Polynomial Algorithms for Open Plane Graph and Subgraph Isomorphisms[☆]

Colin de la Higuera[a], Jean-Christophe Janodet[b,*], Émilie Samuel[c], Guillaume Damiand[d], Christine Solnon[d]

[a] *Université de Nantes, CNRS, UMR 6241, LINA, F-44000, France*
[b] *Université d'Évry UniverSud Paris, EA 4526, IBISC, F-91037, France*
[c] *Université de Lyon, CNRS, UMR 5516, Laboratoire Hubert Curien, F-42023, France*
[d] *Université de Lyon, CNRS, UMR 5205, LIRIS, F-69622, France*

Abstract

Graphs are used as models in a variety of situations. In some cases, *e.g.* to model images or maps, the graphs will be drawn in the plane, and this feature can be used to obtain new algorithmic results. In this work, we introduce a special class of graphs, called *open plane graphs*, which can be used to represent images or maps for robots: they are planar graphs embedded in the plane, in which certain faces can be removed, are absent or unreachable. We give a normal form for such graphs and prove that one can check in polynomial time if two normalised graphs are isomorphic, or if two open plane graphs are equivalent (their normal forms are isomorphic). Then we consider a new kind of subgraphs, built from subsets of faces and called patterns. We show that searching for a pattern in an open plane graph is tractable if and only if the faces are contiguous, that is, we prove that the problem is $\mathcal{NP}$-complete otherwise.

Keywords: Open plane graphs, equivalence and isomorphism, subgraphs and patterns, polynomial and $\mathcal{NP}$-complete problems.

1. Introduction

Graphs are widely used to represent data and knowledge in a variety of application domains, including electrical, logistical and civil engineering, computer and social networks, linguistics, *etc.* The key point is that graphs are

[☆]The authors acknowledge the support of an ANR grant BLANC 07-1_184534: this work was done in the context of project SATTIC. This work was partially supported by the IST Programme of the European Community, under the PASCAL 2 Network of Excellence, IST–2006-216886.

[*]Corresponding author

Email addresses: `cdlh@univ-nantes.fr` (Colin de la Higuera), `Jean-Christophe.Janodet@ibisc.univ-evry.fr` (Jean-Christophe Janodet), `emilie.samuel@univ-st-etienne.fr` (Émilie Samuel), `guillaume.damiand@liris.cnrs.fr` (Guillaume Damiand), `christine.solnon@liris.cnrs.fr` (Christine Solnon) rich enough to describe the data in a relevant way (*e.g.* by including notions such as adjacency or topology), while allowing —under some assumptions— an efficient exploitation. Thereby, graphs databases dedicated to the development of algorithms have been created, either in an artificial way [1], or from real data such as handwritten characters, fingerprints or web pages [2].

The attention has also turned to modelling images by the mean of graphs for pattern recognition tasks [3, 4]. In consequence, many graph similarity measures have been investigated [5]. These measures often rely on (sub)graph isomorphism —which checks for equivalence or inclusion— or graph edit distances and alignments —which evaluate the cost of transforming a graph into another graph. If there exist rather efficient heuristics for solving the graph isomorphism problem[1] [6, 7, 8], this is not the case for the other measures which are often computationally intractable ($\mathcal{NP}$-hard), and therefore practically unsolvable for large scale graphs. In particular, the best performing approaches for subgraph isomorphism are limited to graphs with up to a few thousands of nodes [9, 7].

However, when measuring graph similarity, it is overwhelmingly forgotten that graphs actually model images and, therefore, have special features that could be exploited to obtain both more relevant measures and more efficient algorithms. Indeed, these graphs are planar, *i.e.*, they may be drawn in the plane, but even more specifically just one of the possible planar embeddings is relevant as it actually models the topology, that is, the order in which faces are encountered when turning around a node.

In the case where just one planar embedding is considered, graphs are called *plane graphs* [10]. It is known since the mid-70's that the isomorphism problem is solvable in polynomial time for plane graphs [11, 12]. However, it has also been noted that the subisomorphism problem is still $\mathcal{NP}$-complete, in particular because the Hamiltonian cycle problem is $\mathcal{NP}$-complete for planar graphs. This result has been refined using parameterized complexity: it can be shown that the subisomorphism problem is linear with respect to the mother graph [13, 14], the problem thus being tractable in practice for small subgraphs. Also many subclasses of planar graphs have been considered with respect to the subisomorphism problem, *e.g.*, the *trees*, or the *outerplanar* graphs [15]. As far as we know, the isomorphism problem is generally solvable in polynomial time for all such graphs, but the subisomorphism is $\mathcal{NP}$-complete.

In [16], we have adopted another strategy. In the framework of plane graphs, we have considered a restricted class of subgraphs built from sets of contiguous faces, and called *compact plane submaps*: such submaps were considered more meaningful than general submaps. In this case, we have shown that the isomorphism and subisomorphism problems were solvable in polynomial time. Besides, we have shown in [17] that these results could be extended to higher-dimension maps, yielding the possibility to work with shapes in 3D-spaces for

[1]The theoretical complexity of graph isomorphism is an open question: If it clearly belongs to $\mathcal{NP}$, it has not been proved to be $\mathcal{NP}$-complete.

instance. Note that similar approaches have been followed by Jiang and Bunke in [18, 19]. These authors introduce so-called ordered graphs to deal with graphs extracted from images: the edges are ordered at each vertex, and this construction allows to consider objects similar to plane maps. They have shown that the subgraph isomorphism was also efficiently solvable for constrained types of subgraphs (called *marked* subgraphs); nevertheless, these subgraphs have no relationships with ours (since they are not based on the faces).

In this article, we extend the results presented in [16] in order to be able to represent situations where part of what is being modelled is unknown, blurred or unimportant. We introduce *open plane graphs*, which correspond to plane graphs whose faces can be either visible or invisible. This, for instance, may enable us to model and search for a mug with a handle, independently of the background (see Figure 1). More precisely, the background of the mug, that is visible through the handle, must not be integrated to the modelling graph, since otherwise, the mug is dependent of the background and cannot be searched in another scene. Using open plane graphs, we simply need to declare invisible all the faces that correspond to the background, and the mug can be retrieved efficiently in other images.

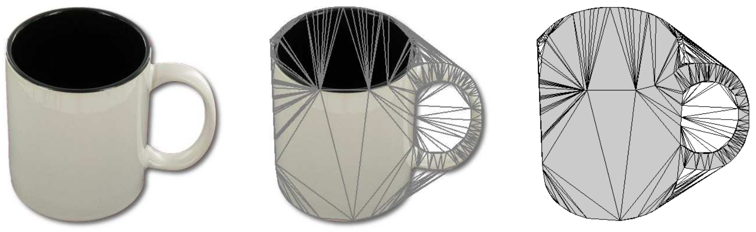


Figure 1: Modelling an image by an open plane graph (on the right). Interest points have been extracted, and the graph is the result of the Delaunay triangulation. The visible faces are in grey, whereas the invisible ones are in white.

Another situation for which open plane graphs are adapted is the model of an environment in which a robot would evolve. *E.g.*, think to famous “intelligent” vacuum-cleaner of Russell and Norvig’s book [20]. The robot needs to have in mind a representation of its world, which contains informations such as the rooms shapes and their topology, as well as the opportunity to go directly from one room to an adjacent one or not, using doors, and avoiding forbidden rooms. If intelligent enough, the robot may have to learn this model by itself. In this case, the faces of the plane graph represent the rooms, the vertices being the corners. Two rooms are connected by a door if they share a common edge in the graph, and the forbidden rooms are tagged invisible (see Figure 2).

Hence, in this paper, we introduce open plane graphs and study the isomophism and subisomorphism problems for this class of graphs. In particular, we prove that depending on the assumptions, some subgraphs called *patterns* can be efficiently searched, whereas others, called *piecewise patterns*, cannot.

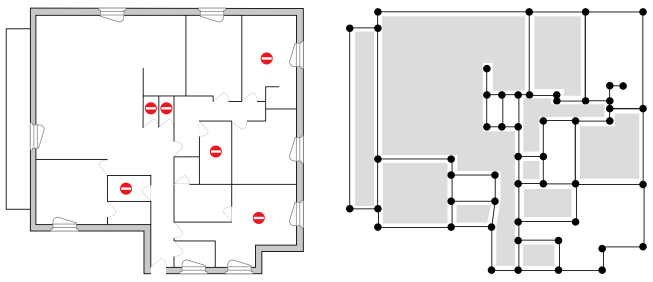


Figure 2: Modelling the environment for a robot. The rooms are on the left (a special symbol indicates which rooms are forbidden), and the corresponding open plane graph is on the right. The visible faces are in grey, whereas the invisible ones are in white.

As the latters are obtained by slightly relaxing the definition of the formers, our result are optimal, that is, inextendable to significantly larger classes of subgraphs built from the faces.

Outline of the Paper. The formal definition of open plane graphs is given in Section 2. After a short presentation of plane graphs, mainly based on the Eric Fusy's thesis [10], we propose a formal system to describe any plane graph through its faces, some of them being visible, and the other invisible. Note that we could have used another formal system, called *partial maps* [17, 21]. Nevertheless, the corner stone of maps is the so-called *darts*, which correspond to directed edges, whereas the corner stone of open plane graphs is the faces directly. So, since our results mainly concern patterns, which are subgraphs induced sets of faces, the latter was more appropriate than the former in this paper, for clarity reasons.

In Section 3, we discuss possible definitions of open plane graph isomorphism: one of them, called *equivalence*, corresponds to isomorphism over the visible faces only. We show that every open plane graph can be reduced to a unique normal form, in a way such that two open plane graphs are equivalent if and only if their normal forms are isomorphic. This reduction is crucial to define our main class of subgraphs, the *patterns*.

The definition of such patterns is given in Section 4, where we also state the problem of searching for patterns in open plane graph. We show that this problem is solvable in polynomial time. As a side effect, we also establish the tractability of equivalence and isomorphism of open plane graphs.

In Section 5, we aim at relaxing the conditions on the patterns that can efficiently be retrieved in an open plane graph. We consider the case of piecewise patterns, that is patterns whose faces are not contiguous. We show that in this case, searching for patterns is $\mathcal{NP}$-complete.

We conclude and give further research directions in Section 6.

2. Open Plane Graphs

In this section, we recall standard notions about plane graphs, that is, planar graphs embedded in the plane. In order to have operational definitions, *i.e.* that can be algorithmically manipulated and allow to study data structures and computational issues, we introduce a new formalism, called *plane graph systems*, which enables us to fully describe any connected plane graph through its *faces*. An *open plane graph* will then be defined as a plane graph system in which only some faces are *visible*.

2.1. Plane Graphs

A detailed introduction to plane graphs can be found in Eric Fusy's Thesis [10]. We mainly use his notations below. Note that planar graph drawing is a research topic *per se* (see *e.g.* [22]).

Let $G = \langle X, E \rangle$ be a *simple graph*, with X the set of *vertices* and $E \subseteq \{e = \{x, y\} : x, y \in X\}$ the set of undirected *edges*. An *embedding* (or drawing) η of graph G in the plane is defined by

1. an injective mapping η_X from the vertices of G to points in the plane, and
2. a mapping η_E from the edges of G to smooth curves in the plane such that the extremities of any edge e are mapped by η_X to the extremities of curve $\eta_E(e)$.

An embedding is *planar* if for all distinct edges $e, e' \in E$, curves $\eta_E(e)$ and $\eta_E(e')$ do not meet except at common extremities. A graph that admits a planar embedding is called a *planar graph.*

In fact, a planar graph admits infinitely many planar embeddings: one only has to slightly move a vertex to get a new planar embedding. However, a graph has only finitely many planar embeddings if the embeddings are considered up to isotopy, that is, two planar embeddings η_1 and η_2 are *isotopic* if η_2 can be obtained from η_1 by moving the vertices in such a way that no edge is crossed. This property is illustrated in Figures 3 and 4.

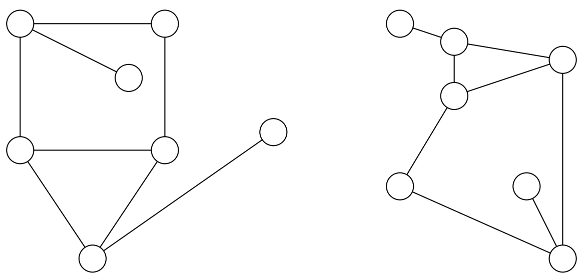


Figure 3: Two isotopic planar embeddings of a planar graph.

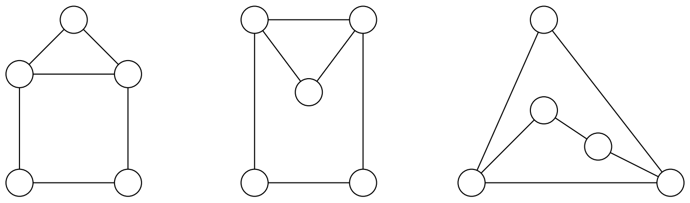


Figure 4: A planar graph with 3 non-isotopic planar embeddings.

Thus, in the following, a *plane graph* will stand for an isotopy class of planar embeddings for a given planar graph. In other words, a plane graph is a planar graph that is embedded in the plane without edge-crossing and up to continuous deformations. A theorem by Fáry states that given any planar embedding of a planar graph, it is always possible to move the vertices, within the same isotopy class, so that the edges are drawn with straight-line segments [23]. We shall use such straight-line drawings in the following.

Any plane graph has *faces*, each face corresponding to a piece of the plane split by the embedding. We denote by F the set of faces. One face, denoted by $o \in F$, is unbounded and called the *outer* or *external* face. The number of faces of a plane graph is an invariant of the underlying planar graph. Indeed, whichever the embedding chosen for planar graph G, one has:

$$|X| - |E| + |F| = 1 + c$$

by Euler's formula, where c denotes the number of connected components.

What is not invariant, however, is the *co-degree* of each face in two non-isotopic planar embeddings (see Figure 5). By the co-degree of some face, we mean the number of edges one meets when walking along the border of the face, keeping the border on the right-hand side. Pendant edges are met twice, thus count twice.

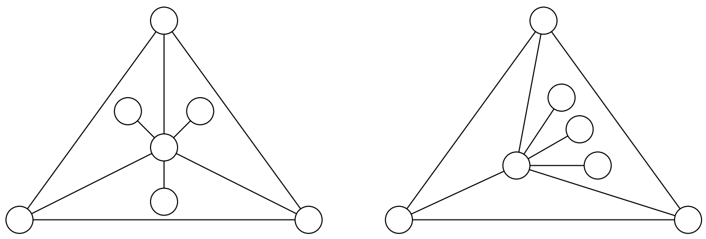


Figure 5: Two non-isotopic embeddings of the same planar graph with 4 faces. On the left, faces have co-degrees 5, 5, 5 and 3 respectively. On the right, faces have co-degrees 9, 3, 3 and 3 respectively.

2.2. Connectivity Issues

In the following, we shall only consider connected plane graphs.

Definition 1 (Connected graph). Any graph $G = \langle X, E \rangle$ is *connected* if for all vertices $x, x' \in X$, there is a sequence $x = x_0, x_1, \dots x_n = x'$ such that for all $i \in \{0, 1, \dots n-1\}$, $\{x_i, x_{i+1}\} \in E$.

A stronger notion of connectivity will also be needed, based on the faces. Indeed, consider again the example of the robot that visits the rooms of a flat, some of them being closed or forbidden. If we wish the robot to map legal rooms, moving from faces to faces and traversing the edges (seen as doors), then the legal faces must also be "connected" through the edges of the graph:

Definition 2 (Set of contiguous faces). Consider a plane graph $G = \langle X, E \rangle$ having F as set of faces.

- Two faces $f, f' \in F$ are *adjacent* is they share a common edge $e \in E$.
- We say that the faces of a subset $K \subseteq F$ are *contiguous* if for all faces $f, f' \in K$, there is a sequence $f = f_0, f_1, \dots f_n = f'$ of faces in K such that for all $i \in \{0, 1, \dots n-1\}$, f_i and f_{i+1} are adjacent.

For instance, consider the plane graph of Figure 6; $K = \{f_1, f_4, f_5\}$ is a set of contiguous faces, whereas $K' = \{f_1, f_2, f_6\}$ is not (since face f_6 is adjacent neither to f_1 nor to f_2.

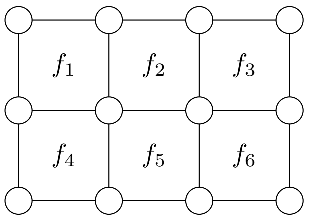


Figure 6: $K = \{f_1, f_4, f_5\}$ is a set of contiguous faces, whereas $K' = \{f_1, f_2, f_6\}$ is not.

2.3. Plane Graph Systems

We have defined the plane graphs using the notion of embeddings, *i.e.*, functions that map vertices to points, and edges to curves. However, this mathematical approach is quite unsuitable for designing algorithms. The set of faces being the corner stone for describing plane graphs, we introduce *plane graph systems*, allowing us to syntactically define a whole isotopy class of planar embeddings.

We first need some notations. Let X be a finite nonempty set of symbols. A *string* over X is a concatenation of symbols. The set of all strings is denoted by X^* and the empty string is denoted by ϵ. A *circular string* is intuitively a string in which the last symbol is followed by the first; that is, there is no first symbol but just a function associating to each symbol the next one. We denote a circular string by $[u]$, with the convention that if u and v are two strings, then $[uv] = [vu]$. The set of all circular strings over X is denoted by $X^\top$. For sake of clarity, we shall use $a, b, \ldots$ as names for symbols in X, and $u, v, w, \ldots$ as names for strings in X^*, and Greek letters $\alpha, \beta, \ldots$ as names for circular strings in $X^\top$.

Now consider the plane graph of Figure 7. The outer face is f_1 and bounded (internal) faces are f_2 and f_3. Each face has only one boundary since the graph is connected. Such a boundary can be described by a circular string of vertices in which two consecutive vertices, and the last and the first are linked by an edge. Conventionally, we follow a boundary by leaving it to the right. In other words, the bounded face is on the left of the walk. Therefore, the boundary of face f_3 is $[ecfcd]$, or equivalently $[fcdec]$, by circular permutation.

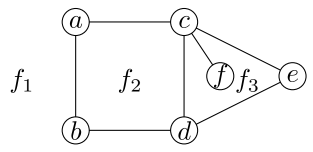


Figure 7: A plane graph with 3 faces.

Definition 3 (Plane Graph System). A *plane graph system* is a tuple $S = \langle X, E, F, o, \mathcal{D} \rangle$ such that:

- $\langle X, E \rangle$ is a planar graph,
- F is a finite nonempty set of symbols called *faces*,
- $o \in F$ is a distinguished symbol called the *outer face* and
- $\mathcal{D} : F \to X^\top$ is a function, called the *boundary descriptor*, that maps any face to its boundary.

For the sake of simplicity, we shall denote by $\mathbf{f}$ the value of $\mathcal{D}(f)$. With such a convention, we shall keep implicit the boundary descriptor $\mathcal{D}$, thus denote by $\langle X, E, F, o \rangle$ the plane graph system S.

For instance, consider the plane graph of Figure 7 again. The corresponding plane graph system is $S = \langle X, E, F, o \rangle$ with $X = \{a, b, c, d, e, f\}$, $E = \{\{a, b\}, \{a, c\}, \{b, d\}, \{c, d\}, \{c, e\}, \{c, f\}, \{d, e\}\}$, $F = \{f_1, f_2, f_3\}$, $o = f_1$ and $\mathbf{f_1} = [acedb]$, $\mathbf{f_2} = [abdc]$, $\mathbf{f_3} = [cdecf]$.

Note that there exist few redundancies in the definition of a plane graph system: both the vertices and the edges could be recovered from the boundaries of the faces. Nevertheless, our choices will ease several proofs and algorithms.

2.4. Open Plane Graphs

If we consider plane graphs extracted from images or modelling maps, specific parts of the image may be hidden or part of the background, and specific rooms may be closed or forbidden. In order to take into account these notions we introduce plane graphs in which some faces may be visible and others not.

Definition 4 (Open Plane Graph). An *open plane graph* is a tuple $G = \langle X, E, F, V, o\rangle$ such that:

1. $\langle X, E, F, o\rangle$ is a plane graph system (see Definition 3), and
2. $V \subseteq F$ is a set of contiguous faces, called the *visible* faces.

Every face in $(F \setminus V)$ will be said *invisible.* These are also called the *holes.*

An open plane graph may have several "holes", and the outer face can be visible or not. See Figure 8 for an example. Note that the invisible faces will generally not be contiguous.

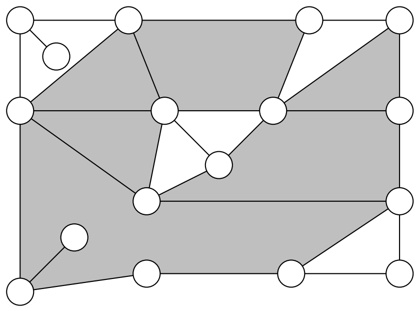


Figure 8: An open plane graph with 12 faces, 6 of them being visible (in grey). The outer face is invisible.

2.5. Further Notations

Let $G = \langle X, E, F, V, o\rangle$ be an open plane graph. Given a vertex $x \in X$, we define the *neighbourhood* $\Gamma(x)$ as the circular string of vertices adjacent to x as read in a counter clockwise order. Length $|\Gamma(x)|$ is known as the *degree* of x and denoted by $\deg(x)$. For instance, in Figure 7, we have $\Gamma(d) = [cbe]$ and $\deg(d) = 3$.

Now with respect to the edges, we denote by faces(e) the set of faces incident to edge e. Set faces(e) can contain either 1 or 2 faces. In order to distinguish them, we shall use $\underrightarrow{xy}$ to denote the face f that stands on the left of edge $\{x, y\}$ when "walking" from vertex x to vertex y. From a mathematical standpoint, f is the face such that $\mathbf{f} = [xyu]$ for some string $u \in X^*$. In consequence, we have that faces$(\{x, y\}) = \{\underrightarrow{xy}, \underrightarrow{yx}\}$. For instance, in Figure 7, we have $\underrightarrow{dc} = f_2$ and $\underrightarrow{cd} = f_3$, thus faces$(\{c, d\}) = \{f_2, f_3\}$. We also have faces$(\{c, f\}) = \{f_3\}$.

3. Open Plane Graphs Isomorphisms and Equivalence

In pattern recognition tasks where graphs are extracted from images, the goal is often to compare the extracted graphs in order to develop similarity measures between the images [5]. In this paper, we aim at comparing the planar embeddings of the graphs, noticing that two isotopic planar embeddings are more likely to represent similar images than non-isotopic embeddings.

It is important to make a distinction between isomorphisms, which concern graphs as mathematical objects, and isotopies, which concern embeddings (drawings) of these graphs. For instance, the notion of isotopy makes sense as soon as the supporting surface for the drawings of the graphs is fixed; this surface is not necessarily a plane, but could be a sphere or a torus. Conversely, comparing embeddings situated on different surfaces is meaningless. Now concerning the notion of isomorphism, it does not take the embeddings into account. Thus, two graphs may be isomorphic while their embeddings are non-isotopic or simply not drawn on the same sort of surface, thus incomparable in terms of embeddings.

3.1. Which Notion of Isomorphism?

Two graphs $G_1 = \langle X_1, E_1 \rangle$ and $G_2 = \langle X_2, E_2 \rangle$ are usually said to be *isomorphic* if there exists a one-to-one mapping $\phi : X_1 \to X_2$ that preserves the edges: $\{x, y\} \in E_1 \Leftrightarrow \{\phi(x), \phi(y)\} \in E_2$. However, this traditional notion of isomorphism does not fit our picture, as the only thing that is checked is whether the edges are preserved. Indeed, consider both graphs of Figure 9 for instance. They are isomorphic (as shown by the labels of the vertices), but the faces are different (since one of them is bounded by 4 edges in the right-hand graph, while no such face exists in the left-hand graph).

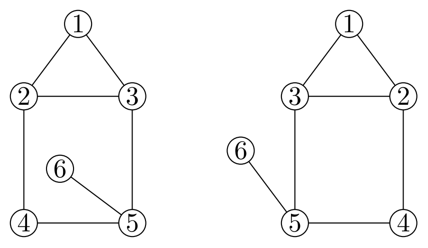


Figure 9: Two standard-isomorphic graphs.

In the case of plane graphs, preserving the faces is much more important, and this is the reason why we add the new constraints hereafter:

Definition 5 (Sphere-Isomorphism). Let $G_1 = \langle X_1, E_1, F_1, V_1, o_1 \rangle$ and $G_2 = \langle X_2, E_2, F_2, V_2, o_2 \rangle$ be two open plane graphs. We say that G_1 and G_2 are *sphere-isomorphic*, denoted $G_1 \equiv_s G_2$, if

1. there exists a one-to-one mapping $\chi : X_1 \to X_2$ over the vertices;
2. there exists a one-to-one mapping $\xi : F_1 \to F_2$ over the faces whose boundaries are preserved: $\forall f_1 \in F_1, \forall f_2 \in F_2$,

$$\text{if } \xi(f_1) = f_2 \text{ and } \mathbf{f_1} = [x_1 x_2 \dots x_p], \text{ then } \mathbf{f_2} = [\chi(x_1)\chi(x_2)\dots\chi(x_p)];$$

3. the visible faces are mapped together:

$$\xi(V_1) = V_2 \text{ (and thus } \xi(F_1 \setminus V_1) = F_2 \setminus V_2).$$

Figure 10 shows three open plane graphs that are pairwise sphere-isomorphic (as shown by the labels of the vertices; the visible faces are in grey). As we did not specify any constraints concerning the preservation of the outer faces, the outer face of a plane graph can be mapped to an internal face of another graph. In other words, all the faces play the same role, as if we had drawn them on a sphere rather than on the plane. This is what was done in Figure 11; one just has to roll the sphere and then project the drawing on the plane to get one of the plane graphs of Figure 10. This phenomenon justifies why these plane graphs are said *sphere*-isomorphic.

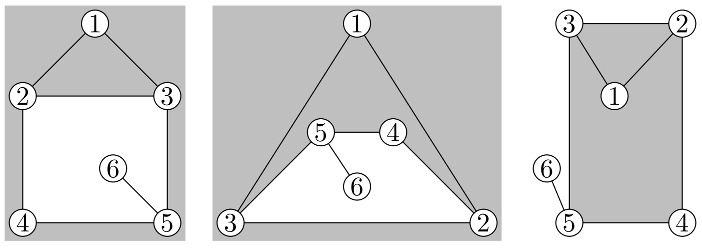


Figure 10: Three sphere-isomorphic open plane graphs.

Although sphere-isomorphism is not exactly the notion which we would like to test over the open plane graphs, we will need this notion to establish theoretical results. Moreover, it is interesting to note that the isomorphism notion proposed by Cori in [11, 12] and Lienhardt in [24] for combinatorial maps actually corresponds to sphere-isomorphism for open plane graphs.

Finally, the most relevant notion of isomorphism that we aim at testing is the one that preserves all the faces and also the outer face:

Definition 6 (Plane-Isomorphism). Let $G_1 = \langle X_1, E_1, F_1, V_1, o_1 \rangle$ and $G_2 = \langle X_2, E_2, F_2, V_2, o_2 \rangle$ be two open plane graphs. We say that G_1 and G_2 are *plane-isomorphic*, denoted $G_1 \equiv_p G_2$, if

1. G_1 and G_2 are sphere-isomorphic with respect to mappings $\chi : X_1 \to X_2$ and $\xi : F_1 \to F_2$;
2. the outer faces are mapped together: $\xi(o_1) = o_2$.

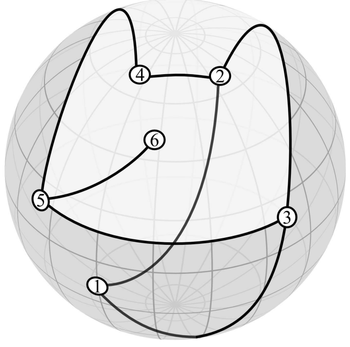


Figure 11: A "sphere-graph" with three faces. Declaring any face as an outer face yields one plane graph among those of Figure 10.

An example is given in Figure 12. Clearly, plane-isomorphism preserves the vertices, the edges, the faces, the visible faces and the outer faces of open plane graphs. The job seems done ... however, plane-isomorphism is still not satisfying, as we are going to explain in next paragraph.

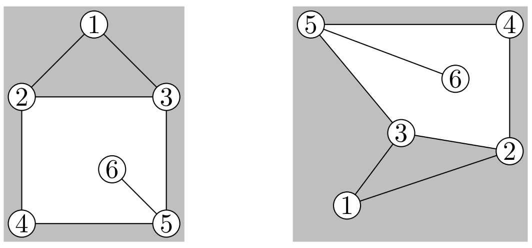


Figure 12: Two plane-isomorphic open plane graphs.

3.2. An Equivalence rather than an Isomorphism?

In a pattern recognition task, the visible faces of an open plane graph will correspond to a pattern that was searched and found. In consequence, the visible faces and their relative positions must be taken into account, but the invisible faces must not play any role. In other words, we actually need to compare the open plane graphs over their visible faces only.

To clarify this point, consider the case of a robot that would explore an open plane graph, going from visible faces to visible faces. Two adjacent visible faces would be materialised by a line on the floor, whereas if either of the faces is invisible, a wall would separate them. Posts (or poles) would be placed to join the walls where the vertices are. Then the robot would be able to map out the visible faces, while it would have no idea about the invisible area. Thus a robot that would visit the visible faces of two open plane graphs could decide that these graphs are *equivalent* up to visible faces.

We formalise open plane graph equivalence as follows:

Definition 7 (Open Plane Graph Equivalence). Two open plane graphs $G_1 = \langle X_1, E_1, F_1, V_1, o_1 \rangle$ and $G_2 = \langle X_2, E_2, F_2, V_2, o_2 \rangle$ are *equivalent*, denoted $G_1 \cong G_2$, if there exists a pair $\langle \phi, (\psi_f)_{f \in V_1} \rangle$ such that:

1. $\phi : V_1 \to V_2$ is a one-to-one mapping over the *visible* faces;
2. $(\psi_f)_{f \in V_1}$ is an indexed family of mappings such that if $\phi(f_1) = f_2$ for some $f_1 \in V_1, f_2 \in V_2$, then:
   (a) ψ_{f_1} : vertices$(f_1) \to$ vertices(f_2) is a one-to-one mapping from the vertices of $\mathbf{f_1}$ to the vertices of $\mathbf{f_2}$, and
   (b) if $\mathbf{f_1} = [x_1 x_2 \dots x_p]$ then $\mathbf{f_2} = [\psi_{f_1}(x_1)\psi_{f_1}(x_2) \dots \psi_{f_1}(x_p)]$.
3. If two visible faces of G_1 share a common edge their images also share a corresponding common edge in G_2. That is, if $\{x, y\} \in E_1$ is an edge and faces$(\{x, y\}) = \{f_1, f_2\}$, with $f_1 \in V_1$, then
   - either $f_2 \in V_1$ and $\psi_{f_1}(x) = \psi_{f_2}(x)$ and $\psi_{f_1}(y) = \psi_{f_2}(y)$, thus faces$(\{\psi_{f_1}(x), \psi_{f_1}(y)\}) =$ faces$(\{\psi_{f_2}(x), \psi_{f_2}(y)\}) = \{\phi(f_1), \phi(f_2)\}$,
   - or $f_2 \notin V_1$ and faces$(\{\psi_{f_1}(x), \psi_{f_1}(y)\}) = \{\phi(f_1), g\}$ with $g \notin V_2$.
4. The outer faces are preserved:
   - if $o_1 \in V_1$, then $o_2 \in V_2$ and $\phi(o_1) = o_2$;
   - if $o_1 \notin V_1$, then $o_2 \notin V_2$.

By Definition 7, two open plane graphs G and G' are equivalent if a robot that would visit the visible faces of both graphs would not be able to distinguish one from the other. Each ψ_f gives the vision the robot would have when looking from face f. See Figures 13 and 15 for two equivalent open plane graphs and Figure 14 for two non equivalent open plane graphs.

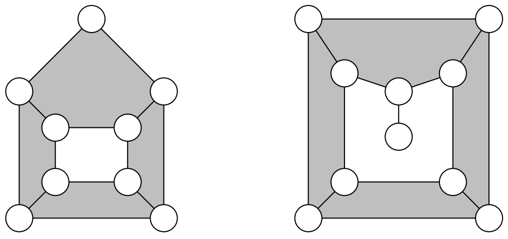


Figure 13: Two equivalent open plane graphs.

We finally have the following property:

Theorem 1. *Relation $\cong$ is an equivalence relation over the open plane graphs.*

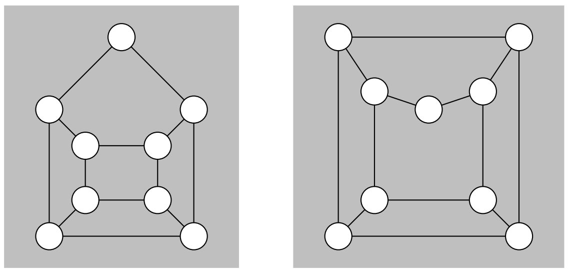


Figure 14: Two non equivalent open plane graphs.

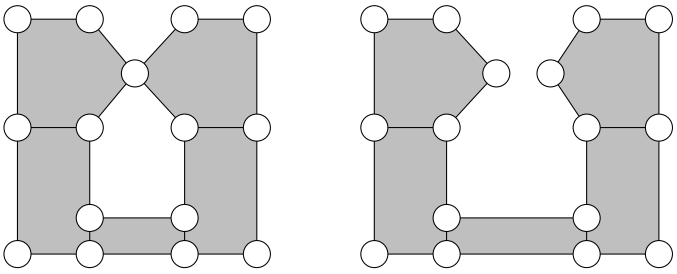


Figure 15: Two equivalent open plane graphs.

Proof. Reflexivity and transitivity are straightforward. Concerning symmetry, assume that graph $G = \langle X, E, F, V, o \rangle$ is mapped to graph $G' = \langle X', E', F', V', o' \rangle$ using pair $\langle \phi, (\psi_f)_{f \in V} \rangle$. We define a pair $\langle \phi', (\psi_g)_{g \in V'} \rangle$ in order to prove that $G' \cong G$. As $\phi : V \to V'$ is a one-to-one mapping over the visible faces, we fix $\phi' = \phi^{-1}$. We now consider any face $g \in V'$ and define ψ_g. Face g has a unique predecessor f by ϕ, and since ψ_f is a one-to-one mapping from the vertices of $\mathbf{f}$ to the vertices of $\mathbf{g}$, we fix $\psi'_g = \psi_f^{-1}$. Clearly, ψ'_g is a one-to-one mapping from the vertices of $\mathbf{g}$ to the vertices of $\mathbf{f}$ that preserves the boundaries of g. Now consider an edge $e' = \{x, y\} \in E'$ and let $\{g, g'\} = \text{faces}(e')$ with $g \in V'$. Then $\{\psi'_g(x), \psi'_g(y)\}$ denotes some edge $e \in E$ (since g is a visible face whose boundary is preserved by ψ'_g). Moreover, we have $\text{faces}(e) = \{\phi'(g), f'\}$, and $g' \in V'$ if and only if $f' \in V$. Finally, if $f' \in V$ then $\psi_f(\psi'_g(x)) = \psi_{f'}(\psi'_g(x))$, that is, $x = \psi_{f'}(\psi'_g(x))$, thus $\psi'_{g'}(x) = \psi'_g(x)$. And for the same reason, $\psi'_{g'}(y) = \psi'_g(y)$. □

3.3. On Bridges, Branches, Hinges. . .

In the paragraphs above, we have introduced several definitions of isomorphisms and a further notion of equivalence, which is a form of isomorphism over the visible faces only. In fact, we are going to show that these notions coincide for plane graphs whose invisible area are “empty”, that is, do not contain any edge nor vertex. The elimination of such objects will be achieved thanks to a procedure that we call an open plane graph *normalisation.* In this section, we identify all the situations the normalisation procedure will be faced with.

On the one hand, the invisible area of an open plane graph may contain edges that should disappear. This elimination nevertheless depends on the nature of these edges. An example is given in Figure 16. Some of the edges may be *pendant*:

Definition 8 (Pendant Edge). An edge $e \in E$ is *pendant* in an open plane graph $G = \langle X, E, F, V, o\rangle$ if some of its endpoint has degree one.

The second type of edges to be considered are those that separates two invisible faces. We call them *bridges* and define them as follows:

Definition 9 (Bridge). A *bridge* in an open plane graph $G = \langle X, E, F, V, o\rangle$ is an edge e such that (1) $|\text{faces}(e)| = 2$ and (2) $\text{faces}(e) \cap V = \emptyset$.

The third and last type of edges are those that are adjacent to only one invisible face but are not pendant. We call them the *branches*:

Definition 10 (Branch). A *branch* in an open plane graph $G = \langle X, E, F, V, o\rangle$ is an edge $e = \{x, y\}$ such that (1) $|\text{faces}(e)| = 1$ and (2) $\text{faces}(e) \cap V = \emptyset$ and (3) $\deg(x) > 1$ and (4) $\deg(y) > 1$.

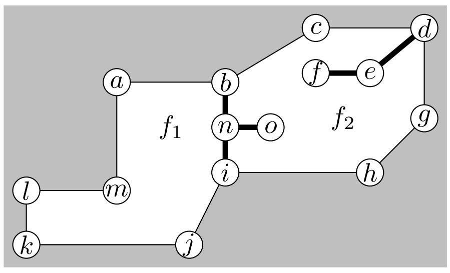


Figure 16: An open plane graph with 2 pendant edges $\{e, f\}$ and $\{n, o\}$, 2 bridges $\{b, n\}$ and $\{n, i\}$ and 1 branch $\{d, e\}$.

On the other hand, the invisible area may also contain vertices that require specific treatment. Indeed, let us consider the case of vertex x in Figure 17 and suppose that a robot is exploring the graph, going from visible faces to visible faces. The robot will meet vertex x at least 3 times when visiting the visible faces. However, it will be impossible for him to detect that all these occurrences correspond to the same vertex, since in order to decide this, he would need to turn around x and thus traverse forbidden invisible faces. Such a vertex is called a *hinge.*

Definition 11 (Hinge). A vertex x in an open plane graph $G = \langle X, E, F, V, o\rangle$ is a *hinge* if $\Gamma(x) = [y_1 y_2 ... y_n]$, and there exist $1 \leq i < j < k < l \leq n$ such that $\underrightarrow{xy_i} \in V$, $\underrightarrow{xy_j} \notin V$, $\underrightarrow{xy_k} \in V$, $\underrightarrow{xy_l} \notin V$.

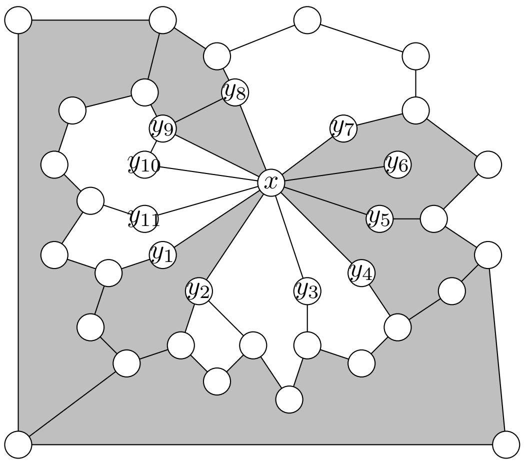


Figure 17: Vertex x is a hinge.

Remember that $\Gamma(x)$ denotes the neighbourhood of vertex x, that is, the circular string of vertices adjacent to x as read in a counter clockwise order. Moreover, $\underrightarrow{xy}$ denotes the face f such that $\mathbf{f} = [xyu]$ for some string $u \in X^*$. Thus, in Figure 17, we have $\Gamma(x) = [y_1 y_2 \dots y_{11}]$, $\underrightarrow{xy_1} \in V$, $\underrightarrow{xy_2} \notin V$, $\underrightarrow{xy_4} \in V$ and $\underrightarrow{xy_7} \notin V$, that is, vertex x is a hinge.

Hence, the normalisation procedure that we describe in next paragraph will have to eliminate pendant edges, bridges, branches and hinges. We shall prove (in Theorem 3) that two open plane graphs are equivalent if their normal forms are isomorphic.

3.4. Normalising an Open Plane Graph

First of all, we define:

Definition 12 (Irreducible graph). An open plane graph G is *irreducible* if G has neither hinge, nor bridge, nor branch, nor pendant edge in any invisible face.

The aim of this section is to develop the algorithm which, given any open plane graph G, returns a *unique equivalent irreducible* open plane graph denoted $\widehat{G}$. The normalisation procedure is described in Algorithm 18 and consists in successively eliminating the hinges, then the bridges, then the pendant edges. The branches do not need any specific treatment. Let us explicit all subroutines before proving the properties of the entire normalisation process.

3.4.1. Eliminating the Hinges

Procedure eliminateHinge is given in Algorithm 19. The key idea of this algorithm is to replace the hinge by a new invisible face. This operation introduces many new bridges but no new hinge. New vertices are also created. An example

```
    Input: An open plane graph $G = \langle X, E, F, V, o\rangle$
    Output: Irreducible open plane graph $\widehat{G}$
1   while there is some hinge in $G$ do
2       let $x$ be such an hinge;
3       eliminateHinge($G, x$)
4   while there is some bridge in $G$ do
5       let $e = \{x, y\}$ be such a bridge;
6       eliminateBridge($G, e$)
7   while there is some pendant edge in an invisible face of $G$ do
8       let $e = \{x, y\}$ be such a pendant edge;
9       eliminatePendantEdge($G, e$)
10  return modified open plane graph $G$
```

Figure 18: normaliseGraph(G)

is given in Figure 20: this is the open plane graph obtained by the elimination of hinge x in Figure 17. Various situations are depicted in this example, including adjacent visible faces, adjacent invisible faces, bridges and pendant edges. Note that $\Gamma(x) = [y_1 y_2 \ldots y_{11}]$ and $\underrightarrow{xy_1} \in V$ and $\underrightarrow{xy_{11}} \in (F \setminus V)$. The substring of $\Gamma(x)$ that is extracted by Algorithm 19 at lines 2-5 is $[y_1 y_2 y_4 y_7 y_8 y_9]$. The hinge is replaced by a new invisible face $\mathbf{h} = [y_1 x_1 y_2 x_2 y_4 x_4 y_7 x_7 y_8 x_8 y_9 x_9]$.

Lemma 1. *Let $G = \langle X, E, F, V, o\rangle$ be an open plane graph having a hinge x. Then graph G' obtained by eliminating x using Algorithm 19 is equivalent to G.*

Proof. Each visible face $f = \underrightarrow{xy_j}$ for some $j \in \{1, 2 \ldots n\}$ is transformed into some face f' where $\mathbf{f}'$ is deduced from $\mathbf{f}$ by replacing every occurrence of x by vertex x_p for some p. So for such faces, we fix $\phi(f) = f'$ and $\psi_f(x) = x_p$ and $\psi_f(y) = y$ for all $y \neq x$. Concerning all other visible faces, we fix $\phi(f) = f$ and $\psi_f(y) = y$ for all vertex y in $\mathbf{f}$. Clearly, Condition 1 of Definition 7 holds because Algorithm 19 does not modify the visibility or invisibility of existing faces. Condition 2 holds because all the x_p are fresh vertices, thus all ψ_f are one-to-one mappings that preserve the boundaries.

Finally, consider an edge e and let $\{f, g\} = \text{faces}(e)$. Condition 3 basically holds, except if $f = \underrightarrow{xy_j}$ and $g = \underrightarrow{xy_{j+1}}$ and $e = \{x, y_{j+1}\}$ for some j. Let us assume *w.l.o.g.* that f is visible, and suppose that vertex x in $\mathbf{f}$ is replaced by some x_p to get $\mathbf{f}'$. In this situation, edge $\{x, y_{j+1}\}$ is transformed into edge $\{x_p, y_{j+1}\}$. There are two cases. If g is also visible, then p is the largest index in $\{i_1, i_2, \ldots i_k\}$ such that $p \leq j+1$, so x will also be replaced by x_p in $\mathbf{g}$. Thus we have $\psi_f(x) = x_p = \psi_g(x)$ and $\psi_f(y_{j+1}) = y_{j+1} = \psi_g(y_{j+1})$. On the other hand, if g is not visible, then $\{x_p, y_{j+1}\}$ is an edge of the new invisible face h that is added to the graph. So $\text{faces}(\{x_p, y_{j+1}\}) = \{\phi(f), h\}$ and h is invisible. □

```
Input: An open plane graph $G = \langle X, E, F, V, o \rangle$, a hinge $x$
Output: Graph $G$ modified
let $[y_1 y_2 \dots y_n] = \Gamma(x)$ such that $\underrightarrow{xy_1} \in V$ and $\underrightarrow{xy_n} \in (F \setminus V)$;
select largest substring $[y_{i_1} y_{i_2} \dots y_{i_k}]$ of $\Gamma(x)$ such that
        $1 = i_1 < i_2 < i_3 < \dots i_k \leq n$ and, $\forall s \in \{1, 2, \dots, k-1\}$
        either $(\underrightarrow{xy_{i_s}} \in V$ and $\underrightarrow{xy_{i_s+1}} \in (F \setminus V))$
        or $(\underrightarrow{xy_{i_s}} \in (F \setminus V)$ and $\underrightarrow{xy_{i_s+1}} \in V)$;
// each edge $\{x, y_{i_s}\}$ separates a visible and an invisible face
$X \leftarrow X \setminus \{x\} \cup \{x_{i_1}, x_{i_2}, \dots x_{i_k}\}$;
foreach face $f = \underrightarrow{xy_j}$ with $j \in 1, 2 \dots n$ do
    let $p$ be the largest index in $\{i_1, i_2, \dots i_k\}$ such that $p \leq j$;
    replace all occurrences of $x$ by $x_p$ in $\mathbf{f}$
$\mathbf{h} \leftarrow [y_{i_1} x_{i_1} y_{i_2} x_{i_2} \dots y_{i_k} x_{i_k}]$;
$F \leftarrow F \cup \{h\}$; // $h$ is an invisible face
update $E$;
return modified open plane graph $G$
```

Figure 19: eliminateHinge(G, x)

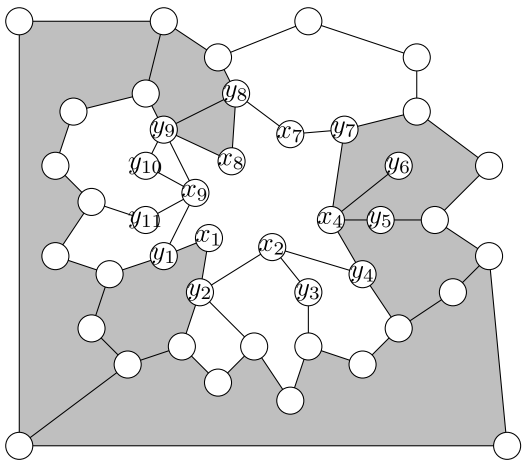


Figure 20: Elimination of the hinge of Figure 17.

3.4.2. Eliminating the Bridges

In Algorithm 21 each individual bridge is removed. In doing so, two invisible faces will be merged into one, with a new boundary. This operation may create branches and pendant edges but no new bridge, nor new hinge. An example is given in Figure 22: the upper bridge in Figure 16 is eliminated, while the lower bridge is transformed into a branch.

```
   **Input**: An open plane graph $G = \langle X, E, F, V, o \rangle$, a bridge $\{x, y\}$
   **Output**: Graph $G$ modified
1  assume that faces($\{x, y\}$) = $\{f, f'\}$ and
2          $\mathbf{f} = [xyu]$ and
3          $\mathbf{f}' = [yxv]$ for some $u, v \in X^*$;
4  $\mathbf{f} \leftarrow [xvyu]$;
5  $F \leftarrow F \setminus \{f'\}$;
6  if outer face $o$ was face $f'$ then $o \leftarrow f$;
7  $E \leftarrow E \setminus \{\{x, y\}\}$;
8  return modified open plane graph $G$
```

Figure 21: eliminateBridge($G, \{x, y\}$)

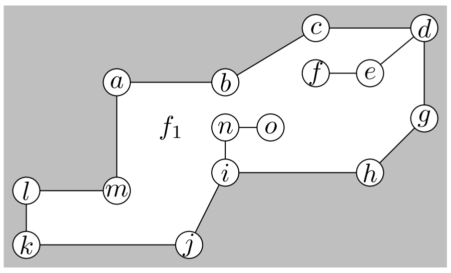


Figure 22: The open plane graph of Figure 16 where bridge $\{b, n\}$ was eliminated.

Lemma 2. *Let $G = \langle X, E, F, V, o \rangle$ be an open plane graph having some bridge $e = \{x, y\}$. Then graph G' obtained by eliminating e using Algorithm 21 is equivalent to G.*

Proof. Algorithm 21 does not modify the boundaries of visible faces. Indeed, no vertex is eliminated. Moreover the bridge is the only edge that disappears, and the bridge is adjacent to invisible faces only (by definition). Therefore, we choose ϕ and all ψ_f to be Identity and get the equivalence of G and G'. □

3.4.3. Eliminating Pendant Edges in Invisible Faces

The last procedure described in Algorithm 23 allows us to remove the pendant edges. Basically, this operation does not reintroduce any new bridge nor hinge. Note nevertheless that it transforms branches into pendant edges. In other words, the number of pendant edges does not necessarily decrease at each call of this procedure, but the total number of edges (and vertices) does. This will ensure the termination of the normalisation process.

```
Input: An open plane graph $G = \langle X, E, F, V, o \rangle$, a pendant edge $\{x, y\}$
       in an invisible face $f$, with $\deg(x) = 1$
Output: Graph $G$ modified
1 assume that $\mathbf{f} = [uyxy]$ with $u \in V^*$;
2 $\mathbf{f} \leftarrow [uy]$;
3 $X \leftarrow X \setminus \{x\}$;
4 $E \leftarrow E \setminus \{\{x, y\}\}$;
5 return modified open plane graph $G$
```

Figure 23: eliminatePendantEdge(G, $\{x, y\}$)

Lemma 3. *Let $G = \langle X, E, F, V, o \rangle$ be an open plane graph and e a pendant edge in an invisible face. Then graph G' obtained by eliminating e using Algorithm 23 is equivalent to G.*

Proof. Straightforward since the algorithm does not modify the visible faces. □

3.5. Conclusion and Main Result

Concerning Algorithm 18, we finally get:

Theorem 2. *Let G be an open plane graph. Procedure normaliseGraph(G) converges in polynomial time towards an irreducible open plane graph that is equivalent to G. This open plane graph is called* the normal form *of graph G, and denoted by $\widehat{G}$.*

Proof. Each step of the normalisation process preserves the equivalence. Moreover, the elimination of the hinges introduces many bridges but no hinge. Then the elimination of the bridges may introduce branches and pendant edges but no hinge nor bridge. Finally the elimination of the pendant edge eliminates the branches, and decreases the number of edges, and does not re-introduce any hinge nor bridge. So normaliseGraph(G) converges towards irreducible open plane graph $\widehat{G}$ in polynomial time. □

Furthermore, we are now able to precisely describe the relationship between equivalence relation and isomorphism relation:

Theorem 3. *Let G_1 and G_2 two open plane graphs.*

- *When the outer faces of G_1 and G_2 are invisible, G_1 and G_2 are equivalent if and only if their normal forms are sphere-isomorphic:*

$$G_1 \cong G_2 \text{ iff } \widehat{G_1} \equiv_s \widehat{G_2}.$$

- *When the outer faces of G_1 and G_2 are visible, G_1 and G_2 are equivalent if and only if their normal forms are plane-isomorphic:*

$$G_1 \cong G_2 \textit{ iff } \widehat{G_1} \equiv_p \widehat{G_2}.$$

- *If the outer face of G_1 is visible and the one of G_2 is not, or vice-versa, then G_1 and G_2 are not equivalent.*

Proof. First of all, note that if $\widehat{G_1} \equiv_s \widehat{G_2}$ (or $\widehat{G_1} \equiv_p \widehat{G_2}$), then $\widehat{G_1} \cong \widehat{G_2}$. Moreover $G_1 \cong \widehat{G_1}$ and $G_2 \cong \widehat{G_2}$, by Theorem 2. So we deduce that $G_1 \cong G_2$, by transitivity (Theorem 1).

Conversely, let us suppose that $G_1 \cong G_2$, and the outer face of G_1 and G_2 are invisible. We need few notations: For each $j \in \{1, 2\}$, we assume that $G_j = \langle X_j, E_j, F_j, V_j, o_j \rangle$ and denote $\widehat{G_j} = \langle \widehat{X_j}, \widehat{E_j}, \widehat{F_j}, \widehat{V_j}, \widehat{o_j} \rangle$ their normal forms. We know that $\widehat{G_1} \cong G_1$, so let $\langle \phi'_1, (\psi'_{1f})_{f \in \widehat{V_1}} \rangle$ be the pair of mappings that satisfy the conditions of Definition 7. As $G_1 \cong G_2$ and $G_2 \cong \widehat{G_2}$, there also exist pairs $\langle \phi, (\psi_g)_{g \in V_1} \rangle$ and $\langle \phi_2, (\psi_{2h})_{h \in V_2} \rangle$. As we aim at showing that $\widehat{G_1} \equiv \widehat{G_2}$, we need to define a pair $\langle \xi, \chi \rangle$ that satisfies the conditions of Definition 5. The construction is illustrated in Figure 24.

$$\begin{tikzcd}
G_1 \arrow[r, "{\langle \phi, (\psi_g)_{g \in V_1} \rangle}"] & G_2 \arrow[d, "{\langle \phi_2, (\psi_{2h})_{h \in V_2} \rangle}"] \\
\widehat{G_1} \arrow[u, "{\langle \phi'_1, (\psi'_{1f})_{f \in \widehat{V_1}} \rangle}"] \arrow[r, dashed, "{\langle \xi, \chi \rangle}"'] & \widehat{G_2}
\end{tikzcd}$$

Figure 24: Notations for the proof of Theorem 3.

Concerning mapping $\chi : \widehat{X_1} \to \widehat{X_2}$, let $x \in \widehat{X_1}$. As $\widehat{G_1}$ is irreducible, x is adjacent to at least one visible face $f \in \widehat{V_1}$. Let $g = \phi'_1(f)$ and $h = \phi(g)$; we fix $\chi(x) = \left(\psi_{2h} \circ \psi_g \circ \psi'_{1f}\right)(x)$. We claim that this definition does not depend on chosen (visible) face f. Indeed, consider the neighbourhood $\Gamma(x) = [y_1 y_2 \dots y_n]$ in $\widehat{G_1}$. As $\widehat{G_1}$ is irreducible, at most one adjacent face to x is invisible, say $\underrightarrow{xy_n}$. Then for all $i \in \{1, 2 \dots n-2\}$, faces $f_i = \underrightarrow{xy_i}$ and $f_{i+1} = \underrightarrow{xy_{i+1}}$ are visible and share edge $\{x, y_{i+1}\}$. So by Condition 3 of Definition 7, we get $\psi'_{1,f_i}(x) = \psi'_{1,f_{i+1}}(x)$. As adjacent visible faces are also preserved by ϕ and ϕ_2, the same argument can be repeated, that yields a unique value for $\chi(x)$. Also note that χ is a one-to-one mapping over the vertices: this property is inherited from mappings $(\psi'_{1f})_{f \in \widehat{V_1}}$, $(\psi_g)_{g \in V_1}$ and $(\psi_{2h})_{h \in V_2}$.

Now let us define mapping $\xi : \widehat{F_1} \to \widehat{F_2}$. For any visible face f, we fix $\xi(f) = (\phi_2 \circ \phi \circ \phi'_1)(f)$. As ϕ'_1, ϕ and ϕ_2 are one-to-one mappings over the visible faces, ξ is a one-to-one mapping from the visible faces of $\widehat{G_1}$ to the visible faces of $\widehat{G_2}$. Let us now define $\xi(f)$ for an invisible face f of $\widehat{G_1}$. Suppose that $\mathbf{f} = [v_0 v_1 \dots v_n]$. We claim that $[\chi(v_0)\chi(v_1)\dots\chi(v_n)]$ is the boundary of a *unique* invisible face of $\widehat{G_2}$ that we denote $\xi(f)$.

Indeed, consider the edges $e_0 = \{v_0, v_1\}$ and $e_1 = \{v_1, v_2\}$. Edge e_0 separates the invisible face f and a visible face g_0, since $\widehat{G_1}$ has no bridge. So by Condition 3 in Definition 7, $\{\chi(v_0), \chi(v_1)\}$ is an edge of $\widehat{G_2}$ that separates an invisible face h_0 and the visible face $\xi(g_0)$. For the same reason, edge e_1 separates the invisible face f and an visible face g_1, and $\{\chi(v_1), \chi(v_2)\}$ separates an invisible face h_1 and the visible face $\xi(g_1)$. Graph $\widehat{G_2}$ is irreducible, so vertex $\chi(v_1)$ is not an hinge. As there is at most one invisible face adjacent to $\chi(v_1)$ in $\widehat{G_2}$, we conclude that $h_0 = h_1$. In consequence, $\chi(v_0), \chi(v_1)$ and then $\chi(v_2), \dots \chi(v_n)$ surround an invisible area which corresponds to some face h in $\widehat{G_2}$. So we fix $\xi(f) = h$.

With this definition, ξ is a one-to-one mapping over the visible and invisible faces, and χ preserves their boundaries by construction. Note that if the outer faces are invisible, then they are not necessarily mapped together, that is, $\widehat{G_1}$ and $\widehat{G_2}$ are sphere-isomorphic. Conversely, if the outer faces are visible, then they are mapped together by Condition 4 of Definition 7, so $\widehat{G_1}$ and $\widehat{G_2}$ are plane-isomorphic. $\square$

4. Searching for Patterns

4.1. Subgraphs and Patterns

In the framework of standard graph theory, we say that graph $G_1 = \langle X_1, E_1 \rangle$ is a subgraph of $G_2 = \langle X_2, E_2 \rangle$ if $X_1 \subseteq X_2$ and $E_1 \subseteq X_2$. That is, G_1 is obtained from G_2 by erasing vertices and edges. Nevertheless, the most interesting component of a plane graph is not its vertices nor its edges, but its faces. Indeed, when used to model images, the faces of the plane graph may represent homogeneous regions computed by segmentation. In this case, searching for a pattern in an image consists in detecting the presence of a subset of the faces in corresponding plane graph.

Now, in terms of open plane graphs, the distinction between visible and invisible faces allows us to construct such "face-based subgraphs" quite easily, by transforming visible faces into invisible ones. The remaining set of visible faces will have to be contiguous. Moreover, the subgraph that we get is generally not irreducible, that is, it may contain bridges, hinges, *etc.* As these junks are meaningless in a pattern recognition task, we eliminate them by normalisation:

Definition 13 (Pattern). Let P be an *irreducible* open plane graph, and $G = \langle X, E, F, V, o \rangle$ an open plane graph. We say that P is a *pattern* of G if there exists an open plane graph $G' = \langle X, E, F, W, o \rangle$ such that:

1. G' has less visible faces than G : $W \subseteq V$;
2. W is a set of contiguous faces;
3. P and the normal form of G' are plane-isomorphic: $P \equiv_p \widehat{G'}$.

An example of pattern is given in Figure 25. Note that in the left-hand plane graph, the boundary that surrounds the feet of the penguin, crosses twice one of the vertices. When the pattern was chosen, neither the face between the legs was selected, nor the ground. So this vertex has become a hinge which was divided into two vertices, by the normalisation procedure, in the right-hand pattern. Also note that a pattern could contain the outer face.

Finally, as a side effect of the normalisation, it is interesting to note that a pattern P can be larger (in the traditional sense: more vertices) than the original graph. This is a reason why the term *pattern* is preferable to the more standard term *subgraph*.

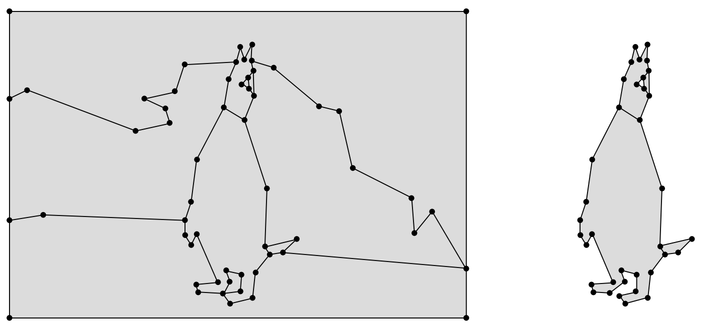


Figure 25: An open plane graph (on the left) and a pattern (on the right).

4.2. Searching for Patterns is Tractable

In many pattern recognition tasks, one aims at deciding in polynomial time whether an open plane graph P is a pattern of an open plane graph G. So we consider the following problem:

Problem: Pattern of an Open Plane Graph (Popg)
Instance: two open plane graphs P and G
Question: is P a pattern of G?

We shall show that:

Theorem 4. *Problem* Popg *is in class* $\mathcal{P}$.

Besides, as the definition of a pattern tests whether two open plane graph are plane-isomorphic or not, we will first need to solve:

Problem: Open Plane Graphs Plane-Isomorphism (Opgpi)
Instance: two open plane graphs G and G'
Question: are G and G' plane-isomorphic?

We shall prove that:

Theorem 5. *Problem* Opgpi *is in class $\mathcal{P}$.*
More precisely, it is decidable in $\mathcal{O}(|E| \cdot |E'|)$ time whether open plane graphs $G = \langle X, E, F, V, o\rangle$ and $G' = \langle X', E', F', V', o'\rangle$ are plane-isomorphic or not.

Finally, almost as corollaries, this will ensure the tractability of both the following problems:

Problem: Open Plane Graphs Sphere-Isomorphism (Opgsi)
Instance: two open plane graphs G and G'
Question: are G and G' sphere-isomorphic?

Problem: Open Plane Graphs Equivalence (Opge)
Instance: two open plane graphs G and G'
Question: are G and G' equivalent?

Theorem 6. *Both problems* Opge *and* Opgsi *are in class $\mathcal{P}$.*

The remainder of this section is devoted to the proof of all these results.

4.3. Deciding Plane-Isomorphism in Polynomial Time

The proof of Theorem 5 is similar to the one which can be found in [17] concerning the tractability of the isomorphism problem for *combinatorial maps.* The idea is to use the arcs (called *darts* in [17], *half-edges* in [10]) in order visit both graphs G and G' in parallel.

More precisely, suppose that the faces of a plane graph has to be described using a plane graph system. Then each edge $e = \{x, y\}$ is used twice, once when "walking" from vertex x to vertex y, and once from y to x. In other words, we implicitly consider that an edge is composed of two arcs, xy and yx, with opposite direction (see Figure 26). Given an open plane graph $G = \langle X, E, F, V, o\rangle$, we shall denote by A the set of arcs, that is, $A = \{xy \in X^2 : \exists f \in F, \exists u \in X^*, \mathbf{f} = [xyu]\}$.

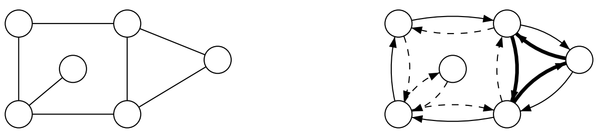


Figure 26: Any edge is met twice, thus composed of two arcs (represented with different line styles), when used to describe the faces of a plane graph.

We now introduce two low-level operations over the set of arcs, called next : $A \to A$ and opp : $A \to A$ (following the terminology of [10]), that will be used to traverse graph G:

- Given an arc $xy \in A$, we denote by opp(xy) the counter-arc (or *opposite* arc) yx;

- Let $xy \in A$ and suppose that $[xyu] = \mathbf{f}$ for some $f \in F, u \in X^*$. Then function next(xy) returns either yx if $u = \epsilon$, or yz if $u = zu'$ for some $z \in X, u' \in X^*$.

For example, in Figure 27, we have next$(ec) = cf$ and next$(cf) = fc$ and next$(fc) = cd$... In other words, one visits the boundary of a face by iterating function next. As for function opp, it allows one to swap from a face to an adjacent face. Note that in [17], function next is called β_1 and function opp is called β_2.

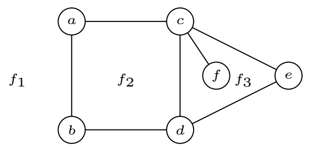


Figure 27:

We now reformulate the plane-isomorphism property in terms of arcs:

Lemma 4. *Let $G = \langle X, E, F, V, o\rangle$ and $G' = \langle X', E', F', V', o'\rangle$ be two open plane graphs having A and A' as sets of arcs respectively. If G and G' are not reduced to isolated vertices, then G and G' are plane-isomorphic if and only if there exists a one-to-one mapping $\rho : A \to A'$ such that*

1. *function ρ commutes with functions next and opp: $\forall a \in A$,*

$$\rho(\mathit{next}(a)) = \mathit{next}(\rho(a)) \text{ and } \rho(\mathit{opp}(a)) = \mathit{opp}(\rho(a));$$

2. *function ρ preserves both the visible faces and the outer faces: $\forall a \in A$,*

$$(\underrightarrow{a} \in V \text{ iff } \underrightarrow{\rho(a)} \in V') \text{ and } (\underrightarrow{a} = o \text{ iff } \underrightarrow{\rho(a)} = o').$$

Proof. ($\Rightarrow$) Suppose that $G \equiv_p G'$ using functions $\langle \xi, \chi\rangle$ with $\chi : X \to X'$ (see Definition 6). For all arcs $a = xy$, we define $\rho(h) = \chi(x)\chi(y)$ and the conditions hold. ($\Leftarrow$) As graph G is connected, every vertex $x \in X$ is the extremity of an edge $\{x, y\}$ for some $y \in X$, thus of an arc xy. Assuming that $\rho(xy) = x'y'$, we fix $\chi(x) = x'$. As for any face $f = \underrightarrow{xy}$, we fix $\xi(f) = \underrightarrow{\chi(x)\chi(y)}$. Then the conditions of Definition 6 immediately hold, thus $G \equiv_p G'$. □

We are ready to tackle the proof of Theorem 5. Consider Algorithms 28 and 29. The former first fixes an arc $a_0 \in A$ lying on the boundary of the outer face of G. Then, for every arc $a'_0 \in A'$ of the outer face of G', we call Algorithm 29 to build a candidate matching function $f : A \to A'$, and finally checks whether f satisfies the conditions of Lemma 4. Algorithm 29 performs a traversal of graph G, starting from arc a_0, and using functions next and opp to discover new darts from discovered darts. Initially, $f[a_0]$ is set to a'_0 whereas $f[a]$ is set to *nil* for all the other arcs. Each time an arc $a' \in A$ is discovered from another arc $a \in A$ using function next (*resp.* opp), $f[a']$ is set to arc next($f[a]$) (*resp.* opp($f[a]$)).

```
Input: two open plane graphs $G = \langle X, E, F, V, o\rangle$ and
       $G' = \langle X', E', F', V', o'\rangle$ having $A$ and $A'$ as sets of arcs
Output: TRUE if $G \equiv_p G'$, FALSE otherwise
choose $a_0 \in A$ such that $\overrightarrow{a_0} = o$ ;
foreach $a'_0 \in A'$ such that $\overrightarrow{a'_0} = o'$ do
    $f \leftarrow$ traverseAndBuildMatching($G, G', a_0, a'_0$);
    if $f$ satisfies the conditions of Lemma 4 then
        return TRUE
return FALSE
```

Figure 28: checkPlaneIsomorphism(G, G')

```
Input: two open plane graphs $G = \langle X, E, F, V, o\rangle$ and
       $G' = \langle X', E', F', V', o'\rangle$ and two arcs $a_0 \in A$ and $a'_0 \in A'$
Output: returns an array $f : A \rightarrow A'$
foreach arc $a \in A$ do $f[a] \leftarrow nil$;
$f[a_0] \leftarrow a'_0$;
let $S$ be an empty stack;
push $a_0$ in $S$;
while $S$ is not empty do
    pop an arc $a$ from $S$;
    if $f[\text{next}(a)] = nil$ then
        $f[\text{next}(a)] \leftarrow \text{next}(f[a])$;
        push next($a$) in $S$
    if $f[\text{opp}(a)] = nil$ then
        $f[\text{opp}(a)] \leftarrow \text{opp}(f[a])$;
        push opp($a$) in $S$
return $f$
```

Figure 29: traverseAndBuildMatching(G, G', a_0, a'_0)

Proof of Theorem 5. Essentially the same as that of [17]. If procedure checkPlaneIsomorphism(G, G') returns TRUE, then there exists $f : A \rightarrow A'$ that fulfils the conditions of Lemma 4, thus $G \equiv_p G'$. Conversely, assume that $G \equiv_p G'$. Then there exists a function $\rho : A \rightarrow A'$ that satisfies the conditions of Lemma 4. Let $a_0 \in A$ be the arc chosen at line 1 of Algorithm 28 (such that $\underrightarrow{a_0} = o$). As the loop (lines 2-5) iterates on every arc $a'_0 \in A'$ that lies on the outer face of G', there is an iteration for which $a'_0 = \rho(a_0)$. We claim that for this iteration, traverseAndBuildMatching(G, G', a_0, a'_0) returns f such that for all $a \in A$, $f[a] = \rho(a)$. Indeed, we have the two following properties:

1. *When pushing an arc a in S, $f[a] = \rho(a)$.* This is true for the push of line 4 as $f[a_0]$ is set to $a'_0 = \rho(a_0)$ at line 2. This is also true for the push of line 9 as $f[\text{next}(a)]$ is set to $\text{next}(f[a])$ at line 8, and $f[a] = \rho(a)$ (by induction hypothesis), and $(\rho(a) = a' \Rightarrow \rho(\text{next}(a)) = \text{next}(a'))$ (by Lemma 4). And the same for function opp.
2. *Each arc $a \in A$ is pushed once and only once in S.* Indeed, G is connected. So there exists at least one sequence $a_0, \dots, a_n$ such that $a_n = a$ and, for all $k \in \{1, 2, \dots n\}$, either $a_k = \text{next}(a_{k-1})$, or $a_k = \text{opp}(a_{k-1})$. Therefore, each time an arc a_i of this sequence is popped from S (line 6), a_{i+1} is pushed in S (lines 9 and 12) if it had not been pushed before through another path (lines 7 and 10).

In consequence, checkPlaneIsomorphism(G, G') will return True.

Finally, concerning complexity issues, Algorithm 29 is in $\mathcal{O}(|A|)$ time. Indeed, the while loop is iterated $|A|$ times as (1) exactly one arc a is removed from stack S at each iteration, and (2) each arc $a \in A$ is pushed in S at most once. In Algorithm 28, the test of line 4 can be performed in $\mathcal{O}(|A|)$ time. Hence, the overall time complexity of Algorithm 28 is $\mathcal{O}(|A| \cdot |A'|)$, that is, $\mathcal{O}(|E| \cdot |E'|)$. □

4.4. Deciding Sphere-Isomorphism and Equivalence in Polynomial Time

The tractability of sphere-isomophism (claimed in Theorem 6) follows from what we have just done for plane-isomorphism. Indeed, Lemma 4, without the condition concerning the outer faces, holds for sphere-isomorphism. So, one just has to use algorithm checkPlaneIsomorphism(G, G'), leaving out the verification of the preservation of the outer faces at line 4, and gets an algorithm that solves problem Opgsi in polynomial time.

Concerning the equivalence problem, let us describe the procedure, based on Theorem 3. The first step consists in testing the outer faces: if one is visible and the other is not, then both graphs are not equivalent. Otherwise, we need to normalise, in polynomial time, the graphs by using Algorithm 18 and get their normal forms. Then, there are two cases. If the outer faces are both invisible, then the graphs are equivalent if and only if their normal forms are sphere-isomorphic, which can be decided in polynomial time. And if the outer faces are visible, then the graphs are equivalent if and only if their irreducible forms are plane-isomorphic, which can also be checked in polynomial time by Theorem 5. So Problem Opge is solvable in polynomial time.

4.5. The Tractability of the Subisomorphism Problem

The proof of Theorem 4 rests on two procedures that allow one to check whether an irreducible open plane graph is a pattern of another graph (see Algorithm 30 and 31). Procedure checkPattern(P, G)

1. maps all the visible faces of P with faces of G (that may be visible or not, at this point),
2. checks that the selected faces in G forms a contiguous set of visible faces,
3. normalises the graph built from G using the selected faces and
4. checks that both the latter graph and pattern P are plane-isomorphic.

Concerning Steps 2 to 4, we simply follow the definition of a pattern and reuse Procedure checkPlaneIsomorphism (see Algorithm 28). As for Step 1, we use a variant of Procedure traverseVisibleFacesAndBuildMatching (see Algorithm 31) where we restrict the traversal of the pattern to its visible faces only (that is, we modify line 10 only).

This algorithm is only designed to establish the polynomiality of Problem Popg, thus is not optimized.

```
Input: an irreducible open plane graphs $P = \langle X, E, F, V, o\rangle$ and an open
       plane graph $G = \langle X', E', F', V', o'\rangle$, respectively having $A$ and $A'$
       as sets of arcs
Output: True if $P$ is a pattern of $G$, False otherwise
1  choose $a_0 \in A$ such that $\underrightarrow{a_0} \in V$ ;
2  foreach $a'_0 \in A'$ such that $\underrightarrow{a'_0} \in V'$ do
3      $f \leftarrow$ traverseVisibleFacesAndBuildMatching$(P, G, a_0, a'_0)$;
4      $W \leftarrow \emptyset$;
5      foreach $a \in A$ such that $\underrightarrow{a} \in V$ do
6          $W \leftarrow W \cup \{\underrightarrow{f[a]}\}$
7      if $W \subseteq V'$ and $W$ is a set of contiguous faces in $G$ then
8          $G'' \leftarrow \langle X', E', F', W, o'\rangle$;
9          $G'' \leftarrow$ normaliseGraph$(G'')$;
10         if checkPlaneIsomorphism$(P, G'')$ then return True
11 return False
```

Figure 30: checkPattern(P, G)

5. Searching for Piecewise Patterns

Open plane graphs are defined by declaring that some faces are visible, and the others invisible. This distinction is essential to define the patterns and the subisomorphism problem. However, the tractability of the latter was based on an important property: the visible faces must be *contiguous.* In this section, we study the case of patterns whose visible faces are non-contiguous. We call them *piecewise patterns.*

```
    Input: two open plane graphs $G = \langle X, E, F, V, o \rangle$ and
           $G' = \langle X', E', F', V', o' \rangle$ and two arcs $a_0 \in A$ and $a'_0 \in A'$ such
           that $\underrightarrow{a_0} \in V$ and $\underrightarrow{a'_0} \in V'$
    Output: an array $f : A \rightarrow A'$
 1  foreach $a \in A$ do $f[a] \leftarrow nil$;
 2  $f[a_0] \leftarrow a'_0$;
 3  let $S$ be an empty stack;
 4  push $a_0$ in $S$;
 5  while $S$ is not empty do
 6      pop an arc $a$ from $S$;
 7      if $f[\mathrm{next}(a)] = nil$ then
 8          $f[\mathrm{next}(a)] \leftarrow \mathrm{next}(f[a])$;
 9          push $\mathrm{next}(a)$ in $S$
10      if $\underrightarrow{\mathrm{opp}(a)} \in V$ and $f[\mathrm{opp}(a)] = nil$ then
11          $f[\mathrm{opp}(a)] \leftarrow \mathrm{opp}(f[a])$;
12          push $\mathrm{opp}(a)$ in $S$
13  return $f$
```

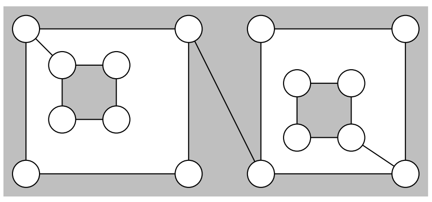
Figure 31: traverseVisibleFacesAndBuildMatching(G, G', a_0, a'_0)

5.1. The Importance of the Contiguity Assumption

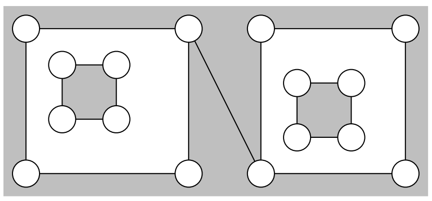
Until now, a pattern was got from an open plane graph by (1) selecting a subset of contiguous visible faces and (2) normalising, in order to “empty” the invisible area. However, this construction is not extensible to piecewise patterns, in particular because the normalisation procedure is undefined for a plane graph whose visible faces are not contiguous.

Indeed, consider the plane graph of Figure 32(a). The elimination of the bridges yields the plane graphs of Figure 32(b) which is disconnected. In consequence, some of its faces are *compound*, that is, they need several boundaries to be completely specified. Thus, they cannot be described with a plane graph system, which is nevertheless what our normalisation procedure returns.

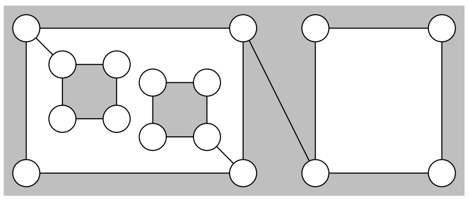
Besides, allowing non-contiguity introduces several new problems that are outside the scope of this paper. To illustrate one of these, consider the plane graph of Figure 32(c); this graph is equivalent to that of Figure 32(a) since we can pair together the visible faces with a one-to-one mapping, while preserving the boundaries. However, once normalised, the bridges are eliminated in both graphs, but the normal forms are not plane-isomorphic. Indeed, the invisible faces now play a role; however, both invisible faces of the former graph will have two boundaries after normalisation, while those of the latter graph will have either one or three boundaries.



(a) An example of (connected) plane graph with a set of non-contiguous visible faces.



(b) The plane graph 32(a) once normalised: Eliminating the bridges disconnects the graph. Such a plane graph cannot be described with a plane graph system.



(c) Another plane graph with a set of non-contiguous visible faces. This graph is equivalent to that of Figure 32(a). However, once the bridges are eliminated, we get a disconnected plane graph which is not plane-isomorphic to the plane graph of Figure 32(b).

Figure 32: About contiguity

5.2. Piecewise Patterns

Piecewise patterns are sets of disconnected plane graphs whose internal faces are all visible and outer face is shared and invisible (see Figure 33(a)):

Definition 14 (Piecewise-compact plane graph).

- An open plane graph $G = \langle X, E, F, V, o \rangle$ is *compact* if all its faces but the outer face are visible: $V = F \setminus \{o\}$.
- A *piecewise-compact* plane graph is a finite set $P = \{G_1, G_2, \dots G_k\}$ with $k \geq 2$ such that
  1. Each component $G_i = \langle X_i, E_i, F_i, V_i, o_i \rangle$ is a compact open plane graph.
  2. The components are pairwise independent: $X_i \cap X_j = \emptyset$ and $V_i \cap V_j = \emptyset$ for all $1 \leq i < j \leq k$.
  3. The components share their outer face: $o_i = o_j$ for all $1 \leq i < j \leq k$.

Given a piecewise-compact plane graph $P = \{G_1, G_2, \dots G_k\}$, we denote by $X_P = \cup_{i=1}^k X_i$ the set of vertices, $E_P = \cup_{i=1}^k E_i$ the set of edges, $F_P = \cup_{i=1}^k F_i$ the set of faces, $V_P = \cup_{i=1}^k V_i$ the set of visible faces and o_P the (invisible) outer face (with $o_P = o_i$ for any $1 \leq i \leq k$).

Now searching for a piecewise-compact plane graph in an open plane graph will consist in finding all its components independently. That is, two components do not share any vertex nor edge, so their images must have the same property. This yields the following definition:

Definition 15 (Piecewise pattern). Let P be a piecewise-compact plane graph and $G = \langle X, E, F, V, o \rangle$ an open plane graph. We say that P is a *piecewise pattern* of G if:

1. there exists an *injection* $\chi : X_P \to X$ over the vertices;
2. there exists an *injection* $\xi : V_P \to V$ over the visible faces whose boundaries are preserved: $\forall f_1 \in V_P, \forall f_2 \in V$,

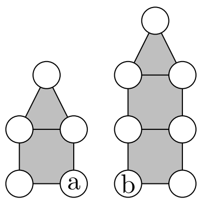
$$\text{if } \xi(f_1) = f_2 \text{ and } \mathbf{f_1} = [x_1 x_2 \dots x_p], \text{ then } \mathbf{f_2} = [\chi(x_1)\chi(x_2) \dots \chi(x_p)];$$

3. the outer face of G is not matched: $\xi^{-1}(o) = \emptyset$.

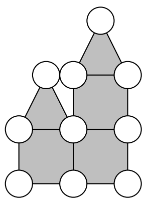
An example is given in Figure 33: the piecewise-compact plane graph 33(a) is a piecewise pattern of only the open plane graph 33(d); indeed, in the other cases, vertices a and b from Figure 33(a) should be merged, which is forbidden by Definition 15.

In this section, we are finally going to show that searching for a piecewise pattern is intractable. Formally, we consider the following problem:

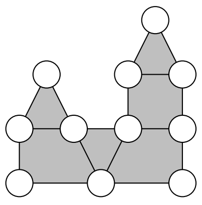
Problem: Piecewise Pattern of an Open Plane Graph (Ppopg)
Instance: a piecewise-compact plane graph P and an open plane graph G
Question: is P a piecewise-pattern of G?



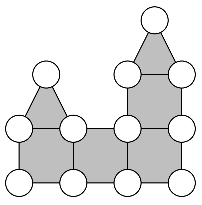
(a) A piecewise-compact plane graph with two components.



(b)



(c)



(d)

Figure 33: Piecewise-compact plane graph 33(a) is not a piecewise pattern of open plane graphs 33(b) and 33(c), because vertices a and b would have to be merged together. On the other hand, it is a piecewise pattern of plane graph 33(d).

and we get:

Theorem 7. *Problem* Ppopg *is $\mathcal{NP}$-complete.*

5.3. Proof of intractability

Problem Ppopg belongs to class $\mathcal{NP}$ since one can check in polynomial time whether two given injections χ and ξ satisfy the conditions of Definition 15. In order to prove that problem Ppopg is $\mathcal{NP}$-complete, we show that problem Planar-4 3-SAT can be reduced to it.

Planar-4 3-SAT is a special case of the SAT problem, which involves deciding if there exists a truth assignment for a set X of variables such that a boolean formula F over X is satisfied. We assume that F is in Conjunctive Normal Form (CNF), *i.e.*, it is a conjunction of clauses such that each clause is a disjunction of litterals which are either variables of X or negations of variables of X. The formula-graph associated with a CNF formula F over a set of variables X is the bipartite graph $G_{X,F} = (V, E)$ such that V associates a vertex with every variable $x_i \in X$ and every clause c_j of F, and E associates an edge (x_i, c_j) with every variable/clause couple such that variable x_i occurs in clause c_j. See Figure 34 for an instance of Planar-4 3-SAT and its associated formula-graph.

$$\begin{aligned} X &= \{x, y, z, u, w\} \\ F &= (\neg x \vee y \vee u)\wedge \\ &\quad (\neg x \vee y \vee \neg z)\wedge \\ &\quad (\neg y \vee z \vee u)\wedge \\ &\quad (\neg z \vee u \vee \neg w)\wedge \\ &\quad (x \vee w \vee \neg u) \end{aligned}$$

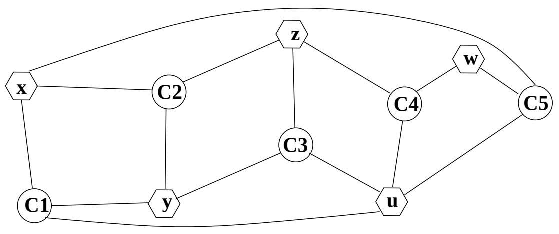


Figure 34: An instance of Planar-4 3-SAT and its associated formula graph.

Problem Planar-4 3-SAT is formally defined as follows:

Problem: Planar-4 3-SAT
Instance: A CNF formula F over a set of variables X such that (1) every clause of F is a disjunction of 3 litterals, (2) the formula-graph $G_{X,F}$ is planar, and (3) the degree of every vertex of $G_{X,F}$ is bounded by 4 (*i.e.*, each variable occurs in at most 4 different clauses)
Question: Does there exist a truth assignment for X which satisfies F?

Planar-4 3-SAT has been shown to be $\mathcal{NP}$-complete in [25]. To reduce an instance (X, F) of Planar-4 3-SAT to an instance (P, G) of Ppopg, we first perform a preprocessing to eliminate variables which occur in only one clause of F: We iteratively eliminate from (X, F) every variable $x_i \in X$ which occurs in only one clause c_j of F (those whose degree is equal to 1 in the formula-graph), set x_i to the truth value which satisfies c_j, and eliminate c_j from F, until either X and F become empty (thus showing that the intial instance is trivially consistent), or all variables in X occur in 2, 3, or 4 clauses of F.

Let us now show how to build a piecewise-compact plane graph P and an open plane graph G by combining building blocks (which are open plane graphs). Table 1 displays the building blocks (gadgets) associated with the variables and the clauses:

- For each variable $x_i \in X$ such that the degree of x_i in the formula-graph $G_{X,F}$ is equal to k (with $2 \leq k \leq 4$), we build two variable patterns V_k and V'_k which will respectively occur in G and P. These variable patterns look like flowers which have $2k$ petals in G and k petals in P, where each petal is a 6-edge face. The core of the flower is composed of two adjacent 5-edge faces.
- For each clause, we build two clause patterns C and C' which will respectively occur in G and P. In G, the clause pattern is composed of a 3-edge central face which has 3 adjacent 4-edge faces whereas in P it is composed of a 3-edge face which has 1 adjacent 4-edge face.

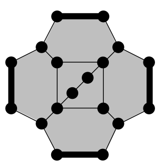
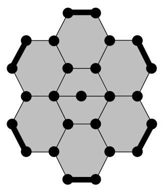
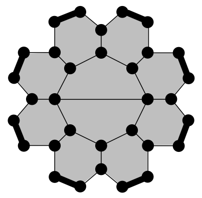
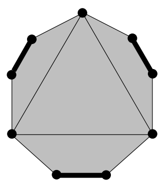
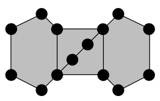
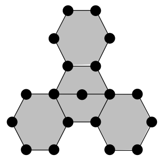
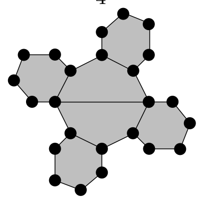
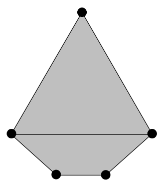
	Variable patterns			Clause patterns
in G:	V_2 : 	V_3 : 	V_4 : 	C : 
in P:	V'_2 : 	V'_3 : 	V'_4 : 	C' : 

Table 1: Variable and clause patterns used as building blocks to define G and P. Connecting edges in G are displayed in bold.

Variable and clause patterns in G are plugged together by merging some edges, called connecting edges, to form an open plane graph. These connecting edges are displayed in bold in Table 1: For each petal in each variable pattern V_k, the edge opposite to the core of the flower is a connecting edge and, for each 4-edge face in the clause pattern C, the edge opposite to the 3-edge face is also a connecting edge.

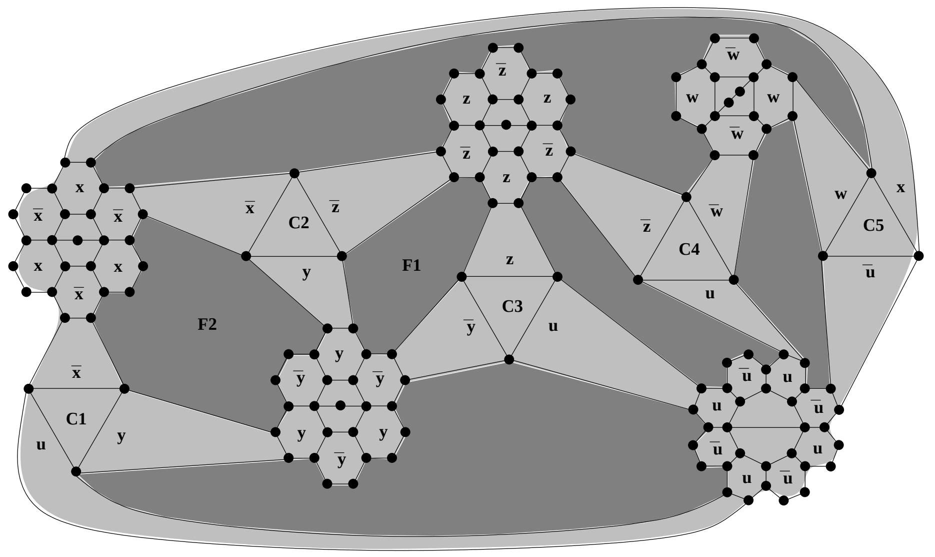


Figure 35: Open plane graph G associated with the SAT instance displayed in Figure 34. Faces in light grey correspond to variable and clause patterns; faces in dark grey are new faces which have been created when merging edges of variable and clause patterns. These new faces have at least 8 edges. They have 8 edges when variable and clause patterns are connected by adjacent petals for the two variables (see, e.g., face F_1); they have more than 8 edges when they are connected by non adjacent petals (see, e.g., face F_2).

Given these building blocks, we define G and P as follows:

Open plane graph G: For each variable $x_i \in X$ such that the degree of x_i in the formula-graph $G_{X,F}$ is equal to k, G contains an occurrence of the variable pattern V_k. Each petal of this occurrence of V_k is alternatively labeled with x_i and $\neg x_i$. For each clause c_j of F, G contains an occurrence of the clause pattern C. Each 4-edge face of this occurrence of C is labeled with a different literal of c_j. Variable and clause patterns are connected to define an open plane graph by merging every connecting edge of each clause pattern with a different connecting edge of a variable pattern such that the two faces which become adjacent by this merge are labeled with the same litteral. We can easily check that this open plane graph can always be built as the formula-graph $G_{X,F}$ is planar. Figure 35 displays the open plane graph associated with the formula displayed in Figure 34. Note that new faces have been created when merging connecting edges of variable patterns with connecting edges of clause patterns. These new faces have at least 8 edges (see Figure 35).

Piecewise-compact plane graph P: If the SAT instance has n variables and c clauses, then P is composed of $n + c$ different components: a component V'_k is associated with every variable $x_i \in X$, where k is the degree of x_i in the formula-graph $G_{X,F}$; a component C' is associated with every

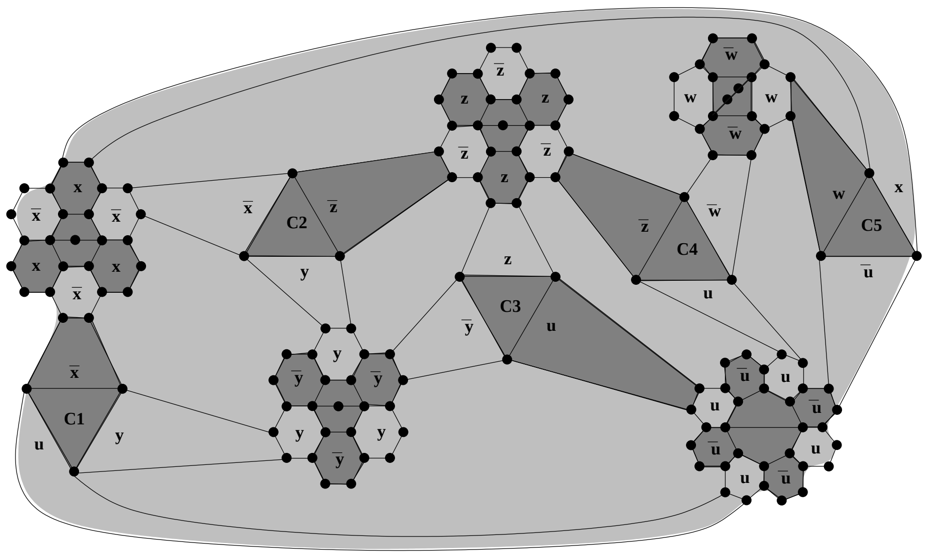


Figure 36: Solution of the Ppopg instance associated with the Planar-4 3-SAT instance displayed in Figure 34. The images of the components of P in G are displayed in dark grey.

clause. For example, the piecewise-compact plane graph associated with the formula displayed in Figure 34 contains 10 components: 3 occurrences of V'_3, 1 occurrence of V'_4, 1 occurrence of V'_2, and 5 occurrences of C'.

Let us show that P is a piecewise pattern of G iff there exists a truth assignment of X which satisfies F.

($\Rightarrow$). Let us assume that there exist two injections χ and ξ which satisfy the conditions of Definition 15, and let us show that there exists a truth assignment of X which satisfies F. ξ matches every visible face of P to a different visible face of G so that the images of the components of P do not share any vertex nor edge. Figure 36 displays an example of such a solution for the instance (P, G) of Ppopg associated with the instance (X, F) of Planar-4 3-SAT displayed in Figure 34. P contains c occurrences of C', where c is the number of clauses of F. Each occurrence of C' has a 3-edge face adjacent to a 4-edge face. These faces can only be matched to faces which belong to occurrences of C in G as all other faces in G have a different number of edges (5 for the cores of the flowers, 6 for the petals of the flowers, and 8 or more for the new faces which are created when merging variable and clause patterns). As there are c occurrences of C in G, each occurrence of C' in P is matched with a different occurrence of C in G. For the same reasons, each occurrence of a variable pattern V'_k in P is matched with a different occurence of a variable pattern V_k in G: Petal and core faces in P can only be matched with petal and core faces in G, and an occurrence of V'_i cannot be matched with faces of an occurrence of V_j if $i \neq j$.

For each variable pattern V_k, the label of the petals of V_k which are not matched with petals of variable patterns of P gives the truth assignment for the corresponding variable. For each clause pattern C, the label of the 4-edge face of C which is matched with a 4-edge face of C' corresponds to a litteral which satisfies the clause associated with C. As the images in G of the components of P by ξ cannot share edges, we ensure that when a 4-edge face is matched, then the adjacent petal is not matched, *i.e.*, when a clause is satisfied by a litteral l, then no other clause can be satisfied by the negation of this litteral so that the truth assignment deduced from the flower matching actually satisfies all clauses of F.

For example, the truth assignment corresponding to the solution displayed in Figure 36 is $\{\neg x, y, \neg z, u, w\}$.

($\Leftarrow$). Let us assume that there exists a truth assignment of X which satisfies F and let us show that there exist two injections χ and ξ which satisfy the conditions of Definition 15. For each variable pattern V'_k in P associated with a variable x_i, we match the two core 5-edge faces with the two core 5-edge faces of the variable pattern associated with x_i in G and we match the k 6-edge petals of V'_k with the k 6-edge petals which are labeled with the negation of the truth value of x_i. For each clause pattern C' in P associated with a clause c_j, we match the 3-edge face of C' with the 3-edge face of the clause pattern associated with c_j in G and we match the 4-edge face of C' with one of the three 4-edge faces: We choose a 4-edge face which is labeled with a literal which is satisfied by the truth assignment (this 4-edge face cannot be adjacent to a matched 6-edge petal). From the face matching ξ, we can easily deduce a vertex matching χ.

6. Conclusion

Open plane graphs have been introduced in order to define embedded planar graphs in which faces can be visible or invisible. Invisible faces define holes, obtained by removing faces, or stating that certain faces are unreachable. This induces a special equivalence relation over open plane graphs for which a normal form exists and can be computed.

Beyond, open plane graphs allowed us to investigate a new class of subgraphs, called patterns, by declaring that some faces of a plane graph are invisible. We have shown that searching for patterns was tractable in polynomial time as soon as the selected faces are contiguous. This result is quite strong since we have also established that the search of piecewise patterns, a slight generalization of patterns, was a $\mathcal{NP}$-complete problem.

A certain number of questions remain open and are of interest:

- When introducing open plane graph systems, the hypothesis was that the system is built from the graph, once drawn. But the converse question remains to be explored: given an open graph system respecting syntaxically Definition 3, can a graph always be drawn, and how?

- For practical reasons approximate isomorphisms would be of interest: this is related to having a definition of an edit distance. Both definitions would need to take into account normalisation.
- In many graph mining applications, the important question is that of extracting maximal common subgraphs of various graphs. The status of the problem, for the definitions introduced in this paper, is left open.
- When introducing piecewise patterns in Section 5, it was shown (see Figure 32) that several notions introduced earlier had to be updated in the case of non connected graphs. There are a number of complex issues related with this question.
- If open plane graphs are to be used to model images, alternative problems appear: a first one is to find, given two open plane graphs, the largest common pattern. The largest common piecewise pattern is also of interest.

Acknowledgments

The authors would like to thank Daniel Gonçalves (University of Montpellier) for his pointer to problem Planar-3SAT, and fruitful remarks.

References

[1] M. D. Santo, P. Foggia, C. Sansone, M. Vento, A large database of graphs and its use for benchmarking graph isomorphism algorithms, Pattern Recognition Letters 24 (2003) 1067–1079.

[2] K. Riesen, H. Bunke, Iam graph database repository for graph based pattern recognition and machine learning, in: Proc. 12th IAPR International Workshop on Structural and Syntactic Pattern Recognition (SSPR'08), volume 5342 of *Lecture Notes In Computer Science*, Springer-Verlag, 2008, pp. 287–297.

[3] D. Conte, P. Foggia, C. Sansone, M. Vento, How and why pattern recognition and computer vision applications use graphs, in: Applied Graph Theory in Computer Vision and Pattern Recognition, Springer, 2007, pp. 85–135.

[4] E. Samuel, C. de la Higuera, J.-C. Janodet, Extracting plane graphs from images, in: Proc Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition (SSPR&SPR'10), volume 6218 of *Lecture Notes in Computer Science*, Springer, 2010, pp. 233–243.

[5] D. Conte, P. Foggia, C. Sansone, M. Vento, Thirty years of graph matching in pattern recognition, International Journal of Pattern Recognition and Artificial Intelligence 18 (2004) 265–298.

[6] B. McKay, Practical graph isomorphism, Congressus Numerantium 30 (1981) 45–87.

[7] S. Zampelli, Y. Deville, C. Solnon, S. Sorlin, P. Dupont, Filtering for subgraph isomorphism, in: Proc. 13th Principles and Practice of Constraint Programming (CP'07), volume 4741 of *LNCS*, Springer, 2007, pp. 728–742.

[8] S. Sorlin, C. Solnon, A parametric filtering algorithm for the graph isomorphism problem, Constraints 13 (2008) 518–537.

[9] L. P. Cordella, P. Foggia, C. Sansone, M. Vento, An improved algorithm for matching large graphs, in: Proc. 3rd IAPR International Workshop on Graph-based Representations in Pattern Recognition (GbR'01), volume 2059 of *LNCS*, Springer, 2001, pp. 149–159.

[10] E. Fusy, Combinatoire des graphes planaires et applications algorithmiques, Ph.D. thesis, Ecole Polytechnique, 2007.

[11] R. Cori, Un code pour les graphes planaires et ses applications, in: Astérisque, volume 27, Société Mathématique de France, Paris, France, 1975.

[12] R. Cori, Computation of the Automorphism Group of a Topological Graph embedding, Technical Report I-8612, UER de mathématique et informatique, CNRS équipe du laboratoire associé 226, Université Bordeaux I, 1985.

[13] F. Dorn, Planar subgraph isomorphism revisited, in: Proc. 27th International Symposium on Theoretical Aspects of Computer Science (STACS'10), volume 5 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010, pp. 263–274.

[14] D. Eppstein, Subgraph isomorphism in planar graphs and related problems, Journal of Graph Algorithms and Applications 3 (1999) 1–27.

[15] T. B. W. Jones, S. Mitchel, Linear algorithms for isomorphism of maximal outerplanar graphs, Journal Association Computer Machine 26 (1979) 603–610.

[16] G. Damiand, C. de la Higuera, J.-C. Janodet, E. Samuel, C. Solnon, A polynomial algorithm for submap isomorphism: Application to searching patterns in images, in: Proceedings 7th IAPR Int. Workshop on Graph-based Representation in Pattern Recognition (GbR'09), volume 5534 of *LNCS*, Springer, 2009, pp. 102–112.

[17] G. Damiand, C. Solnon, C. de la Higuera, J.-C. Janodet, E. Samuel, Polynomial algorithms for subisomorphism of nd open combinatorial maps, Computer Vision and Image Understanding 115 (2011) 996–1010.

[18] X. Jiang, H. Bunke, Marked subgraph isomorphism of ordered graphs, in: Advances in Pattern Recognition, volume 1451 of *LNCS*, Springer, 1998, pp. 122–131.

[19] X. Jiang, H. Bunke, Optimal quadratic-time isomorphism of ordered graphs, Pattern Recognition 32 (1999) 1273–1283.

[20] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach (3rd ed.), Pearson, 2010.

[21] M. Poudret, A. Arnould, Y. Bertrand, P. Lienhardt, Cartes Combinatoires Ouvertes, Research Notes 2007-1, Laboratoire SIC E.A. 4103, F-86962 Futuroscope Cedex - France, 2007.

[22] T. Nishizeki, M. S. Rahman, Planar graph drawing, volume 12 of *Lecture notes series on computing*, World Scientific, 2004.

[23] I. Fàry, On straight-line representation of planar graphs, Acta Scientiarum Mathematicarum 11 (1948) 229–233.

[24] P. Lienhardt, N-dimensional generalized combinatorial maps and cellular quasi-manifolds, International Journal of Computational Geometry and Applications 4 (1994) 275–324.

[25] K. Jansen, H. Müller, The minimum broadcast time problem for several processor networks, Theoretical Computer Science 147 (1995) 69–85.