliques, but the value $dC_{H,k}$ obtained is not exceptionally large. Such cases seem rare.

Table 1: Average execution times, $dC_{H,k}$, $dH_{H,k}$ and number of edges for 5 graphs H and $k = 2$

$dC_{H,k}$	$dH_{H,k}$	**Number df edges**	**Runtime (s)**
139	141	434	7.507
245	247	434	7.773
326	328	461	25.613
336	338	475	27.57
384	386	410	144.02
369	371	394	137.044

Some other experiments have shown that executing again the covering algorithm after each correction phase on a vertex in $\mathcal{Z}$ does not improve the performances. Moreover, similar experiments have been done considering no more the number of the edges for the cost of a compression but the sum of the corresponding correlation values, for real data of measurements within Data Centers. The behavior of the algorithms remained similar to those described in this article [6].

5 Conclusion

The contributions of this article have a dual purpose. First, to propose an effective heuristic approach to solving the Proxiline problem, for which we have shown a new complexity result. Then, to evaluate the impact of the heuristic resolution of this problem for the problem of discovering the topology of an electrical distribution network with measurement errors. We have proposed an approach consisting of executing two algorithms consecutively, an approach which is relatively effective for small degree graphs. Experiments show that these algorithms achieve good performance in solving the Proxiline problem. These experiments also show that this resolution only makes it possible to respond to the problem of discovering topologies only for a very small number of correlation errors. On this second point, it is therefore interesting to determine to what extent partial knowledge of the topology would improve the quality of this discovery. Another improvement could be to search for the list of p-closest line-graphs instead of the optimal one and to let an expert decide if the topology matches one of them. Finally, even if the complexity of our approach is due to the correction algorithm, it could be interesting to determine a covering algorithm with linear complexity trying to cover as many vertices as possible.

References

[1] L. Beineke. Derived graphs and digraphs. *Beitrage zur Graphentheorie*, pages 17–23, 1968. URL: https://scholar.google.fr/scholar?q=L.W+Beineke.+Derived+graphs+and+digraphs.+Beitrage+zur+Graphentheorie.

[2] C. Berge. *The theory of graphs and its applications*. Methuen, 1962.

[3] D. Cvetkovic, P. Rowlinson, and S. Simic. *Spectral Generalizations of Line Graphs*. Cambridge University Press, Cambridge, 2004. URL: http://ebooks.cambridge.org/ref/id/CBO9780511751752, doi:10.1017/CBO9780511751752.

[4] D. G. Degiorgi and K. Simon. A dynamic algorithm for line graph recognition. In M. Nagl, editor, *Graph-Theoretic Concepts in Computer Science*, pages 37–48. Springer, Berlin, Heidelberg, 1995. URL: http://link.springer.com/10.1007/3-540-60618-1{_}64, doi:10.1007/3-540-60618-1_64.

[5] B. Donnet and T. Friedman. Internet topology discovery: a survey. *IEEE Communications Surveys & Tutorials*, 9(4):56–69, 2007. URL: http://ieeexplore.ieee.org/document/4444750/, doi:10.1109/COMST.2007.4444750.

[6] W. Ehounou. *Découverte de graphes par la dynamique de ses flots*. PhD thesis, Université de Versailles/ Université Paris Saclay, Oct. 2018.

[7] D. Eppstein, M. Löffler, and D. Strash. Listing all maximal cliques in sparse graphs in near-optimal time. In O. Cheong, K.-Y. Chwa, and K. Park, editors, *International Symposium on Algorithms and Computation*, pages 403–414, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. doi:10.1007/978-3-642-17517-6_36.

[8] B. V. Halldórsson, D. Blokh, and R. Sharan. Estimating population size via line graph reconstruction. *Algorithms for molecular biology : AMB*, 8(1):17, jul 2013. URL: http://www.ncbi.nlm.nih.gov/pubmed/23829669http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3716786, doi:10.1186/1748-7188-8-17.

[9] F. Harary. Line Graphs. In *Graph Theory*, chapter 8. Addison Wesley, 1972.

[10] P. Hines, S. Blumsack, E. C. Sanchez, and C. Barrows. The Topological and Electrical Structure of Power Grids. In *2010 43rd Hawaii International Conference on System Sciences*, pages 1–10. IEEE, 2010. URL: http://ieeexplore.ieee.org/document/5428366/, doi:10.1109/HICSS.2010.398.

[11] P. Lawton. Balancing the energy network. *Ingenia*, (53):20–26, 2012. URL: https://www.ingenia.org.uk/Ingenia/Articles/e6d3c924-241a-49cb-9352-a105f267c80d.

[12] P. G. H. Lehot. An Optimal Algorithm to Detect a Line Graph and Output Its Root Graph. *Journal of the ACM*, 21(4):569–575, oct 1974. URL: http://portal.acm.org/citation.cfm?doid=321850.321853, doi:10.1145/321850.321853.

[13] D. Liu, S. Trajanovski, and P. Van Mieghem. ILIGRA: An Efficient Inverse Line Graph Algorithm. *Journal of Mathematical Modelling and Algorithms in Operations Research*, 14(1):13–33, mar 2015. URL: http://link.springer.com/10.1007/s10852-014-9251-2, doi:10.1007/s10852-014-9251-2.

[14] J. Naor and M. B. Novick. An efficient reconstruction of a graph from its line graph in parallel. *Journal of Algorithms*, 11(1):132–143, mar 1990. URL: https://www.sciencedirect.com/science/article/pii/019667749090034C, doi:10.1016/0196-6774(90)90034-C.

[15] N. D. Roussopoulos. A max {m,n} algorithm for determining the graph H from its line graph G. *Information Processing Letters*, 2(4):108–112, oct 1973. URL: https://www.sciencedirect.com/science/article/pii/002001907390029X, doi:10.1016/0020-0190(73)90029-X.

[16] H. Whitney. Congruent Graphs and the Connectivity of Graphs. *American Journal of Mathematics*, 54(1):150, jan 1932. URL: https://www.jstor.org/stable/2371086?origin=crossref, doi:10.2307/2371086.

[17] M. Yannakakis and Mihalis. Node-and edge-deletion NP-complete problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 253–264, New York, New York, USA, 1978. ACM Press. URL: http://portal.acm.org/citation.cfm?doid=800133.804355, doi:10.1145/800133.804355.