Received <day> <Month>, <year>; | Revised <day> <Month>, <year>; | Accepted <day> <Month>, <year>

DOI: xxx/xxxx

**ARTICLE TYPE**

# An algorithm for the determination of graphs associated to fold maps between closed surfaces †

Pantaleón D. Romero*[1] | Jéferson R. P. Coêlho[2,3] | Catarina Mendes de Jesus S.[3]

[1]ESI International Chair@CEU-UCH. Departamento de Matemáticas, Física y Ciencias, Tecnológicas. Universidad Cardenal Herrera-CEU,CEU Universities. Email: pantaleon.romero@uchceu.es

[2]Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro. R. Marquês de São Vicente, 225, Rio de Janeiro. Email: jcoelho@tecgraf.puc-rio.br

[3] Departamento de Matemática, Universidade Federal de Juiz de Fora, 36036900, Juiz de Fora - MG, Brazil. Email: cmendesjesus@gmail.com

**Correspondence**
Pantaleón D. Romero,
Departamento de Matemáticas, Física y Ciencias, Tecnológicas.
Universidad Cardenal Herrera-CEU.
C/ San Bartolomé, 55
46115
Alfara del Patriarca (Spain)
Email: pantaleon.romero@uchceu.es.

**Abstract**

The aim of this paper is to introduce a computational tool that checks theoretical conditions in order to determine whether a weighted graph, as a topological invariant of stable maps, can be associated to stable maps without cusps (i.e. fold maps) from closed surfaces to the projective plane[8,9].

**KEYWORDS:**
Fold maps, graphs, topological aspects of graph theory, signed and weighted graphs, graph algorithms design.

**AMS subject classification**:00-01 99-00

## 1 | INTRODUCTION

Different techniques for image segmentation or scene illumination in 3D depends on the type of images and the desired objectives[2]. Most of them are based on the contour theory of Marr-Hildreth[1,3] whose fundamental hypothesis states that the relevant information of an image is represented in the trajectory of the apparent contours of physical objects in the image. If we photograph a dark object on a white background, we can identify this object by the silhouettes that form curves in which the light intensity changes abruptly. These curves are called the *apparent contour* and, in many cases, may contain cusps.

From a formal point of view, there are studies on apparent contours focused on images about maps between surfaces (see[4,5]) that classify these applications. In the case of stable maps without cusps, known as “fold maps", the apparent contour is the image of the singular set.

In[11], the authors introduced graphs associated to stable maps from closed and orientable surfaces to the plane, as a global invariant of these maps. Graphs properties were determined in[6,7], that can be associated with stable maps with zero cusps from closed orientable surfaces to the plane, sphere and connected sum of a torus.

†This work has been supported by grant Generalitat Valenciana (GV/2010/066), PRECEU-UCH INDI 18/11 and ESI International Chair@CEU-UCH.

In the case of non orientable surfaces,[8,9], the authors introduced graphs with pairs of weights on the vertices, associated to stable maps from closed surfaces to the projective plane, where the edges and vertices of this graph correspond to the singular curves and the connected components of the non-singular (regular) set; each vertex of the graph receives weights $(t, 0)$ if it is an orientable connected component with genus $t$ and the weight $(0, p)$ if it is a non-orientable connected component with genus $p$. In[9] was proved that all graph with weights on their vertices could be associated with some stable map of a closed surface on the projective plane.

The case of fold maps (stable maps without cusps) was treated in[8,12], where three definitions of graphs: balanced graph, quasi-balanced graph, pre-balanced graph for these maps were introduced. It has been proven that any pre-balanced graph can be associated with some map from a closed and oriented surface to the plane. Balanced graph, quasi-balanced graph and pre-balanced graph are sufficient conditions to associate a graph with a fold map of a closed and oriented surface on the projective plane. The difficulty arises when we have to classify a given graph in one of those mentioned above.

Our goal is to study if there exists or not a fold map associated with a given graph. The problem is how to verify such conditions in practice. If we have a graph with $V$ vertices and $E$ edges, the number of subgraphs that can be checked is $2^{|E|}$, each one with a combinatorial number of possible weights. Hence, we are also concerned with the computational time. In order to simplify the calculus, we have designed a program called DFMG[1] (Determining Fold Maps Graphs) in C++ language using Gurobi libraries[10]. The purpose of this algorithm is to check if a given graph is quasi-balanced or not, providing the possible auxiliary quasi-balanced subgraphs. This algorithm has been developed in parallel with the theoretical framework of searching maps between closed surfaces and in particular for fold maps.

For fold maps, with apparent planar contour of closed surfaces on the projective plane, it was proved in,[8,12] that graphs containing bipartite balanced subgraphs are associated with fold maps. However, there are fold map graphs that do not contain bipartite balanced subgraphs. This fact leads to define the concept of quasi balanced graphs, modifying the weights on the vertices, extending the aforementioned set, but it no contains all fold maps graphs. To complete the last set, we need to compute on of the pre-balanced graphs which involves graph surgeries. It remains as an open problem.

The rest of the paper is organised as follows: in (§2), we introduce the basic concepts about graphs associated to fold maps; in (§3), we review the important features about graphs related with fold maps ; in (§4), we describe the algorithm DFMG for determining fold maps; we detail the aspects and conditions of the program DFMG from a computational point of view as a linear program for solving balanced and quasi-balanced graph case . Finally, in (§5), we draw results and conclusions.

## 2 | GRAPHS LINKED TO FOLD MAPS WITH APPARENT PLANAR CONTOUR

Let $M$ be a smooth closed orientable surface. Let $\mathbb{P}$ be the real projective plane. Two smooth maps $f, g : M \to \mathbb{P}$ are **$\mathcal{A}$-equivalent** (or **equivalent**) if there are orientation-preserving diffeomorphisms, $l$ and $k$, such that $g \circ l = k \circ f$. A map $f : M \to \mathbb{P}$ is said to be **stable** if all maps are sufficiently close to $f$, in the Whitney $C^\infty$-topology are equivalent to $f$. The singularities of a stable map $f$ are locally fold type $(x, y) \mapsto (x^2; y)$, and cusp type $(x, y) \mapsto (x^3 + yx, y)$ (for more details we refer the reader to[4,5]). A **fold map** $f : M \to \mathbb{P}$ is a stable map without cusps, so that the branch set consists of curves immersed in the projective plane.

Let $f : M \to \mathbb{P}$, be a fold map. We denote by $\Sigma f$ the **singular set** of a map $f$ and its image $f(\Sigma f)$ is the **apparent contour or branch set** of $f$. We will denote by $D_P$ a simply connected region (homeomorph to a disc) on the projective plane $\mathbb{P}$, by $F_P$ the strip of Möebius, the bolt of the complement of $D_P$ in $\mathbb{P}$.

**Definition 1.** If $f : M \to \mathbb{P}$, is a fold map. We say that $f$ ***has an apparent planar contour*** if there is a homotopy $H : [0, 1] \times M \to \mathbb{P}$ is a diffeomorphism $\phi : M \to M$ such that $H_0 = f$, $H_1 = h$, $h(\Sigma h) \subset D_P$ and $\phi(\Sigma f)$ is equivalent $\phi(\Sigma h)$. If it also satisfies that $h(M) \subset D_P$ then we say that $f$ is a ***planar fold map*** (see Figure 1).

*The singular set of a fold map* $\Sigma f$, is the union of embedded closed curves on $M$ and $Bf$ is a union of smooth curves on $\mathbb{P}$ with transverse double points (does not have cusp points). The regular set (which is immersed into the surface $\mathbb{P}$ by the map) consists of finitely many regions.

In[8], the authors related the graph with pair of weights on its vertices with the pair $(M, \Sigma f)$, where each component of $\Sigma f$ is associated to an edge $a$, and each component $U$ of $M \setminus \Sigma f$ corresponds to a vertex $v$ on the graph. *An edge is incident to a*

[1] http://www.freedownload64.com/details/147991/software-for-determining-balanced-graphs.html

*vertex*, if and only if, the singular curve corresponding to the edge lies in the boundary of the regular region corresponding to the vertex. An edge always determines a loop because it matches a unique vertex of the graph. Each vertex $v$ of the graph receives a weight $(t, 0)$ to an orientable connected component with genus $t$ and a weight $(0, p)$ to a non-orientable connected component with genus $p$ (see Figure 1). In some cases, a singular curve can be on two boundaries of the same regular component, and this corresponds to a loop on the graph.

We denote by $\mathcal{G}_{(T,P)}(V, E)$ the graph with $V$ vertices and $E$ edges linked to $M$. Let $(T, P)$ be the total sum of the graph weights. The genus of $M$ is computed as follows ([8,9]):

$$g(M) = \begin{cases} 1 - V + E + T, & \textit{if } M \textit{ is orientable.} \\ 2(1 - V + E + T) + P, & \textit{if } M \textit{ is non orientable.} \end{cases}$$

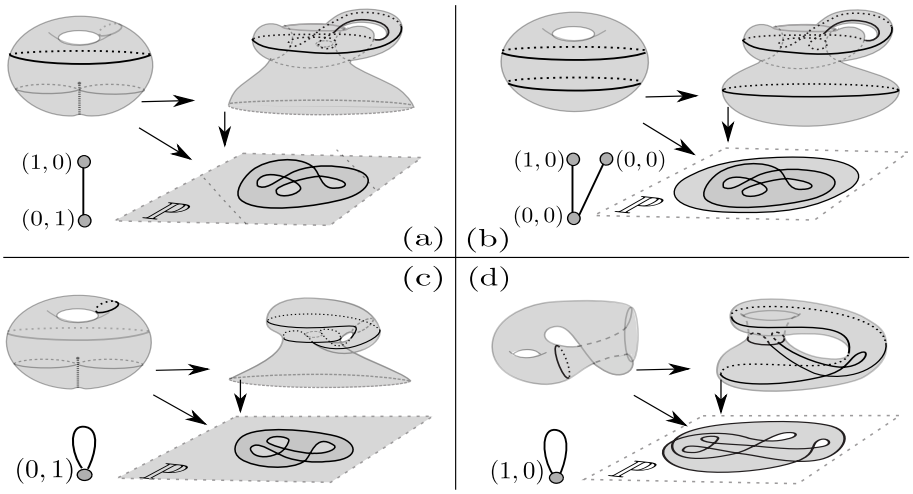


**FIGURE 1** Examples of fold maps with apparent planar contour.

Figure 1, shows four different fold maps on the projective plane, as a composition of a surface immersion on the space and its projection on the plane: (a) fold map with apparent planar contour, where $M$ is the sum of three projective planes, the singular set has one curve detaching from one side, a torus with a hole, on the other side, a projective plane with a hole (Möebius band); (b) planar fold map, where $M$ is the torus, two singular curves that detaches the tree regular regions: a torus with a hole, cylindrical and simple connected region; (c) fold map with apparent planar contour, where $M$ is the sum of three projective planes, it has a unique single curve and only regular homeomorphic region to the projective plane with two holes; (d) planar fold map, where $M$ is the sum of four projective planes, it has a unique single curve and only regular region homeomorphic to a torus with two holes.

A non-orientable surface with genus $p$ can be decomposed in different ways as the connected sum of an orientable surface of genus $q$ with a non-orientable surface of genus $r$, for pairs $q, r \in \mathbb{N}$ such that $2q + r = p$ and $r > 0$ if $p > 0$.

**Definition 2.** Two pairs $(q_1, r_1), (q_2, r_2) \in \mathbb{N} \times \mathbb{N}$ are *equivalent weights* if it satisfies one of the following conditions:

1) $r_1 = r_2 = 0$ and $q_1 = q_2$, or

2) $r_1, r_2 \neq 0$ and $2q_1 + r_1 = 2q_2 + r_2$.

**Definition 3.** We say that two graphs are *equivalent* if they are isomorphic as graphs and the corresponding vertices have *equivalent weights*.

## 3 | GRAPHS ASSOCIATED TO FOLD MAPS

In this section, we summarise the definitions that were introduced in [8,12].

A given graph $\mathcal{G}_{(T,P)}(V, E)$ is said *bipartite* if it is possible to set signs, like + and − for example, on their vertices, so that each edge connects vertices with opposite signs. In other words, a *bipartite* graph is a graph whose vertices can be divided into two disjoint and independent sets $V_1, V_2 \subset V$ such that every edge connects a vertex in $V_1$ to one in $V_2$. In a *bipartite* graph, all cycles have an even number of edges. A tree is a connected graph, has no cycles and satisfies $V = 1 + E$, because it is a bipartite graph by definition. From now on, in this paper, we use the signs + and − to designate the labels assigned to vertices in $V_1$ and $V_2$ subsets.

**Definition 4.** [8,9] A *horizontal sum of two graphs* is carried out by identifying an edge of one of them with an edge of the other one. Consequently, it creates a new graph and can be done in one of the following ways (see Figure 2):

1. The identification of two edges, both of which end at two different vertices (i.e. none of them is a loop) produces an edge that also ends at two different vertices. The weights of the involved vertices are added according to the following rules (see Figure 2):

   (a) $(s, 0) + (t, 0) = (s + t, 0)$.

   (b) $(0, p) + (0, q) = (0, p + q)$.

   (c) $(t, 0) + (0, p) = (0, 2t + p)$.

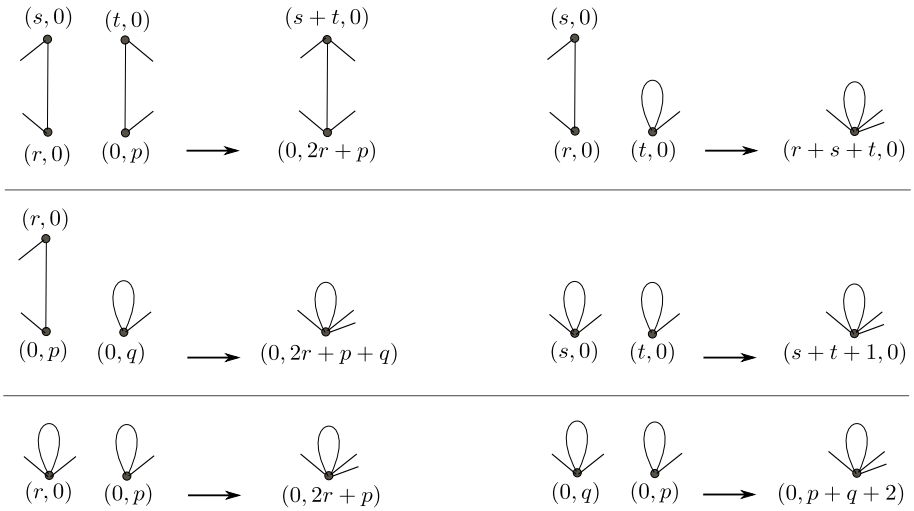


**FIGURE 2** Examples illustrating horizontal surgeries of graphs.

2. The identification of an edge that ends at two different vertices with a loop produces a loop. The weight of the corresponding vertex is the sum of the three vertices following the above rules.

3. The identification of two loops produce a loop and the weights of the vertices are the following:

   (a) $(s, 0) + (t, 0) = (s + t + 1, 0)$.

   (b) $(0, p) + (0, q) = (0, p + q + 2)$.

   (c) $(t, 0) + (0, p) = (0, 2t + p + 2)$.

## 3.1 | Balanced and quasi-balanced graphs

**Definition 5.** A *maximal bipartite subgraph* of $\mathcal{G}_{(T,P)}(V, E)$ is a graph obtained by removing an edge of each non bipartite cycle of $\mathcal{G}_{(T,P)}(V, E)$. It will be denoted by $\mathcal{G}^b_{(T,P)}$.

We will denote by $\mathcal{G}^{B}_{(T,P)}$ the set of all bipartite subgraphs of $\mathcal{G}_{(T,P)}(V,E)$.

Let us remark, if a graph has $m$ independent cycles with an odd number of edges, we can remove at least $m$ edges to obtain a bipartite graph.

**Definition 6.** Given a graph $\mathcal{G}_{(T,P)}(V,E)$ with weights $(t_i,0)$ and $(0,p_i)$ on its vertices. Let be $\mathcal{G}'_{(Q,R)}(V,E)$ the graph obtained from $\mathcal{G}_{(T,P)}(V,E)$, by decomposing its weights: $p_i = 2q_i + r_i$, with $q_i \geq 0$ and $r_i > 0$, if $p_i > 0$ and $q_i = t_i$, when $p_i = 0$. We denote by $Q = \sum q_i$ and $R = \sum r_i$. The graph $\mathcal{G}'_{(Q,R)}(V,E)$ will be called *auxiliary graph* of $\mathcal{G}_{(T,P)}(V,E)$. The set of all auxiliary graphs of $\mathcal{G}_{(T,P)}(V,E)$ will be denoted by $\mathcal{G}^{Aux}_{(T,P)}(V,E)$

*Remark 1.* If $r_i = p_i$ then $\mathcal{G}_{(T,P)}(V,E)$ and $\mathcal{G}'_{(Q,R)}(V,E)$ are equivalents, because $q_i = t_i$. Hence, the graph $\mathcal{G}_{(T,P)}(V,E)$ is contained in $\mathcal{G}^{Aux}_{(T,P)}(V,E)$.

If $\mathcal{G}'_{(Q,R)}(V,E)$ is a bipartite graph of $\mathcal{G}^{Aux}_{(T,P)}(V,E)$, we assign labels $\pm$ to its vertices, so each edge connects to vertices with opposite signs.

We denote by $V^+$ (resp $V^-$) the total number of vertices labelled with $+$ (resp. with $-$) and $Q^+$ (resp $Q^-$) the sum of all weights, $q_i$, corresponding to vertices with positive label (resp. with negative label), by $R^+$ (resp. $R^-$), the sum of all the $r_i$ weights corresponding to vertices with positive label (resp. with negative label).

*Remark 2.*

Give a graph $\mathcal{G}_{(T,P)}(V,E)$ with weights $(t_i,0)$ e $(0,p_i)$ on its vertices. Let be $\mathcal{G}'_{(Q,R)}(V,E)$ the graph obtained from $\mathcal{G}_{(T,P)}(V,E)$, by decomposing its weights: $p_i = 2q_i + r_i$, with $q_i \geq 0$ and $r_i > 0$, and $q_i = t_i$, where $Q = \sum q_i$ and $R = \sum r_i$ (see Figure 3). The graph $\mathcal{G}'_{(Q,R)}(V,E)$ will be called *auxiliary graph* and it is denoted by $\mathcal{G}^{Aux}_{B}$.

We introduce the following notation for the graphs of $\mathcal{G}^{B}_{(T,P)}$: If $T = Q$ and $R = P$, we denote by $\theta_V = V^+ - V^-$, $\theta_W = (T^+ - T^-) + (P^+ - P^-)$. For a graph of $\mathcal{G}^{Aux}_{B}$, we denote by $\theta_{W'} = (Q^+ - Q^-) + (R^+ - R^-)$.

**Definition 7.** For the graph $\mathcal{G}_{(T,P)}(V,E)$:

1. We say that it is **balanced**, if there exists some graph $\mathcal{G}^{b}_{(T,P)}$ in $\mathcal{G}^{B}_{(T,P)}$, satisfying the balancing condition:
$$D = \mid \theta_V - \theta_W \mid = 0. \tag{1}$$
2. If $\mid D \mid > 0$ for all $\mathcal{G}^{b}_{(T,P)}$ in $\mathcal{G}^{B}_{(T,P)}$ and there exists some graph $\mathcal{G}'^{b}_{(Q,R)}$ in $\mathcal{G}^{Aux}_{B}$ that satisfies the following the balancing condition:
$$D' = \mid \theta_V - \theta_{W'} \mid = 0 \tag{2}$$
then we say that $\mathcal{G}'_{(Q,R)}(V,E)$ is a balanced graph and $\mathcal{G}_{(T,P)}(V,E)$ is a **quasi-balanced** graph.

Let us remark that, inverting the labels assigned to the vertices, the balancing expression has it is signal also inverted. However, the balancing criterion is evaluated over the absolute value of the balancing expression. Therefore, once the graph is bipartite, the signs $\pm$ can be assigned to the graph starting from any vertex and keeping vertices at the end of each edge with opposite signs.

A remarkable feature of quasi-balanced graphs is that the sum of two quasi-balanced graphs is a quasi-balanced graph.

Figure 3 illustrates the three sets of auxiliary bipartite subgraphs, except loops , of the graph $\mathcal{G}_{(7,9)}(8,15)$, containing all the possible subgraphs of $\mathcal{G}^{Aux}_{(7,9)}(8,15)$ associated to $\mathcal{G}_{(7,9)}(8,15)$, where the distinguished vertices have the weights $(0,p_i)$ decomposed. We leave to the reader, verify that the graph $\mathcal{G}_{(7,9)}(8,15)$ is not quasi-balanced, because its auxiliary subgraphs are not balanced.

## 3.2 | Pre-balanced graphs

**Definition 8.** A graph type tree with a unique edge (two vertices) it is said to be *irreducible*.

**Definition 9.** A **pre-balanced graph** $\mathcal{G}_{(T,P)}(V,E)$ is a graph that can be obtained as a horizontal sum of some quasi-balanced graph $\mathcal{G}''_{(Q',R')}(V,E)$ with a set suitable of balanced irreducible trees. The graph $\mathcal{G}''_{(Q',R')}(V,E)$ it will be called *base graph* of $\mathcal{G}_{(T,P)}(V,E)$.

*Remark 3.* All quasi-balanced subgraphs are the base graph of $\mathcal{G}_{(T,P)}(V,E)$, considering that all irreducible trees added to the base graph have zero weight.

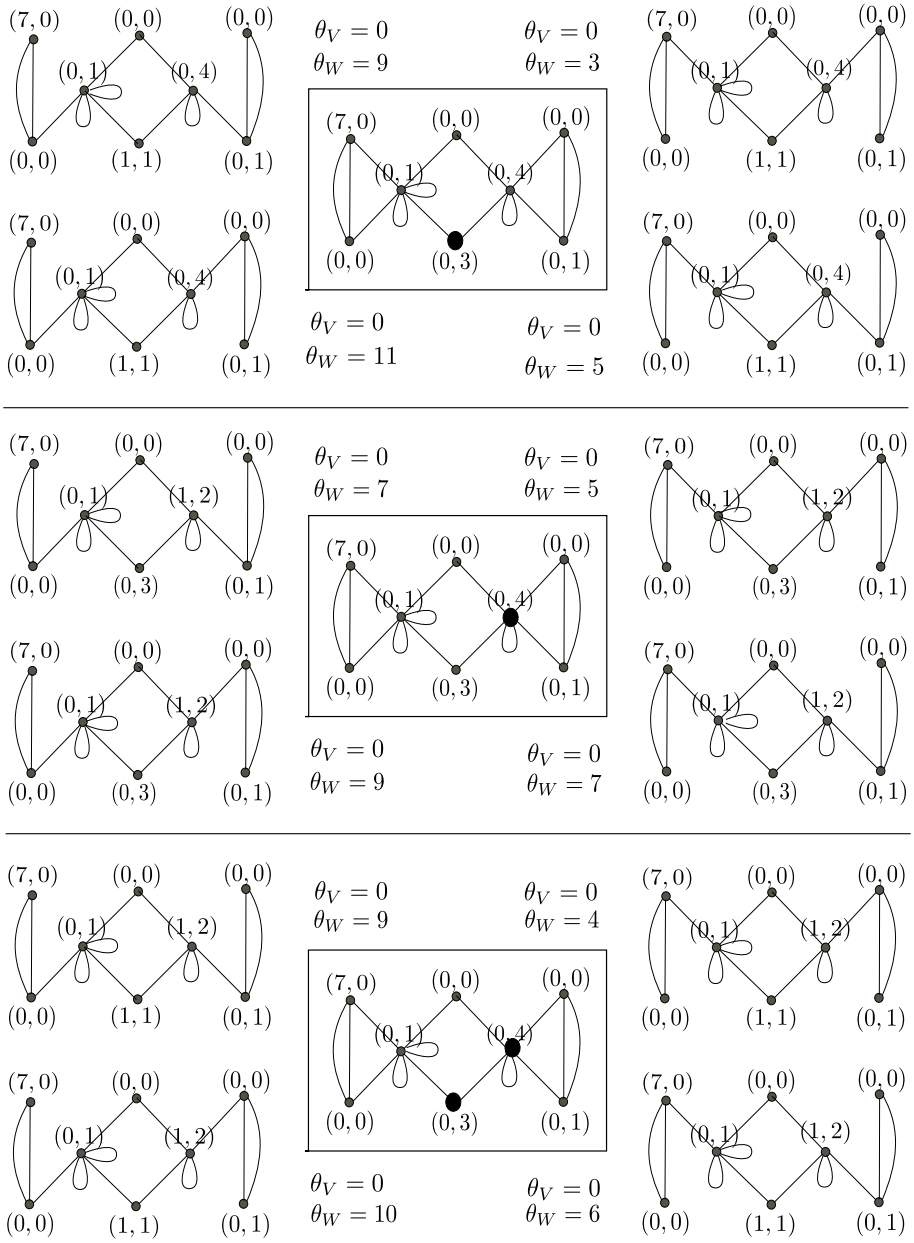


**FIGURE 3** Examples of auxiliary graph and bipartite subgraphs.

Given a base graph $\mathcal{G}''_{(Q',R')}(V,E)$ of $\mathcal{G}_{(T,P)}(V,E)$, a maximal bipartite subgraph of $\mathcal{G}''_{(Q',R')}(V,E)$ it will be denoted by $\mathcal{G}''^{b}_{(Q',R')}$.

We denote by $Q'^{+}$ (resp $Q'^{-}$) the sum of all the $q'_i$ weights corresponding to vertices with positive label (resp. with negative label), by $R'^{+}$ (resp. $R'^{-}$) the sum of all the $r'_i$ weights corresponding to vertices with positive label (resp. with negative label). We denote:

$$\theta_{W''} = (Q'^{+} - Q'^{-}) + (R'^{+} - R'^{-});\ D'' = \mid \theta_V - \theta_{W''} \mid \tag{3}$$

Let us denote by $\mathcal{B}^{\mathcal{G}}_{(T,P)}(V,E)$, the set of all graphs $\mathcal{G}''_{(Q',R')}(V,E)$ satisfying the next condition: exists some graph $\mathcal{G}'_{(Q,R)}(V,E)$ in $\mathcal{G}^{Aux}_{(T,P)}(V,E)$, that can be obtained as horizontal sums among a base graph $\mathcal{G}''_{(Q',R')}(V,E)$ and a set of irreducible balanced trees. Then $\mathcal{G}'_{(Q,R)}(V,E)$ and $\mathcal{G}''_{(Q',R')}(V,E)$ are isomorphic graphs and if the balanced irreducible trees have zero weight, then $\mathcal{G}'_{(Q,R)}(V,E)$ and $\mathcal{G}''_{(Q',R')}(V,E)$ are equal. Consequently $\mathcal{G}^{Aux}_{(T,P)}(V,E)$ is a subset of $\mathcal{B}^{\mathcal{G}}_{(T,P)}(V,E)$.

A remarkable feature of pre-balanced graphs is that the sum of two balanced graphs is a pre-balanced graph.

If $D'' = 0$ *then* $\mathcal{G}''_{(Q',R')}(V,E)$ *is a balanced graph and* $\mathcal{G}'_{(Q,R)}(V,E)$ *pre-balanced graph, hence,* $\mathcal{G}_{(T,P)}(V,E)$ *is a pre-balanced graph.*

Figure 4 illustrates a graph $\mathcal{G}$ with two odd cycles and three loops and shows two sequences that can be used to verify if $\mathcal{G}$ is contained in $\mathcal{B}^{\mathcal{G}}$. Figure 4(a),(b), the graph $\mathcal{G}$ is pre-balanced because it contains as base subgraph quasi-balanced $\mathcal{G}''$, which contains the bipartite subgraph $\mathcal{G}''^{b}$ balanced . Figure 4(c),(d) (e), the graph $\mathcal{G}'$, is obtained by the decomposition of the weights of $\mathcal{G}$, it is unbalanced, but contains a base subgraph $\mathcal{G}''$ balanced, which also contain $\mathcal{G}''^{b}$. Note that in the last sequence we assume $\mathcal{G}' = \mathcal{G}$.

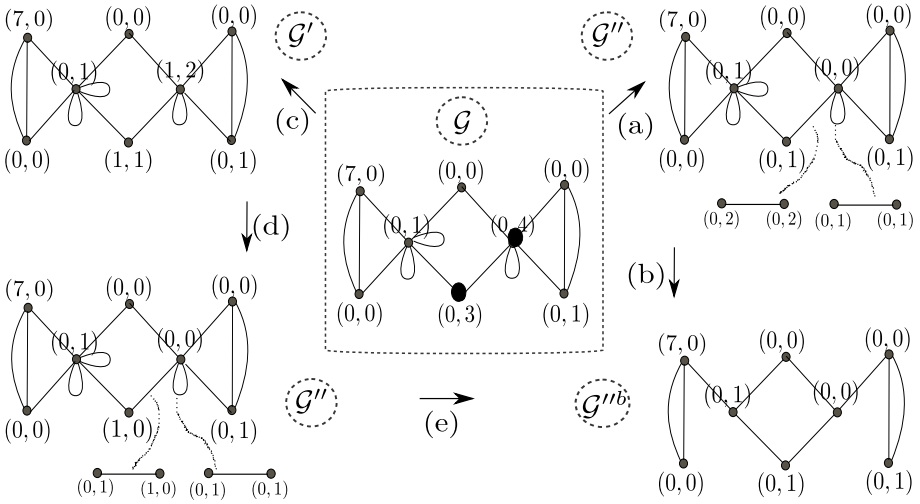


**FIGURE 4** Examples of checking graphs.

Now, the question is, *which graphs with weight pairs $\mathcal{G}_{(T,P)}(V, E)$ can be associated to a fold map with the apparent planar contour of a closed surface $M$ on the projective plane?* The following result has been proved in [8]:

**Theorem 1.** [8] A graph $\mathcal{G}$, with weights type $(t, 0)$ and $(0, p)$ on its vertices, is associated to a fold map with apparent planar contour from a closed surface to the projective plane if and only if the $\mathcal{G}$ is pre-balanced.

Theorem 1 claims that all pre-balanced graph can be associated with some fold map from a closed surface to the projective plane.

## 4 | ALGORITHM FOR DETERMINING FOLD MAPS (DFMG)

In this section, we develop the algorithm that leads us to verify if an input graph associated with a fold map is realisable on the projective plane. It can be translated as a problem of linear combinatorial programming with restrictions as we will see at the next section.

**Definition 10.** A **maximal tree** of a connected graph $\mathcal{G}_{(T,P)}(V, E)$ is the subgraph $\mathcal{A}_{(T,P)}(V, V-1)$ obtained by removing $1 - V + E$ edges of $\mathcal{G}_{(T,P)}(V, E)$.

**Proposition 1.** A *maximal tree* of a bipartite connected graph is a balanced tree if and only if the graph is balanced.

*Proof.* By Definition 10, $\mathcal{A}_{(T,P)}(V, V-1)$ is connected. The set of vertices of the graph $\mathcal{G}_{(T,P)}(V, E)$, with their weights, are the same as the set of the vertices of any maximal tree $\mathcal{A}_{(T,P)}(V, V-1)$. Hence, any maximal trees of a bipartite graph have the same colour as the vertices, which proves the proposition. □

Our problem is to study if the fold map with apparent contour on the projective is realisable. This is equivalent to check if the associated graph $\mathcal{G}_{(T,P)}(V, E)$, is pre-balanced (see Theorem 1), searching a maximal tree that satisfies the equation (3) regarding a colouring using some decomposition of the weights $(0, p_i)$ or removing the equal pairs of weights in consecutive pairs of vertices and weights on vertices with loops.

For simplicity of notation, we write $\mathcal{G}_{(T,P)}$ instead of a $\mathcal{G}_{(T,P)}(V, E)$. The algorithm proposed by the authors verifies if an input graph $\mathcal{G}_{(T,P)}$ type $(t, 0)$ and $(0, p)$ is balanced or not it is schematised in Figure 5. If the conclusion of the test is negative, then the program searches quasi-balanced graphs. If it is not pre-balanced, then we can conclude that it is not realisable by fold map of some closed surface on the projective plane. We will consider equations (1,2,3).

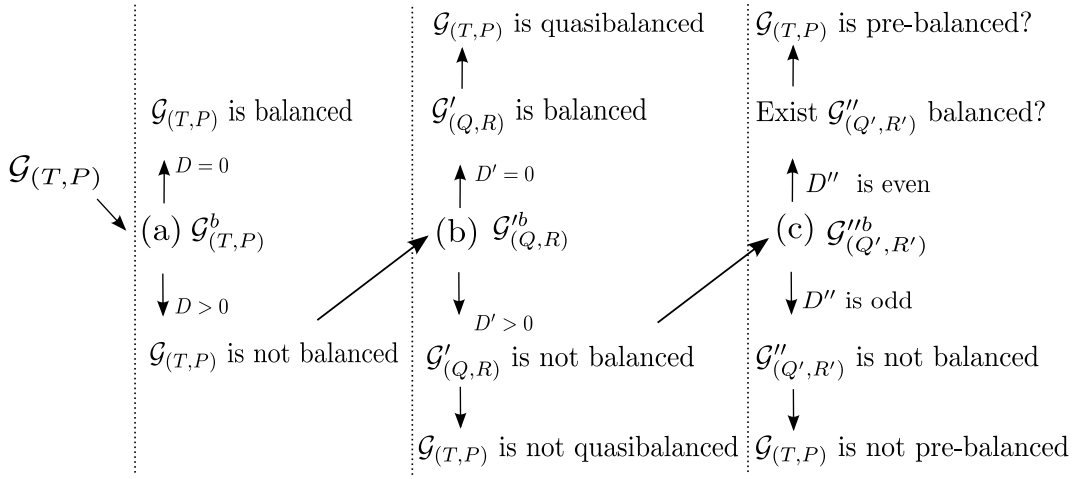


**FIGURE 5** Diagram to verify balanced graphs.

Briefly, the steps to check a given graph $\mathcal{G}_{(T,P)}$ are:

1. To verify if $\mathcal{G}_{(T,P)}$ is balanced, we need to check all their bipartite subgraphs.
   (a) If there exists any bipartite subgraph of $\mathcal{G}_{(T,P)}$ such that $D = 0$, then $\mathcal{G}_{(T,P)}$ is balanced.
   (b) If all of bipartite subgraphs of $\mathcal{G}_{(T,P)}$ produce $D > 0$, then $\mathcal{G}_{(T,P)}$ is not balanced.
2. To verify if $\mathcal{G}_{(T,P)}$ is quasi-balanced: firstly we decompose the weights as $p_i = 2q_i + r_i$, with $q_i \geq 0$ and $r_i > 0$, if $p_i > 0$, to obtain all the possible graphs $\mathcal{G}'_{(Q,R)}$.
   (a) If any of the graphs $\mathcal{G}'_{(Q,R)}$ is balanced, i.e, there exists any bipartite graph with $D' = 0$, then $\mathcal{G}_{(T,P)}$ is quasi-balanced.
   (b) If all $\mathcal{G}'_{(Q,R)}$ are not balanced, i.e., $D' > 0$ for all bipartite subgraphs of $\mathcal{G}'_{(Q,R)}$, then $\mathcal{G}_{(T,P)}$ is not quasi-balanced.
3. To check if $\mathcal{G}_{(T,P)}$ is pre-balanced, we proceed as follows: if $D'$ is odd then $\mathcal{G}_{(T,P)}$ is not pre-balanced.
4. If $D'$ is even, we have to find all possible graphs $\mathcal{G}''_{(Q',R')}$ such that $\mathcal{G}'_{(Q,R)}$ can be obtained as the connected sum of $\mathcal{G}''_{(Q',R')}$ with balanced irreducible trees.
   (a) If any of the graphs $\mathcal{G}''_{(Q',R')}$ is balanced, i.e, there is some bipartite subgraph with $D'' = 0$, then it is pre-balanced.
   (b) If all graphs $\mathcal{G}''_{(Q',R')}$ are not balanced, i.e, $D'' > 0$ for all bipartite graphs $\mathcal{G}''_{(Q',R')}$, then $\mathcal{G}_{(T,P)}$ is not pre-balanced.

The implementation of the algorithm DFMG proposed in this paper covers the first three steps. The computational verification for the last step is still an open problem.

Figure 6 shows a graphical example of how the algorithm works with the graph $\mathcal{G}_{(7,9)}(8, 15)$. The result of the test is that the aforementioned graph is quasi-balanced and consequently is pre-balanced.

## 4.1 | Searching fold maps as a linear programming problem

The DFMG program receives as input the original graph $\mathcal{G}_{(T,P)}$, and determine if there exists balanced maximal tree of one of the graphs $\mathcal{G}_{(T,P)}$ or $\mathcal{G}'_{(Q,R)}$. If there exists at least one balanced maximal tree, the program must give as output the following information:

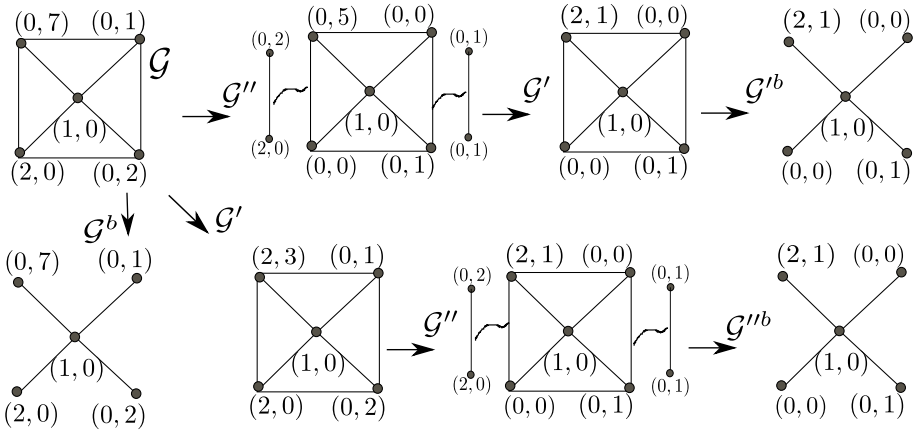


**FIGURE 6** Examples of pre-balanced graph.

- the colour of each vertex.
- the edges in the maximal balanced tree.
- the decomposition of the non-oriented part of each vertex.
- If the graph is balanced, quasi-balanced, or it is impossible to balance it.

DFMG code uses two integer programs to verify if a given graph is balanced or quasi-balanced.

Suppose a binary variable $x_i \in \{0, 1\}$ that is 1 (one) whenever the colour of the vertex $i$ is $+$, and 0 (zero) whenever the colour of the vertex $i$ is $-$. Thus, we can compute the components of the balancing equation (1), for the input graph as follows:

$$\begin{aligned} V^+ &= \sum_{i=1}^{n} x_i; \quad V^- = \|V\| - V^+; \\ T^+ &= \sum_{i=1}^{n} x_i t_i; \; T^- = \sum_{i=1}^{n} t_i - T^+; \\ P^+ &= \sum_{i=1}^{n} x_i p_i; \; P^- = \sum_{i=1}^{n} p_i - P^+. \end{aligned} \tag{4}$$

In Equation 4, $x_i$ is variable and $t_i$ e $p_i$ are inputs values for the oriented and non-oriented components of the vertex $i$ and $n = \|V\|$.

It is not always possible to get a solution that satisfies $D = 0$. The program tries to minimise the absolute value of the equation (1). Replacing the variables of equation (4) in (1), gives:

$$D = \left| 2 \sum_{i=1}^{n} (1 - t_i - p_i) x_i + \sum_{i=1}^{n} (t_i + p_i - 1) \right| \tag{5}$$

The graph is balanced if $D = 0$, and this is equivalent to say that the equation (5) holds when there exists a minimum. The equation (5) must be minimised under some constraints. The resulting graph must be a tree. Is equivalent to state that the resulting graph has the number of edges equal to $\|V\| - 1$ and there are no cycles (or equivalent to say that the graph must be connected). To model these constraints, we introduce new variables to represent which edges are in the resulting graph. The binary variables $e_{i,j} \in \{0, 1\}$ with $i, j \in E : i \in V, j \in V$ is 1 (one) if $e_{ij}$ is in the resulting graph and 0 (zero) otherwise.

These two constraints, number of edges and sub-cycle elimination, can be written as follows:

$$\sum e_{i,j} = \|V\| - 1, \quad i, j \in E \tag{6}$$

$$\sum_{i,j \in E : i \in S, j \in S} e_{ij} \leq |S| - 1, \quad \forall S \subseteq V \tag{7}$$

The equation (7) states that there is no subset $| S |$ of vertices connected by more than $|S| - 1$ edges. Therefore, it is enough to eliminate the sub-cycles of feasible solutions.

| $x_i$ | $x_j$ | $e_{i,j}$ | Eq. 8 | Eq. 9 | **Eq. 8 and Eq. 9** |
|---|---|---|---|---|---|
| 0 | 0 | 0 | $0 \leq 2$ | $1 \leq 1$ | true |
| 0 | 1 | 0 | $0 \leq 1$ | $0 \leq 1$ | true |
| 1 | 0 | 0 | $1 \leq 2$ | $1 \leq 2$ | true |
| 1 | 1 | 0 | $1 \leq 1$ | $0 \leq 2$ | true |
| 0 | 0 | 1 | $0 \leq 1$ | $1 \leq 0$ | **false** |
| 0 | 1 | 1 | $0 \leq 0$ | $0 \leq 0$ | true |
| 1 | 0 | 1 | $1 \leq 1$ | $0 \leq 1$ | true |
| 1 | 1 | 1 | $1 \leq 0$ | $0 \leq 1$ | **false** |

**TABLE 1** All possible values for colouring inequalities.

Another constraint is about the colouring of the vertices, once the resulting graph must be bipartite. As the variable $x_i$ represents the colour of the vertex $i$, we need to add constraints to guarantee that two neighbour vertices have opposite colours. As the resulting graph is a tree, it is always possible to assign different colours to neighbour vertices. Note that, this constraint must be satisfied just when the edge exists on the resulting graph, i. e., when the vertices $i$ and $j$ are not connected ($e_{ij} = 0$) this constraint must to be ignored.

Given two neighbours with vertices $i$ and $j$ on the input graph, this constraint can be implemented as follows:

$$x_i \leq (1 - x_j) + (1 - e_{ij}), \quad j \in \delta(i) \tag{8}$$

$$(1 - x_j) \leq x_i + (1 - e_{ij}), \quad j \in \delta(i) \tag{9}$$

where $\delta(i)$ denotes the set of vertices neighbours to the vertex $i$.

Note that if the edge $e_{ij} = 0$ (the edge does not exist), the constraints are written as follows:

$$x_i \leq (1 - x_j) + 1, \quad j \in \delta(i) \tag{10}$$

$$(1 - x_j) \leq x_i + 1, \quad j \in \delta(i) \tag{11}$$

that are true for every possible values of $x_i$ and $x_j$, as shows in the Table 1.

In contrast, if the edge $e_{i,j}$ exists, i.e. $e_{i,j} = 1$, the constraints are written as follows:

$$x_i \leq (1 - x_j), \quad j \in \delta(i) \tag{12}$$

$$(1 - x_j) \leq x_i, \quad j \in \delta(i) \tag{13}$$

that are true just when $x_i$ e $x_j$ has opposite values, as show in the Table 1.

Consequently, the constraints given by equations (8) and (9) are enough to guarantee the correct colouring of the graph.

The following integer program determines if a graph is balanced or not:

$$\begin{aligned}
\text{minimise } & \left| 2 \sum_{i=1}^{n} (1 - t_i - p_i) x_i + \sum_{i=1}^{n} (t_i + p_i - 1) \right| \\
\text{subject to } & \sum e_{ij} = \|V\| - 1, && 1 \leq i, j \leq n \\
& x_i \leq (1 - x_j) + (1 - e_{ij}), && j \in \delta(i) \\
& (1 - x_j) \leq x_i + (1 - e_{ij}), && j \in \delta(i) \\
& \sum_{ij \in E : i \in S, j \in S} e_{i,j} \leq \mid S \mid -1, && \forall S \subseteq V \\
& x_i, e_{ij} \in \{0, 1\}.
\end{aligned}$$

In order to remove the module from formulation, we can replace the objective function by a variable and add two new constraints [13], as follows:

$$
\begin{aligned}
&\text{minimise } f \\
&\text{subject to } \sum e_{ij} = \|V\| - 1, && 1 \le i, j \le n \\
&\qquad x_i \le (1 - x_j) + (1 - e_{ij}), && j \in \delta(i) \\
&\qquad (1 - x_j) \le x_i + (1 - e_{ij}), && j \in \delta(i) \\
&\qquad \sum_{ij \in E : i \in S, j \in S} e_{ij} \le |\, S \,| - 1, && \forall S \subseteq V \\
&\qquad 2 \sum_{i=1}^{n} (1 - t_i - p_i) x_i + \sum_{i=1}^{n} (t_i + p_i - 1) \le f, \\
&\qquad - \left[ 2 \sum_{i=1}^{n} (1 - t_i - p_i) x_i + \sum_{i=1}^{n} (t_i + p_i - 1) \right] \le f, \\
&\qquad x_i, e_{ij} \in \{0, 1\}.
\end{aligned}
$$

## 4.2 | Quasi-balanced Graph

When the initial graph is not a balanced graph, we need to test if the graph is quasi-balanced. In this case, it is allowed to decompose the non-oriented part of the each vertex such that $p_i = 2q_i + r_i$, with $q_i \ge 0$ and $r_i > 0$.

To verify if a graph is quasi-balanced we need to replace the part of the objective function related with $P$ by $Q$ and $R$ by introducing new variables $q_i$ and $r_i$ to represent the decomposition. Thus, the new balancing equation is $D'$ (see equation 2 for more details):

The new variables can be expressed in terms of the binary variable $x_i$ as follow:

$$
\begin{aligned}
Q^+ = \sum_{i=1}^{n} x_i q_i, \; Q^- = \sum_{i=1}^{n} q_i - Q^+, \\
R^+ = \sum_{i=1}^{n} x_i r_i, \; R^- = \sum_{i=1}^{n} r_i - R^+.
\end{aligned}
\tag{14}
$$

Replacing the equations of 14 in 2 we get the new objective function:

$$
D = \left| 2 \sum_{i=1}^{n} (1 - t_i - q_i - r_i) x_i + \sum_{i=1}^{n} (t_i + q_i + r_i - 1) \right| \tag{15}
$$

Equation (15) has two quadratic terms per vertices: $x_i q_i$ and $x_i r_i$.

In order to guarantee a correct decomposition of the non-oriented part, we need to add a new constraint by each vertex whenever $p_i \ge 0$:

$$
2q_i + r_i = p_i \quad 1 \le i \le n \tag{16}
$$

Moreover, if it is impossible to find a decomposition such that the graph is quasi-balanced, the graph must be tested if it is pre-balanced. However, if there is no solution that $D$ is even, the graph is also not pre-balanced. In order to know if the solution exists, we can add new constraints to eliminate the solutions that $D$ is odd of the feasible solutions. Suppose that $f$ is the value of the objective function. To guarantee that $f$ is odd, it is enough to add a new variable $y$ and a new constraint:

$$
f - 2y = 0 \tag{17}
$$

With these definitions, we can write a quadratic integer program to verify if the graph is quasi-balanced:

$$
\begin{aligned}
\text{minimise} \quad & \left| 2 \sum_{i=1}^{n} (1 - t_i - q_i - r_i) x_i + \sum_{i=1}^{n} (t_i + q_i + r_i - 1) \right| \\
\text{subject to} \quad & \sum e_{ij} = \|V\| - 1, && 1 \leq i, j \leq n \\
& 2q_i + r_i = p_i, && 1 \leq i \leq n \\
& \left| 2 \sum_{i=1}^{n} (1 - t_i - q_i - r_i) x_i + \sum_{i=1}^{n} (t_i + q_i + r_i - 1) \right| - 2y = 0 \\
& x_i \leq (1 - x_j) + (1 - e_{ij}), && x_j \in \delta(x_i) \\
& (1 - x_j) \leq x_i + (1 - e_{ij}), && x_j \in \delta(x_i) \\
& \sum_{ij \in E : i \in S, j \in S} e_{i,j} \leq \mid S \mid -1, && \forall S \subseteq V \\
& r_i > 0, && 1 \leq i \leq n; \text{ if } q_i > 0 \\
& x_i, e_{ij} \in \{0, 1\}, \\
& q_i, r_i \in \mathbb{N}, \\
& y \in \mathbb{R},
\end{aligned}
$$

In general, it is more expensive to solve a quadratic problem than a linear problem. However, it is possible to linearise this formulation. In the terms $x_i q_i$, $x_i r_i$, we want the result $q_i$ and $r_i$ if $x_i = 1$ and 0 otherwise for both terms. Thus, from the decomposition equation, we know that $0 \leq q_i \leq p_i/2$ e $0 \leq r_i \leq p_i$.

Suppose the term $xy$ where $x$ is a binary variable and $y \in [L, U]$. Thus, to linearise this product, we can add a new variable $z = xy$ and four new constraints.

$$
z \leq Ux; \qquad z \geq Lx; \qquad z \leq y - (1 - x)L; \qquad z \geq y - (1 - x)U. \tag{18}
$$

Applying this rule to the terms $x_i q_i$ and $x_i r_i$, we can add new variables $w_i = x_i q_i$ e $z_i = x_i r_i$. Now, we can rewrite the quadratic formulation as a linear formulation using the known bounds for the variables $q_i$ and $r_i$.

$$
\begin{aligned}
\text{minimise} \quad & \left| 2 \sum_{i=1}^{n} (1 - t_i - q_i - r_i) x_i + \sum_{i=1}^{n} (t_i + q_i + r_i - 1) \right| \\
\text{subject to} \quad & \sum e_{ij} = \|V\| - 1, && 1 \le i, j \le n \\
& 2q_i + r_i = p_i, && 1 \le i \le n \\
& \left| 2 \sum_{i=1}^{n} (1 - t_i - q_i - r_i) x_i + \sum_{i=1}^{n} (t_i + q_i + r_i - 1) \right| - 2y = 0, \\
& x_i \le (1 - x_j) + (1 - e_{ij}), && x_j \in \delta(x_i) \\
& (1 - x_j) \le x_i + (1 - e_{ij}), && x_j \in \delta(x_i) \\
& \sum_{ij \in E : i \in S, j \in S} e_{i,j} \le \mid S \mid -1, && \forall\, S \subseteq V \\
& r_i > 0, && 1 \le i \le n; \text{ if } q_i > 0 \\
& w_i \ge 0, && 1 \le i \le n \\
& 2w_i \le x_i p_i, && 1 \le i \le n \\
& w_i \le q_i, && 1 \le i \le n \\
& w_i \ge q_i - (1 - x_i) p_i, && 1 \le i \le n \\
& z_i \ge 0, && 1 \le i \le n \\
& z_i \le x_i p_i, && 1 \le i \le n \\
& z_i \le r_i, && 1 \le i \le n \\
& z_i \ge r_i - (1 - x_i) p_i, && 1 \le i \le n \\
& x_i, e_{ij} \in \{0, 1\}, \\
& q_i, e_i \in \mathbb{N}, \\
& y, w_i, z_i \in \mathbb{R},
\end{aligned}
$$

The final formulation, without the module, is:

$$
\begin{aligned}
&\text{minimise } f \\
&\text{subject to } \sum e_{ij} = \|V\| - 1, && 1 \le i, j \le n \\
&\quad 2q_i + r_i = p_i, && 1 \le i \le n \\
&\quad f - 2y = 0, \\
&\quad x_i \le (1 - x_j) + (1 - e_{ij}), && x_j \in \delta(x_i) \\
&\quad (1 - x_j) \le x_i + (1 - e_{ij}), && x_j \in \delta(x_i) \\
&\quad \sum_{ij \in E : i \in S, j \in S} e_{i,j} \le |\, S - 1, && \forall\, S \subseteq V \\
&\quad r_i > 0, && 1 \le i \le n;\ \text{if } q_i > 0 \\
&\quad 2\sum_{i=1}^{n}(1 - t_i - q_i - r_i)x_i + \sum_{i=1}^{n}(t_i + q_i + r_i - 1) \le f, \\
&\quad -[2\sum_{i=1}^{n}(1 - t_i - q_i - r_i)x_i + \sum_{i=1}^{n}(t_i + q_i + r_i - 1)] \le f, \\
&\quad w_i \ge 0, && 1 \le i \le n \\
&\quad 2w_i \le x_i p_i, && 1 \le i \le n \\
&\quad w_i \le q_i, && 1 \le i \le n \\
&\quad w_i \ge q_i - (1 - x_i)p_i, && 1 \le i \le n \\
&\quad z_i \ge 0, && 1 \le i \le n \\
&\quad z_i \le x_i p_i, && 1 \le i \le n \\
&\quad z_i \le r_i, && 1 \le i \le n \\
&\quad z_i \ge r_i - (1 - x_i)p_i, && 1 \le i \le n \\
&\quad x_i, e_{ij} \in \{0, 1\}, \\
&\quad q_i, r_i \in \mathbb{N}, \\
&\quad y, w_i, z_i \in \mathbb{R},
\end{aligned}
$$

## 5 | RESULTS

We use a data set of 20 complete graphs with the number of vertices varying from 24 to 70 to test the proposed optimisation programs. All optimisation problems were implemented in C++ language and solved using Gurobi Optimizer[10].

To evaluate the efficiency of the proposed optimisation programs in current computers, we used, for all tests, a desktop computer with an Intel Core i7-6700 CPU@ 3.40GHz processor and 16 GB of DDR3 memory. To avoid memory cache fluctuations, each reported time includes running the optimisations five times and averaging the results.

As the vertex weights are small, fewer weights can be decomposed to balance the graph, such that this task becomes challenging. Because of this, each vertex weight has values less than equal to 5, except for the first vertex that can weight from 0 to 20. The difference from the first to the others causes an imbalance of weights in the graph, making it even more challenging to balance.

Table 2 presents the results for the test if a graph is balanced or quasi-balanced. As expected, it is more difficult to obtain a balanced graph than a pre-balanced graph, because in the case of a pre-balanced graph, the optimisation program has more degrees of freedom to minimise the balancing equation by decomposing vertex weights.

For the same reason, it is expected, in most of the cases, to take more time to check if a graph is balanced, as it is shown in Table 3.

| Graph Name | Number of vertices | Balanced | Quasi-balanced |
|---|---|---|---|
| g1.txt | 24 | not | yes |
| g2.txt | 27 | not | yes |
| g3.txt | 36 | not | yes |
| g4.txt | 46 | not | yes |
| g5.txt | 51 | not | yes |
| g6.txt | 60 | not | yes |
| g7.txt | 62 | not | yes |
| g8.txt | 62 | not | yes |
| g9.txt | 65 | yes | yes |
| g10.txt | 65 | not | not |
| g11.txt | 66 | yes | yes |
| g12.txt | 66 | not | yes |
| g13.txt | 67 | not | yes |
| g14.txt | 67 | yes | yes |
| g15.txt | 68 | yes | yes |
| g16.txt | 69 | not | not |
| g17.txt | 70 | not | not |
| g17.txt | 70 | yes | yes |
| g19.txt | 70 | yes | yes |
| g20.txt | 70 | not | not |

**TABLE 2** Result of execution algorithm to determine if the graphs are balanced or quasi-balanced.

Surprisingly, there is no dominance of the linear approach over the quadratic approach to determine if a graph is quasi-balanced or not. We believe that this occurs because, although the approach is linear, the optimisation program has more variables to be calculated, such that this approach consumes more time in some of the instances.

## 6 | ACKNOWLEDGEMENTS

This work has been supported by grant Generalitat Valenciana (GV/2010/066), PRECEU-UCH INDI 18/11 and ESI International Chair@CEU-UCH. The authors are grateful to an unknown referee for interesting suggestions to improve the readability of the paper. There are no conflicts of interest to this work.

## References

1. **D. Marr and E. Hildreth**. *Theory of edge detection*. Proc. Royal Soc. London, B, 207, 187-217 (1980). DOI: 10.1098/rspb.1980.0020

2. **J. Damon, P. Giblin , G. Haslinger.** Local features in natural images via singularity theory. Springer, 2016. DOI 10.1007/978-3-319-41471-3.

3. **Bellettini, G., Beorchia, V., Paolini, M., Pasquarelli, F.**. Shape Reconstruction from Apparent Contours. Springer, 2015. DOI: 10.1007/978-3-662

4. **M. Golubitsky** and **V. Guillemin**, *Stable Mappings and Their Singularities*, Springer Verlag, Berlin (1976). DOI: 10.1007/978-1-4615-7904-5.

| Graph Name | Number of vertices | Balanced Time(s) | Quadratic Quasi-balanced Time(s) | Linear Quasi-balanced Time(s) |
|---|---|---|---|---|
| g1.txt | 24 | 0.68 | 0.06 | 0.07 |
| g2.txt | 27 | 0.13 | 0.08 | 0.02 |
| g3.txt | 36 | 1.81 | 0.20 | 0.18 |
| g4.txt | 46 | 15.11 | 0.55 | 0.21 |
| g5.txt | 51 | 12.69 | 0.36 | 0.89 |
| g6.txt | 60 | 14.08 | 3.29 | 0.89 |
| g7.txt | 62 | 46.51 | 5.54 | 2.81 |
| g8.txt | 62 | 8.06 | 1.40 | 0.51 |
| g9.txt | 65 | 9.61 | 0.75 | 1.06 |
| g10.txt | 65 | 15.26 | 1.32 | 0.59 |
| g11.txt | 66 | 13.38 | 2.57 | 2.48 |
| g12.txt | 66 | 10.53 | 0.91 | 3.50 |
| g13.txt | 67 | 7.02 | 2.13 | 2.43 |
| g14.txt | 67 | 2.01 | 9.70 | 0.96 |
| g15.txt | 68 | 18.83 | 1.41 | 1.50 |
| g16.txt | 69 | 2.12 | 2.29 | 10.84 |
| g17.txt | 69 | 15.46 | 3.93 | 1.70 |
| g18.txt | 70 | 1.61 | 2.37 | 2.65 |
| g19.txt | 70 | 21.97 | 1.20 | 0.59 |
| g20.txt | 70 | 2.53 | 1.85 | 1.21 |

**TABLE 3** Time table for the algorithms executions. Each reported time is an average of five executions.


5. **H. Whitney**, On Singularities of mappings of euclidean spaces. I. Mappings of the Plane into the Plane, *Annals of Mathematics.* 62: 374-410 (1955). DOI: 10.1007/978146122972827.

6. **D. Hacon, C. Mendes de Jesus** and **M. C. Romero Fuster**, Fold maps from the sphere to the plane, *Experimental Maths*, 15:491-497 (2006). DOI: 10.1080/10586458.2006.10128973.

7. **D. Hacon, C. Mendes de Jesus and M. C. Romero Fuster**, Stable maps from surfaces to the plane with prescribed branching data *Topology and Its Appl.* 154 (1): 166-175 (2007). DOI:10.1016/j.topol.2006.04.005.

8. **C. Mendes de Jesus and M. C. Romero Fuster**, Graphs of stable maps between closed surfaces, *RIMS Kôkyûroku Bessatsu*, B55: 147-159, 2016.

9. **C. Mendes de Jesus and M. C. Romero Fuster**, Graphs of stable maps from closed surfaces to the projective plane, *Topology and Its Appl.*, 2017. DOI: 10.1016/j.topol.2017.11.013

10. **Gurobi Optimization, LLC**, Gurobi Optimizer Reference Manual, http://www.gurobi.com, 2018.

11. **D. Hacon, C. Mendes de Jesus and M. C. Romero Fuster**, Topological invariants of stable maps from a surface to the plane from a global viewpoint. *Real and Complex Singularities*. Informa UK Limited (2003). DOI:10.1201/9780203912089.ch10

12. **C. Mendes de Jesus and M. C. Romero Fuster**, Graphs of fold maps from closed surfaces to the projective plane, *Topology and Its Appl.*, preprint 2019.

13. **D. Bertsimas and J. Tsitsiklis.** Introduction to linear optimization. Anthena Scientific, 1997.

14. **M. C. Golumbic**, *Algorithmic graph theory and perfect graphs*, Second edition, Elsevier, Amsterdam (2004).


□