HAL open science

NLC-2 graph recognition and isomorphism

Vincent Limouzy, Fabien de Montgolfier, Michaël Rao

To cite this version:

Vincent Limouzy, Fabien de Montgolfier, Michaël Rao. NLC-2 graph recognition and isomorphism. Graph-Theoretic Concepts in Computer Science 33rd International Workshop, WG 2007, Dornburg, Germany, June 21-23, 2007., Jun 2007, Dornburg, Germany. pp.86-98, 10.1007/978-3-540-74839-7_9 . hal-00134605

HAL Id: hal-00134605

https://hal.science/hal-00134605v1

Submitted on 2 Mar 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

NLC-2 graph recognition and isomorphism

Vincent Limouzy[1] Fabien de Montgolfier[1] Michaël Rao[1]

Abstract

NLC-width is a variant of clique-width with many application in graph algorithmic. This paper is devoted to graphs of NLC-width two. After giving new structural properties of the class, we propose a $O(n^2m)$-time algorithm, improving Johansson's algorithm [14]. Moreover, our alogrithm is simple to understand. The above properties and algorithm allow us to propose a robust $O(n^2m)$-time isomorphism algorithm for NLC-2 graphs. As far as we know, it is the first polynomial-time algorithm.

hal-00134605, version 1 - 2 Mar 2007

1 Introduction

NLC-width is a graph parameter introduced by Wanke [16]. This notion is tightly related to clique-width introduced by Courcelle *et al.* [2]. Both parameters were introduced to generalise the well known tree-width. The motivation on research about such *width* parameter is that, when the width (NLC-, clique- or tree-width) is bounded by a constant, then many NP-complete problems can be solved in polynomial (even linear) time, if the decomposition is provided.

Such parameters give insights on graph structural properties. Unfortunately, finding the minimum NLC-width of the graph was shown to be NP-hard by Gurski *et al.* [12]. Some results however are known. Let NLC-k be the class of graph of NLC width bounded by k. NLC-1 is exactly the class of cographs. Probe-cographs, bi-cographs and weak-bisplit graphs [9] belong to NLC-2. Johansson [14] proved that recognising NLC-2 graphs is polynomial and provided an $O(n^4 \log(n))$ recognition algorithm. Complexity for recognition of NLC-k, $k \geq 3$, is still unknown.

In this paper we improve Johansson's result down to $O(n^2m)$. Our approach relies on graph decompositions. We establish the tight links that exist between NLC-2 graphs and the so-called modular decomposition, split decomposition, and bi-join decomposition.

NLC-2 can be defined as a graph colouring problem. Unlike NLC-k classes, for $k \geq 3$, *recolouring* is useless for prime NLC-2 graphs. That allow us to propose a canonical decomposition of bi-coloured NLC-2 graphs, defined as certain bi-coloured split operations. This decomposition can be computed in $O(nm)$ time if the colouring is provided. If a graph is *prime*, there using split and bi-join decompositions, we show that there is at most $O(n)$ colourings to check. Finally, modular decomposition properties allow to reduce NLC-2 graph decomposition to prime NLC-2 graph decomposition. Section 3 explains this $O(n^2m)$-time decomposition algorithm.

In Section 4 is proposed an isomorphism algorithm. Using modular, split and bi-join decompositions and the canonical NLC-2 decomposition, isomorphism between two NLC-2 graphs can be tested in $O(n^2m)$ time.

2 Preliminaries

A graph $G = (V, E)$ is pair of a set of *vertices* V and a set of *edges* E. For a graph G, $V(G)$ denote its set of vertices, $E(G)$ its set of edges, $n(G) = |V(G)|$ and $m(G) = |E(G)|$ (or V, E, n and m if

[1]LIAFA, Université Paris 7. {limouzy,fm,rao}@liafa.jussieu.fr. Research supported by the French ANR project "Graph Decompositions and Algorithms (GRAAL)"

the graph is clear in the context). $N(x) = \{y \in V : \{x, y\} \in E\}$ denotes the *neighbourhood* of the vertex x, and $N[x] = N(v) \cup \{v\}$. For $W \subseteq V$, $G[W] = (W, E \cap W^2)$ denote the *graph induced by* W. Let A and B be two disjoint subsets of V. Then we note A ① B if for all $(a, b) \in A \times B$, then $\{a, b\} \in E$, and we note A ⓪ B if for all $(a, b) \in A \times B$, then $\{a, b\} \notin E$. Two graphs $G = (V, E)$ and $G' = (V', E')$ are *isomorphic* (noted $G \simeq G'$) if there is a bijection $\varphi : V \to V'$ such that $\{x, y\} \in E \Leftrightarrow \{\varphi(x), \varphi(y)\} \in E'$, for all $u, v \in V$.

A *k-labelling* (or *labelling*) is a function $l : V \to \{1, \ldots, k\}$. A *k-labelled graph* is a pair of a graph $G = (V, E)$ and a k-labelling l on V. It is denoted by (G, l) or by (V, E, l). Two labelled graphs (V, E, l) and (V', E', l') are isomorphic if there is a bijection $\varphi : V \to V'$ such that $\{u, v\} \in E \Leftrightarrow \{\varphi(x), \varphi(y)\} \in E'$ and $l(u) = l'(\varphi(u))$ for all $u, v \in V$.

NLC-k classes. Let k be a positive integer. The class of NLC-k graphs is defined recursively by the following operations.

- For all $i \in \{1, \ldots, k\}$, $\cdot(i)$ is in NLC-k, where $\cdot(i)$ is the graph with one vertex labelled i.
- Let $G_1 = (V_1, E_1, l_1)$ and $G_2 = (V_2, E_2, l_2)$ be NLC-k and let $S \subseteq \{1, \ldots, k\}^2$. Then $G_1 \times_S G_2$ is in NLC-k, where $G_1 \times_S G_2 = (V, E, l)$ with $V = V_1 \cup V_2$,

$$E = E_1 \cup E_2 \cup \{\{u, v\} : u \in V_1, v \in V_2, (l_1(u), l_2(v)) \in S\}$$

$$\text{and for all } u \in V,\ l(u) = \begin{cases} l_1(u) \text{ if } u \in V_1 \\ l_2(u) \text{ if } u \in V_2. \end{cases}$$

- Let $R : \{1, \ldots, k\} \to \{1, \ldots, k\}$ and $G = (V, E, l)$ be NLC-k. Then $\rho_R(G)$ is in NLC-k, where $\rho_R(G) = (V, E, l')$ such that $l'(u) = R(l(u))$ for all $u \in V$.

A graph is NLC-k if there is a k-labelling of G such that (G, l) is in NLC-k. A k-labelled graph is *NLC-k ρ-free* if it can be constructed without the ρ_R operation.

Modules and modular decomposition. A *module* in a graph is a non-empty subset $X \subseteq V$ such that for all $u \in V \setminus X$, then either $N(u) \cap X = \emptyset$ or $X \subseteq N(u)$. A module is *trivial* if $|X| \in \{1, |V|\}$. A graph is *prime* (w.r.t. modular decomposition) if all its modules are trivial. Two sets X and X' *overlap* if $X \cap X'$,$X \setminus X'$ and $X' \setminus X$ are non-empty. A module X is *strong* if there is no module X' such that X and X' overlap. Let $\mathcal{M}'(G)$ be the set of modules, let $\mathcal{M}(G)$ be the set of strong modules of G, and let $\mathcal{P}(G) = \{M_1, \ldots, M_k\}$ be the maximal (w.r.t. inclusion) members of $\mathcal{M}(G) \setminus \{V\}$.

Theorem 1. *[11] Let $G = (V, E)$ be a graph such that $|V| \geq 2$. Then:*

- *if G is not connected, then $\mathcal{P}(G)$ is the set of connected components of G,*
- *if $\overline{G}$ is not connected, then $\mathcal{P}(G)$ is the set of connected components of $\overline{G}$,*
- *if G and $\overline{G}$ are connected, then $\mathcal{P}(G)$ is a partition of V and is formed with the maximal members of $\mathcal{M}' \setminus \{V\}$.*

In all cases, $\mathcal{P}(G)$ is a partition of V, and G can be decomposed into $G[M_1], \ldots, G[M_k]$. The *characteristic graph* G^* of a graph G is the graph of vertex set $\mathcal{P}(G)$ and two $P, P' \in \mathcal{P}(G)$ are adjacent if there is an edge between P and P' in G (and so there is no non-edges since P and P' are two modules). The recursive decomposition of a graph by this operation gives the *modular decomposition* of the graph, and can be represented by a rooted tree, called the

modular decomposition tree. It can be computed in linear time [15]. The nodes of the modular decomposition tree are exactly the strong modules, so in the following we make no distinction between the modular decomposition of G and $\mathcal{M}(G)$. Note that $|\mathcal{M}(G)| \leq 2 \times n - 1$. For $M \in \mathcal{M}(G)$, let $G_M = G[M]$ and G^*_M its characteristic graph.

Lemma 2. *[14] Let G be a graph. G is NLC-k if and only if every characteristic graph in the modular decomposition of G is NLC-k.*

Moreover, a NLC-k expression for G can be easily constructed from the modular decomposition and from NLC-k expressions of prime graphs. On prime graphs, NLC-2 recognition is easier:

Lemma 3. *[14] Let G be a prime graph. Then G is NLC-2 if and only if there is a 2-labelling l such that (G, l) is NLC-2 ρ-free.*

Bi-partitive family. A *bipartition* of V is a pair $\{X, Y\}$ such that $X \cap Y = \emptyset$, $X \cup Y = V$ and X and Y are both non empty. Two bipartitions $\{X, Y\}$ and $\{X', Y'\}$ *overlap* if $X \cap Y$, $X \cap Y'$, $X' \cap Y$ and $X' \cap Y'$ are non empty. A family $\mathcal{F}$ of bipartitions of V is *bipartitive* if (1) for all $v \in V$, $\{\{v\}, V \setminus \{v\}\} \in \mathcal{F}$ and (2) for all $\{X, Y\}$ and $\{X', Y'\}$ in $\mathcal{F}$ such that $\{X, Y\}$ and $\{X', Y'\}$ overlap, then $\{X \cap X', Y \cup Y'\}$, $\{X \cap Y', Y \cup X'\}$, $\{Y \cap X', X \cup Y'\}$, $\{Y \cap Y', X \cup X'\}$ and $\{X \Delta X', X \Delta Y'\}$ are in $\mathcal{F}$ (where $X \Delta Y = (X \setminus Y) \cup (Y \setminus X)$). Bipartitive families are very close to partitive families [1], which generalise properties of modules in a graph.

A member $\{X, Y\}$ of a bipartitive family $\mathcal{F}$ is *strong* if there is no $\{X', Y'\}$ such that $\{X, Y\}$ and $\{X', Y'\}$ overlap. Let T be a tree. For an edge e in the tree, $\{C^1_e, C^2_e\}$ denote the bipartition of leaves of T such that two leaves are in the same set if and only if the path between them avoids e. Similarly, for an internal node α, $\{C^1_\alpha, \ldots, C^{d(\alpha)}_\alpha\}$ denote the partition of leaves of T such that two leaves are in the same set if and only if the path between them avoid α.

Theorem 4. *[3] Let $\mathcal{F}$ be a bipartitive family on V. Then there is an unique unrooted tree T, called the* representative tree *of $\mathcal{F}$, such that the set of leaves of T is V, the internal nodes of T are labelled* `degenerate` *or* `prime`*, and*

- for every edge e of T, $\{C^1_e, C^2_e\}$ is a strong member of $\mathcal{F}$, and there is no other strong member in $\mathcal{F}$,

- for every node α labelled `degenerate`*, and for every $\emptyset \subsetneq I \subsetneq \{1, \ldots, d(\alpha)\}$, $\{\cup_{i \in I} C^i_\alpha, V \setminus \cup_{i \in I} C^i_\alpha\}$ is in $\mathcal{F}$, and there is no other member in $\mathcal{F}$.*

Split decomposition. A *split* in a graph $G = (V, E)$ is a bipartition $\{X, Y\}$ of V such that the set of vertices in X having a neighbour in Y have the same neighbourhood in Y (*i.e.*, for all $u, v \in X$ such that $N(u) \cap Y \neq \emptyset$ and $N(v) \cap Y \neq \emptyset$, then $N(u) \cap Y = N(v) \cap Y$). A *co-split* in a graph G is a split in $\overline{G}$. The family of split in a connected graph is a bipartitive family [4]. The split decomposition tree is the representative tree of the family of splits, and can be computed in linear time [5]. Let α be an internal node of the split decomposition tree of a connected graph G. For all $i \in \{1, \ldots, d(\alpha)\}$ let $v_i \in C^i_\alpha$ such that $N(v_i) \setminus C^i_\alpha \neq \emptyset$. Since G is connected, such a v_i always exists. $G[\{v_1, \ldots, v_{d(\alpha)}\}]$ denote the *characteristic graph* of α. The characteristic graph of a `degenerate` node is a complete graph or a star [4].

Bi-join decomposition. A *bi-join* in a graph is a bipartition $\{X, Y\}$ such that for all $u, v \in X$, $\{N(u) \cap Y, Y \setminus N(u)\} = \{N(v) \cap Y, Y \setminus N(v)\}$. The family of bi-joins in a graph is bipartitive. The *bi-join decomposition tree* is the representative tree of the family of bi-joins, and can be computed in linear time [7, 8]. Let α be an internal node of the bi-join decomposition tree of a graph G. For all $i \in \{1, \ldots, d(\alpha)\}$ let $v_i \in C^i_\alpha$. $G[\{v_1, \ldots, v_{d(\alpha)}\}]$ denote the *characteristic graph* of α. The characteristic graph of a `degenerate` node is a complete bipartite graph or a disjoint union of two complete graphs [7, 8].

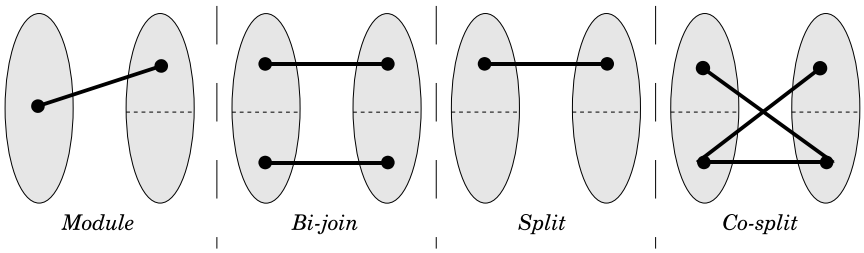

Figure 1: A module, a bi-join, a split and a co-split

3 Recognition of NLC-2 graphs

3.1 NLC-2 ρ-free canonical decomposition

In this section, $G = (V, E, l)$ is a 2-labelled graph such that every mono-coloured module (*i.e.* a module M such that $\forall v, v' \in M$, $l(v) = l(v')$) has size 1. A couple (X, Y) is a *cut* if $X \cup Y = V$, $X \cap Y = \emptyset$, $X \neq \emptyset$ and $Y \neq \emptyset$. Let $S \subseteq \{1,2\} \times \{1,2\}$. A cut (X, Y) is a *S-cut* of G if for all $u \in X$ and $v \in Y$, then $\{u, v\} \in E$ if and only if $(l(u), l(v)) \in S$. For $S \subseteq \{1,2\} \times \{1,2\}$ let $\mathcal{F}_S(G)$ be the set of S-cut of G.

Definition 5 (Symmetry). We say that $S \in \{1,2\} \times \{1,2\}$ is *symmetric* if $(1,2) \in S \iff (2,1) \in S$, otherwise we say that S is *non-symmetric*.

Definition 6 (Degenerate property). A family $\mathcal{F}$ of cuts has the *degenerate property* if there is a partition $\mathcal{P}$ of V such that for all $\emptyset \subsetneq \mathcal{X} \subsetneq \mathcal{P}$, $(\bigcup_{X \in \mathcal{X}} X, \bigcup_{Y \in \mathcal{P} \setminus \mathcal{X}} Y)$ is in $\mathcal{F}$, and there is no others cut in $\mathcal{F}$.

Lemma 7. *For every symmetric $S \subseteq \{1,2\} \times \{1,2\}$, $\mathcal{F}_S(G)$ has the degenerate property.*

Proof. The family $\mathcal{F}_{\{\}}(G)$ has the degenerate property since (X, Y) is a $\{\}$-cut if and only if there is no edges between X and Y ($\mathcal{P}$ is exactly the connected components). For $W \subseteq V$, let $G|W = (V, E\Delta W^2, l)$. For $i \in \{1,2\}$ let $V_i = \{v \in V : l(v) = i\}$. Let $G_1 = G|V_1$, $G_2 = G|V_2$ and $G_{12} = (G|V_1)|V_2$.

- $\mathcal{F}_{\{(1,1)\}}(G) = \mathcal{F}_{\{\}}(G_1)$, $\mathcal{F}_{\{(2,2)\}}(G) = \mathcal{F}_{\{\}}(G_2)$, $\mathcal{F}_{\{(1,1),(2,2)\}}(G) = \mathcal{F}_{\{\}}(G_{12})$,
- $\mathcal{F}_{\{(1,1),(1,2),(2,1),(2,2)\}}(G) = \mathcal{F}_{\{\}}(\overline{G})$, $\mathcal{F}_{\{(1,2),(2,1),(2,2)\}}(G) = \mathcal{F}_{\{\}}(\overline{G_1})$,
  $\mathcal{F}_{\{(1,1),(1,2),(2,1)\}}(G) = \mathcal{F}_{\{\}}(\overline{G_2})$, $\mathcal{F}_{\{(1,2),(2,1)\}}(G) = \mathcal{F}_{\{\}}(\overline{G_{12}})$.

Thus for every symmetric $S \subseteq \{1,2\} \times \{1,2\}$, $\mathcal{F}_S(G)$ has the degenerate property. □

Definition 8 (Linear property). A family $\mathcal{F}$ of cuts has the *linear property* if for all (X, Y) and (X', Y') in $\mathcal{F}$, either $X \subseteq X'$ or $X' \subseteq X$.

Lemma 9. *For every non-symmetric $S \subseteq \{1,2\} \times \{1,2\}$, $\mathcal{F}_S(G)$ has the linear property.*

Proof. Case $S = \{(1,2)\}$: suppose that $X \setminus X'$ and $X' \setminus X$ are both non-empty. Then if $u \in X \setminus X'$ is labelled 1 and $v \in X' \setminus X$ is labelled 2, u and v has to be adjacent and non-adjacent, contradiction. Thus $X \setminus X'$ and $X' \setminus X$ are mono-coloured. Now suppose w.l.o.g. that all vertices in $X\Delta X'$ are labelled 1. Then $X\Delta X'$ is adjacent to all vertices labelled 2 in $Y \cap Y'$ and non adjacent to all vertices labelled 1 in $Y \cap Y'$. Moreover $X\Delta X'$ is non adjacent to all vertices in $X \cap X'$. Thus $X\Delta X'$ is a mono-coloured module, and $|X\Delta X'| \geq 2$. Contradiction. For others non-symmetric S, we bring back to case $\{(1,2)\}$ like in the proof of lemma 7. □

Input: A 2-labelled graph $G = (V, E, l)$
Output: A NLC-2 ρ-free decomposition tree, or fail if G is not NLC-2 ρ-free

```
if |V| = 1 then return the leaf ·(l(v)) (where V = {v})
Let S be the set of subsets of {1,2} × {1,2} and σ be the lexicographic order of S
foreach S ∈ S w.r.t. σ do
    Compute P_S(G), and P'_S(G) if S is non-symmetric (see algorithm 2)
    if |P_S(G)| > 1 then
        Create a new ×_S node β
        foreach P ∈ P_S(G) (w.r.t. P'_S(G) if S is non-symmetric) do
            make NLC-2 ρ-free decomposition tree of G[P] be a child of β.
        return the tree rooted at β
fail with Not NLC-2 ρ-free
```

Algorithm 1: Computation of the NLC-2 ρ-free canonical decomposition tree

For $S \subseteq \{1,2\} \times \{1,2\}$, let $\mathcal{P}_S(G)$ denote the unique partition of V such that (1) for all $(X, Y) \in \mathcal{F}_S(G)$ and $P \in \mathcal{P}_S(G)$, $P \subseteq X$ or $P \subseteq Y$, and (2) for all $P, P' \in \mathcal{P}$, $P \neq P'$, there is a $(X, Y) \in \mathcal{F}_S(G)$ such that $P \subseteq X$ and $P' \subseteq Y$, or $P \subseteq Y$ and $P' \subseteq X$. For a non-symmetric $S \in \{1,2\} \times \{1,2\}$, let $\mathcal{P}'_S(G) = (P_1, \ldots, P_k)$ denote the unique ordering of elements in $\mathcal{P}_S(G)$ such that for all $(X, Y) \in \mathcal{F}_S(G)$, there is a l such that $X = \cup_{i \in \{1,\ldots,l\}} P_i$.

Lemma 10. *If G is in NLC-2 ρ-free, then there is a $S \subseteq \{1,2\} \times \{1,2\}$ such that $\mathcal{F}_S(G)$ is non-empty.*

Proof. If G is NLC-2 ρ-free, then there is a $S \subseteq \{1,2\} \times \{1,2\}$, and two graphs G_1 and G_2 such that $G = G_1 \times_S G_2$. Thus $(V(G_1), V(G_2)) \in \mathcal{F}_S(G)$ and $\mathcal{F}_S(G)$ is non empty. □

Lemma 11. *Let $G = (V, E, l)$ 2-labelled graph and let $S \subseteq \{1,2\} \times \{1,2\}$. If G is NLC-2 ρ-free and has no mono-coloured non-trivial module, then for all $P \in \mathcal{P}_S(G)$, $G[P]$ has no mono-coloured non-trivial module.*

Proof. If M is a mono-coloured module of $G[P]$, then M is a mono-coloured module of G. Contradiction. □

Lemma 12. *Let $G = (V, E, l)$ 2-labelled graph and let $S \subseteq \{1,2\} \times \{1,2\}$. Then G is NLC-2 ρ-free if and only if for all $P \in \mathcal{P}_S(G)$, $G[P]$ is NLC-2 ρ-free.*

Proof. The "only if" is immediate. Now suppose that for all $P \in \mathcal{P}_S(G)$, $G[P]$ is NLC-2 ρ-free. If S is symmetric, let $\mathcal{P}_S(G) = \{P_1, \ldots, P_{|\mathcal{P}_S(G)|}\}$. Then $G = ((G[P_1] \times_S G[P_2]) \times_S \ldots \times_S G[P_{|\mathcal{P}_S(G)|}]$, and G is NLC-2 ρ-free. Otherwise, if S is non-symmetric, let $\mathcal{P}'_S(G) = (P_1, \ldots, P_{|\mathcal{P}_S(G)|})$. Then $G = ((G[P_1] \times_S G[P_2]) \times_S \ldots \times_S G[P_{|\mathcal{P}_S(G)|}]$, and G is NLC-2 ρ-free. □

The *NLC-2 ρ-free decomposition tree* of a 2-labelled graph G is a rooted tree such that the leaves are the vertices of G, and the internal nodes are labelled by $\times_S$, with $S \subseteq \{1,2\} \times \{1,2\}$. An internal node is `degenerated` if S is symmetric, and `linear` if S is non-symmetric. By lemmas 10, 11 and 12, G is NLC-2 ρ-free if and only if it has a NLC-2 ρ-free decomposition tree. This decomposition tree is not unique. But we can define a *canonical decomposition tree* if we fix a total order on the subsets of $\{1,2\} \times \{1,2\}$ (for example, the lexicographic order). If two graphs are isomorphic, then they have the same canonical decomposition tree. Algorithm 1 computes the canonical decomposition tree of a 2-labelled prime graph, or fails if G is not NLC-2 ρ-free.

Algorithm 2 computes $\mathcal{P}_S$ and $\mathcal{P}'_S$ for a 2-labelled prime graph G and $S \subseteq \{1,2\} \times \{1,2\}$ in linear time. We need some additional definitions for this algorithm and its proof of correctness. A

Input: A 2-labelled graph G, and $S \subseteq \{1,2\} \times \{1,2\}$
Output: $\mathcal{P}_S$ if S is symmetric, $\mathcal{P}'_S$ if S is non-symmetric

$V_i \leftarrow \{v : v \in V \text{ and } l(v) = i\}$;
if $(1,1) \in S$ **then** $\mathcal{C}_1 \leftarrow$ co-connected components of $G[V_1]$;
else $\mathcal{C}_1 \leftarrow$ connected components of $G[V_1]$;
if $(2,2) \in S$ **then** $\mathcal{C}_2 \leftarrow$ co-connected components of $G[V_2]$;
else $\mathcal{C}_2 \leftarrow$ connected components of $G[V_2]$;
$\mathcal{B} = (\mathcal{C}_1, \mathcal{C}_2, E_j, E_m) \leftarrow$ the bipartite trigraph between the elements of $\mathcal{C}_1$ and $\mathcal{C}_2$;
if $S \cap \{(1,2),(2,1)\} = \emptyset$ **then**
  return *connected components of* $(\mathcal{C}_1, \mathcal{C}_2, E_j \cup E_m)$
else if $S \cap \{(1,2),(2,1)\} = \{(1,2),(2,1)\}$ **then**
  return *connected components of the bi-complement of* $(\mathcal{C}_1, \mathcal{C}_2, E_j)$
else Search all semi-joins of $\mathcal{B}$ (see appendix) ;

Algorithm 2: Computation of $\mathcal{P}_S$ and $\mathcal{P}'_S$

bipartite graph is a triplet (X, Y, E) such that $E \subseteq X \times Y$. The *bi-complement* of a bipartite graph (X, Y, E) is the bipartite graph $(X, Y, (X \times Y) \setminus E)$. A *bipartite trigraph (BT)* is a bipartite graph with two types of edges: the *join* edges and the *mixed* edges. It is denoted by $\mathcal{B} = (X, Y, E_j, E_m)$ where E_j are the set of *join* edges, and E_m the set of *mixed* edges. A *BT-module* in a BT is a $M \subseteq X$ or $M \subseteq Y$ such that M is a module in (X, Y, E_j) and there is no *mixed* edges between M and $(X \cup Y) \setminus M$. For $v \in X \cup Y$, let $N_j(v) = \{u \in X \cup Y : \{u, v\} \in E_j\}$ and $N_m(v) = \{u \in X \cup Y : \{u, v\} \in E_m\}$. Let $d_j(v) = |N_j(v)|$ and $d_m(v) = |N_m(v)|$. A *semi-join* in a BT (X, Y, E_j, E_m) is a cut (A, B) of $X \cup Y$, such that there is no edges between $A \cap Y$ and $B \cap X$, and there is only *join* edges between $A \cap X$ and $B \cap Y$.

In algorithm 2, $\mathcal{B}$ is obtained from the graph G. Vertices of X correspond to subsets of vertices labelled 1 in G, and vertices of Y correspond to subsets of vertices labelled 2. There is a *join* edge between M and M' in $\mathcal{B}$ if $M \textcircled{1} M'$ in G, and there is a *mixed* edge between $M \in X$ and $M' \in Y$ in $\mathcal{B}$ if there is at least an edge and a non-edge between M and M' in G. Such a graph $\mathcal{B}$ can easily be built in linear time from a given graph G. It suffices to consider a list and an array bounded by the number of component in G with the same colour. The following lemmas are close to observations in [9], but deal with BT instead of bipartite graphs (proofs are given in appendix).

Lemma 13. *Let $G = (X, Y, E_j, E_m)$ be a BT such that every BT-module has size 1. Let $(x_1, \ldots, x_{|X|})$ be X sorted by $(d_j(x), d_m(x))$ in lexicographic decreasing order. If (A, B) is a semi-join of G, then there is a $k \in \{0, \ldots, |X|\}$ such that $A \cap X = \{x_1, \ldots, x_k\}$.*

Lemma 14. *Let $k \in \{0, \ldots, |X|\}$ and $k' \in \{0, \ldots, |Y|\}$. Then $(A, (X \cup Y) \setminus A)$, where $A = \{x_1, \ldots, x_k, y_1, \ldots, y_{k'}\}$, is a semi-join of G if and only if $\sum_{i=1}^{k} d_j(x_i) - \sum_{i=1}^{k'} d_j(y_i) = k \times (|Y| - k')$ and $\sum_{i=1}^{k} d_m(x_i) - \sum_{i=1}^{k'} d_m(y_i) = 0$.*

Theorem 15. *Algorithm 2 is correct and runs in linear time.*

Proof. **Correctness:** Suppose that (A, B) is a S-cut. If $(1,1) \notin S$, then there is no edge between $A \cap V_1$ and $B \cap V_1$, thus (A, B) cannot cut a component $\mathcal{C}_1$ (and similarly for $(1,1) \in S$, and for $\mathcal{C}_2$). Now we work on the BT $\mathcal{B} = (\mathcal{C}_1, \mathcal{C}_2, E_j, E_m)$. If $S \cap \{(1,2),(2,1)\} = \emptyset$, then S-cuts correspond exactly to connected components of $\mathcal{B}$, and if $S \cap \{(1,2),(2,1)\} = \{(1,2),(2,1)\}$ then S-cuts correspond exactly to connected components of the BT of $\overline{G}$, which is $(\mathcal{C}_1, \mathcal{C}_2, (\mathcal{C}_1 \times \mathcal{C}_2) \setminus (E_j \cup E_m), E_m)$. Finally, if S is non-symmetric, S-cuts correspond to semi-joins of $\mathcal{B}$ (see appendix).

Complexity: It is well admitted that we can perform a BFS on a graph or its complement in linear time [13, 6]. The instructions on lines [2-5,8] can be done with a BFS on a graph or its complement. It is easy to see that we can do a BFS on the bi-complement in linear time (like a BFS on a complement graph, with two vertex lists for X and Y), so instruction line 10 can be done in linear time. Finally, the operations at line 11 are done in linear time (see appendix). $\square$

These results can be summarized as:

Theorem 16. *Algorithm 1 computes the canonical NLC-2 ρ-free decomposition tree of a 2-labelled graph in $O(nm)$ time.*

3.2 NLC-2 decomposition of a prime graph

In this section, G is an unlabelled prime (w.r.t. modular decomposition) graph, with $|V| \geq 3$.

Definition 17 (2-*bimodule*). A bipartition $\{X, Y\}$ of V is a 2-*bimodule* if X can be partitioned into X_1 and X_2, and Y into Y_1 and Y_2 such that for all $(i, j) \in \{1, 2\} \times \{1, 2\}$, then either X_i ⓪ Y_j or X_i ① Y_j. It is easy to see that if $\{X, Y\}$ is a 2-bimodule if and only if $\{X, Y\}$ is a split, a co-split or a bi-join. Moreover, if $\min(|X|, |Y|) > 1$ then $\{X, Y\}$ cannot be both of them in the same time (since G is prime).

Let $l : V \to \{1, 2\}$ be a 2-labelling. Then $s(l)$ denote the 2-labelling on V such that for all $v \in V$, $s(l)(v) = 1$ if and only if $l(v) = 2$.

Definition 18 (Labelling induced by a 2-bimodule). Let $\{X, Y\}$ be a 2-bimodule. We define the *labelling* $l : V \to \{1, 2\}$ *of* G *induced by* $\{X, Y\}$. If $|X| = |Y| = 1$, then $l(x) = 1$ and $l(y) = 2$, where $X = \{x\}$ and $Y = \{y\}$. If $|X| = 1$, then $l(v) = 1$ iff $v \in N[x]$. Similarly if $|Y| = 1$, then $l(v) = 1$ iff $v \in N[y]$. Now we suppose $\min(|X|, |Y|) > 1$. If $\{X, Y\}$ is a split, then the set of vertices in X with a neighbour Y and the set of vertices in Y with a neighbour in X is labelled 1, others vertices are labelled 2. If $\{X, Y\}$ is a co-split, then a labelling of G induced by $\{X, Y\}$ is a labelling of $\overline{G}$ induced by the split $\{X, Y\}$. Finally if $\{X, Y\}$ is a bi-join, l is such that $\{v \in X : l(v) = 1\}$ is a join with $\{v \in Y : l(v) = 1\}$ and $\{v \in X : l(v) = 2\}$ is a join with $\{v \in Y : l(v) = 2\}$. Note that if $\{X, Y\}$ is a bi-join, then there is two possibles labelling l_1 and l_2, with $l_1 = s(l_2)$. If $\{X, Y\}$ is a 2-bimodule of G and l a labelling induced by $\{X, Y\}$, then every mono-coloured module has size 1 (since G is prime and $|V| \geq 3$).

Definition 19 (Good 2-bimodule). A 2-bimodule $\{X, Y\}$ is *good* if the graph G with the labelling induced by $\{X, Y\}$ is NLC-2 ρ-free. The following proposition comes immediately from lemma 3.

Proposition 20. *G is NLC-2 if and only if G has a good 2-bimodule.*

Lemma 21. *If G has a good 2-bimodule $\{X, Y\}$ which is a split, then G has a good 2-bimodule which is a strong split.*

Proof. There is a node α in the split decomposition tree and $\emptyset \subsetneq I \subsetneq \{1, \ldots, d(\alpha)\}$ such that $\{X, Y\} = \{\cup_{i \in I} C^i_\alpha, \cup_{i \notin I} C^i_\alpha\}$. Let $l : V \to \{1, 2\}$ be the labelling of G induced by $\{X, Y\}$. For all $i \in \{1, \ldots, d(\alpha)\}$, $(G[C^i_\alpha], l|_{C^i_\alpha})$ is NLC-2 ρ-free (where $l|_W$ is the function l restricted at W).

Let l' be the 2-labelling of V such that for all i, and $v \in C^i_\alpha$, $l(v) = 1$ if and only if v has a neighbour outside of C^i_α. For all i, either $l|_{C^i_\alpha} = l'|_{C^i_\alpha}$, or $\forall v \in C^i_\alpha$, $l(v) = 2$. Then for all i, $(G[C^i_\alpha], l'|_{C^i_\alpha})$ is NLC-2 ρ-free, and thus (G, l') is NLC-2 ρ-free. Since there is a dominating vertex in the characteristic graph of α, there is a j such that the labelling induced by the strong split $\{C^j_\alpha, V \setminus C^j_\alpha\}$ is l'. Thus the strong split $\{C^j_\alpha, V \setminus C^j_\alpha\}$ is good. $\square$

Input: A graph G
Result: Yes iff G is NLC-2
$\mathcal{S} \leftarrow$ the set of strong splits, co-splits and bi-joins of G ;
foreach $\{X, Y\} \in \mathcal{S}$ **do**
  $l \leftarrow$ the labelling of G induced by $\{X, Y\}$;
  if $(G[X], G[Y], l)$ *is NLC-2 ρ-free* **then return** *Yes* ;
return *No* ;

Algorithm 3: Recognition of prime NLC-2 graphs

Previous lemma on $\overline{G}$ say that if G has a good 2-bimodule $\{X, Y\}$ which is a co-split, then G has a good 2-bimodule which is a strong co-split. The following lemma is similar to Lemma 21.

Lemma 22. *If G has a good 2-bimodule $\{X, Y\}$ which is a bi-join, then G has a good 2-bimodule which is a strong bi-join.*

Theorem 23. *Algorithm 3 recognises prime NLC-2 graphs, and its time complexity is $O(n^2m)$.*

Proof. Trivially if the algorithm return Yes, then G is NLC-2. On the other hand, by proposition 20, and lemmas 21 and 22, if G is NLC-2, then it has a good strong 2-bimodule and the algorithm returns Yes.

The set $\mathcal{S}$ can be computed using algorithms for computing split decomposition on G and $\overline{G}$, and bi-join decomposition on G. Note that it is not required to use a linear time algorithm for split decomposition [5]: some simpler algorithms run in $O(n^2m)$ [4, 10]. [7, 8] show that bi-join decomposition can be computed in linear time, using a reduction to modular decomposition. But there also, modular decomposition algorithms simpler than [15] may be used. The set $\mathcal{S}$ has $O(n)$ elements. Testing if a 2-bimodule is good takes $O(nm)$ using algorithm 1. So total running time is $O(n^2m)$. □

3.3 NLC-2 decomposition

Using lemma 2, modular decomposition and algorithm 3, we get:

Theorem 24. *NLC-2 graphs can be recognised in $O(n^2m)$, and a NLC-2 expression can be generated in the same time.*

4 Graph isomorphism on NLC-2 graphs

4.1 Graph Isomorphism on NLC-2 ρ-free prime graphs

The following propositions are direct consequences of properties (linear and degenerate) of S-cuts.

Proposition 25. *Consider a symmetric $S \in \{1, 2\} \times \{1, 2\}$. Two graphs G and H are isomorphic if and only if there is a bijection π between $\mathcal{P}_S(G)$ and $\mathcal{P}_S(H)$ such that for all $P \in \mathcal{P}_S(G)$, $G[P]$ is isomorphic to $H[\pi(P)]$.*

Proposition 26. *Let a non-symmetric $S \in \{1, 2\} \times \{1, 2\}$ and let G and H be two graphs. Let $\mathcal{P}'_S(G) = (P_1, \ldots, P_k)$ and $\mathcal{P}'_S(H) = (P'_1, \ldots, P'_{k'})$ then G and H are isomorphic if and only if $k = k'$ and for all $i \in \{1, \ldots, k\}$, $G[P_i]$ is isomorphic to $H[P'_i]$.*

By the previous 2 propositions, two NLC-2 ρ-free 2-labelled graphs G and H are isomorphic if and only if there is an isomorphism between their canonical NLC-2 ρ-free decomposition tree which respects the order of children of `linear` nodes. This isomorphism can be tested in linear time, thus isomorphism of NLC-2 ρ-free graphs can be done in $O(nm)$ time.

Input: Two prime NLC-2 graphs G and H
Result: Yes if $G \simeq H$, No otherwise
$\mathcal{S} \leftarrow$ the set of strong splits, co-splits and bi-joins of G ;
$\mathcal{S}' \leftarrow$ the set of strong splits, co-splits and bi-joins of H ;
if *there is no good* 2-*bimodule in* $\mathcal{S}$ **then fail with** "G is not NLC-2";
$\{X, Y\} \leftarrow$ a good 2-bimodule in $\mathcal{S}$;
$l \leftarrow$ the labelling of G induced by $\{X, Y\}$;
foreach $\{X', Y'\} \in \mathcal{S}'$ *such that* $\{X', Y'\}$ *is good* **do**
  $l' \leftarrow$ the labelling of H induced by $\{X', Y'\}$;
  if $|X| > 1$ **and** $|Y| > 1$ **and** $\{X, Y\}$ *is a bi-join* **then**
    if $(G, l) \simeq (H, l')$ **or** $(G, l) \simeq (H, s(l'))$ **then return** *Yes ;*
  else if $(G, l) \simeq (H, l')$ **then return** *Yes ;*
return *No* ;

Algorithm 4: Isomorphism for prime NLC-2 graphs

4.2 Graph isomorphism on prime NLC-2 graphs

Theorem 27. *Algorithm 4 test isomorphism between two prime NLC-2 graphs in time* $O(n^2m)$.

Proof. If the algorithm returns "yes", then trivially $G \simeq H$. On the other hand suppose that $G \simeq H$ and let $\pi : V(G) \to V(H)$ be a bijection such that $\{u, v\} \in E(G)$ iff $(\pi(u), \pi(v)) \in E(H)$. Then $\{X', Y'\}$ with $X' = \pi(X)$ and $Y' = \pi(Y)$ is a good 2-bimodule if H. If $\min(|X|, |Y|) > 1$ and $\{X', Y'\}$ is a bi-join, then by definition there is two labelling induced by $\{X, Y\}$, and $(G, l) \simeq (H, l')$ or $(G, l) \simeq (H, s(l'))$. Otherwise the labelling is unique and $(G, l) \simeq (H, l')$.

The sets $\mathcal{S}$ and $\mathcal{S}'$ can be computed in $O(n^2)$ time using linear time algorithms for computing split decomposition on G and $\overline{G}$, and bi-join decomposition on G. The sets $\mathcal{S}$ and $\mathcal{S}'$ have $O(n)$ elements. Test if a 2-bimodule is good take $O(nm)$ using algorithm 1, and test if two 2-labelled prime graphs are isomorphic take also $O(nm)$. Thus the total running time is $O(n^2m)$. □

4.3 Graph isomorphism on NLC-2 graphs

It is easy to show that graph isomorphism on prime NLC-2 graphs with an additional labels into $\{1, \ldots, q\}$ can be done in $O(n^2m)$ time. For that, we add the additional label of v at the leaf corresponding to v in the NLC-2 ρ-free decomposition tree.

We show that we can do graph isomorphism on NLC-2 graphs in time $O(n^2m)$, using the modular decomposition and algorithm 4. Let $\mathcal{M}(G)$ and $\mathcal{M}(H)$ be the modular decomposition of G and H. For $M \in \mathcal{M}(G)$, let G_M be $G[M]$, and for $M \in \mathcal{M}(H)$, let H_M be $H[M]$. Let G^*_M be the characteristic graph of G_M (note that $|V(G^*_M)|$ is the number of children of M in the modular decomposition tree). Let $\mathcal{M}_{(i,*)} = \{M \in \mathcal{M}(G) \cup \mathcal{M}(H) : |M| = i\}$, let $\mathcal{M}_{(*,j)} = \{M \in \mathcal{M}(G) \cup \mathcal{M}(H) : |V(G^*_M)| = j\}$ and let $\mathcal{M}_{(i,j)} = \mathcal{M}_{(i,*)} \cap \mathcal{M}_{(*,j)}$. Note that $\sum_{j=1}^{n}(\mathcal{M}_{(*,j)} \times j)$ is the number of vertices in G plus the number of edges in the modular decomposition tree, and thus is at most $3n - 2$.

Theorem 28. *Algorithm 5 tests isomorphism between two NLC-2 graphs in time* $O(n^2m)$.

Proof. The correctness comes from the fact that at each step, for all $M, M' \in \mathcal{M}(G) \cup \mathcal{M}(H)$ such that $l(M)$ and $l(M')$ are set, G_M and $G_{M'}$ are isomorphic if and only if $l(M) = l(M')$. The

Input: Two NLC-2 graphs G and H
Result: Yes if $G \simeq H$, No otherwise
for *every* $M \in \mathcal{M}(G) \cup \mathcal{M}(H)$ *such that* $|M| = 1$ **do** $l(M) \leftarrow 1$;
for i *from* 2 *to* n **do**
  for j *from* 2 *to* i **do**
    Compute the partition $\mathcal{P}$ of $\mathcal{M}_{(i,j)}$ such that M and M' are in the same class of $\mathcal{P}$ if and only if $(G^*_M, l) \simeq (G^*_{M'}, l)$. ;
    foreach $P \in \mathcal{P}$ **do**
      $a \leftarrow$ a new label (an integer not in $\mathrm{Img}(l)$) ;
      For all $M \in P$, $l(M) \leftarrow a$;

Algorithm 5: Isomorphism on NLC-2 graphs

total time $f(n,m)$ of this algorithm is $O(n^2m)$ since ("big Oh" is omitted):

$$\begin{aligned} f(n,m) &\leq \sum_i \sum_j \left(j^2 m |\mathcal{M}_{(i,j)}|^2\right) \leq m \sum_j \left(j^2 \sum_i \left(|\mathcal{M}_{(i,j)}|^2\right)\right) \\ &\leq m \sum_j \left(j^2 |\mathcal{M}_{(*,j)}|^2\right) \leq m \sum_j \left(\left(j|\mathcal{M}_{(*,j)}|\right)^2\right) \leq n^2 m. \end{aligned}$$

□

References

[1] M. Chein, M. Habib, and M.C. Maurer. Partitive hypergraphs. *Discrete Math.*, 37(1):35–50, 1981.
[2] B. Courcelle, J. Engelfriet, and G. Rozenberg. Handle-rewriting hypergraph grammars. *J. Comput. Syst. Sci.*, 46(2):218–270, 1993.
[3] W. H. Cunnigham and J. Edmonds. A combinatorial decomposition theory. *Canad. J. Math.*, 32:734–765, 1980.
[4] William H. Cunningham. Decomposition of directed graphs. *SIAM J. Algebraic Discrete Methods*, 3(2):214–228, 1982.
[5] E. Dahlhaus. Parallel algorithms for hierarchical clustering and applications to split decomposition and parity graph recognition. *J. Algorithms*, 36(2):205–240, 2000.
[6] E. Dahlhaus, J. Gustedt, and R. M. McConnell. Partially complemented representations of digraphs. *Discrete Math. Theor. Comput. Sci.*, 5(1):147–168, 2002.
[7] F. de Montgolfier and M. Rao. The bi-join decomposition. In *ICGT*, volume 22 of *ENDM*, pages 173–177, 2005.
[8] F. de Montgolfier and M. Rao. Bipartitives families and the bi-join decomposition. Technical report, https://hal.archives-ouvertes.fr/hal-00132862, 2005.
[9] J.-L. Fouquet, V. Giakoumakis, and J.-M. Vanherpe. Bipartite graphs totally decomposable by canonical decomposition. *Internat. J. Found. Comput. Sci.*, 10(4):513–533, 1999.
[10] C. P. Gabor, K. J. Supowit, and W.-L. Hsu. Recognizing circle graphs in polynomial time. *J. ACM*, 36(3):435–473, 1989.
[11] T. Gallai. Transitiv orientierbare Graphen. *Acta Math. Acad. Sci. Hungar.*, 18:25–66, 1967.
[12] F. Gurski and E. Wanke. Minimizing NLC-width is NP-Complete. In *WG*, volume 3787 of *LNCS*, pages 69–80, 2005.
[13] M. Habib, C. Paul, and L. Viennot. Partition refinement techniques: An interesting algorithmic tool kit. *Internat. J. Found. Comput. Sci.*, 10(2):147–170, 1999.
[14] Ö. Johansson. NLC_2-decomposition in polynomial time. *Internat. J. Found. Comput. Sci.*, 11(3):373–395, 2000.
[15] R. M. McConnell and J. P. Spinrad. Modular decomposition and transitive orientation. *Discrete Math.*, 201(1-3):189–241, 1999.
[16] E. Wanke. k-NLC Graphs and Polynomial Algorithms. *Discrete Appl. Math.*, 54(2-3):251–266, 1994.

Appendix

A.1 Proof of lemma 13

Let $G = (X, Y, E_j, E_m)$ be a BT such that every BT-module has size 1. Let $(x_1, \ldots, x_{|X|})$ be X sorted by $(d_j(x), d_m(x))$ in lexicographic decreasing order. If (A, B) is a semi-join of G, then there is a $k \in \{0, \ldots, |X|\}$ such that $A \cap X = \{x_1, \ldots, x_k\}$.

Proof. For all $v \in A \cap X$, $d_j(v) \geq |B \cap Y|$, and for all $v \in B \cap X$, $d_j(v) \leq |B \cap Y|$. Moreover, if there is a $v \in B \cap X$ with $d_j(v) = |B \cap Y|$, then $d_m(v) = 0$. Let $C = \{v \in X : d_j(v) = |B \cap Y| \text{ and } d_m(v) = 0\}$. Then C is a BT-module of G, and thus $|C| \leq 1$. Every vertex in $A \cap X \setminus C$ are before every vertex in $B \cap X \setminus C$ in the ordering. Moreover, if $|C| > 0$, then vertices in $A \cap X \setminus C$ are before the vertex in C, and vertices in $B \cap X \setminus C$ are after the vertex in C in the ordering. □

A.2 Proof of lemma 14

Let $k \in \{0, \ldots, |X|\}$ and $k' \in \{0, \ldots, |Y|\}$. Then $(A, (X \cup Y) \setminus A)$, where $A = \{x_1, \ldots, x_k, y_1, \ldots, y_{k'}\}$, is a semi-join of G if and only if $\sum_{i=1}^{k} d_j(x_i) - \sum_{i=1}^{k'} d_j(y_i) = k \times (|Y| - k')$ and $\sum_{i=1}^{k} d_m(x_i) - \sum_{i=1}^{k'} d_m(y_i) = 0$.

Proof. The "If" part is by definition. Now let us consider the "Only if" part. Let us assume that the degree condition holds. We will denote a the number of join edges between $A \cap X$ and $B \cap Y$, b the number of join edges between $A \cap X$ and $A \cap Y$, and c the number of mixed edges between $A \cap X$ and $A \cap Y$. Note that $a \leq k(|Y| - k')$, $a + b = \sum_{i=1}^{k} d_j(x_i)$ and $b \leq \sum_{i=1}^{k'} d_j(y_i)$, thus $a \geq k(|Y| - k')$. So we have $a = k(|Y| - k')$, and $\sum_{i=1}^{k'} d_j(y_i) - b = 0$. In other words, there is only join edges between $A \cap X$ and $B \cap Y$, and there is no join edges between $A \cap Y$ and $B \cap X$. Now since there is only join edges between $A \cap X$ and $B \cap Y$, $c = \sum_{i=1}^{k} d_m(x_i) = \sum_{i=1}^{k'} d_m(y_i)$, thus there is no mixed edges between $A \cap Y$ and $B \cap X$. □

A.3 Algorithm to compute $\mathcal{P}'_S$ when S is non-symmetric

Proof. **Correctness:** Algorithm 6 generates all the semi-joins of $\mathcal{B}$. At any time, $s_j = \sum_{i=1}^{k} d_j(x_i)$, $s_m = \sum_{i=1}^{k} d_m(x_i)$, $s'_j = \sum_{i=1}^{k'} d_j(y_i)$ and $s'_m = \sum_{i=1}^{k'} d_m(y_i)$. In $\mathcal{B}$, every BT-module has size 1, otherwise there is a mono-coloured module in G of size at least 2. If (A, B) is a semi-join, then by lemma 13 on $(\mathcal{C}_1, \mathcal{C}_2, E_j, E_m)$ and $(\mathcal{C}_2, \mathcal{C}_1, E_j, E_m)$, there is a a and b such that $A \cap \mathcal{C}_1 = \{x_1, \ldots, x_a\}$ and $A \cap \mathcal{C}_2 = \{y_1, \ldots, y_b\}$. At any time, $(A', (\mathcal{C}_1 \cup \mathcal{C}_2) \setminus A')$ with $A' = \{x_1, \ldots, x_l, y_1, \ldots, y_{l'}\}$ is the last semi-join found. At $k = a$, the **while** line 12 will stop when $s_j - s'_j = k \times (|\mathcal{C}_2| - k')$ since for every $v \in A \cap \mathcal{C}_2$, $d_j(v) \leq k$, and $s'_j + k \times (|\mathcal{C}_2| - k')$ decrease with k'. Moreover, when the **while** loop stops, $s_m = s'_m$ since s'_m increase with k'. Thus if $b \neq k'$, then $\{y_{k'+1}, \ldots y_b\}$ is a BT-module and $b = k' + 1$ (since every BT-module has size 1). In all cases the algorithm finds (A, B), and adds the partition in $\mathcal{P}'$.

Complexity: As we see in proof of theorem 15, every instruction lines [2-5] can be done in linear time, and clearly every instruction lines [6-22] can be done in linear time, thus the total running time is $O(n + m)$. □

Input: A 2-labelled graph G, and a non-symmetric $S \subseteq \{1,2\} \times \{1,2\}$
Output: $\mathcal{P}'_S$

$V_i \leftarrow \{v : v \in V \text{ and } l(v) = i\}$;
if $(1,1) \in S$ **then** $\mathcal{C}_1 \leftarrow$ co-connected components of $G[V_1]$;
else $\mathcal{C}_1 \leftarrow$ connected components of $G[V_1]$;
if $(2,2) \in S$ **then** $\mathcal{C}_2 \leftarrow$ co-connected components of $G[V_2]$;
else $\mathcal{C}_2 \leftarrow$ connected components of $G[V_2]$;
$\mathcal{B} = (\mathcal{C}_1, \mathcal{C}_2, E_j, E_m) \leftarrow$ the bipartite trigraph between the elements of $\mathcal{C}_1$ and $\mathcal{C}_2$;
$(x_1, \ldots, x_{|\mathcal{C}_1|}) \leftarrow \mathcal{C}_1$ sorted by lexicographic order on $(-d_j(v), -d_m(v))$;
$(y_1, \ldots, y_{|\mathcal{C}_2|}) \leftarrow \mathcal{C}_2$ sorted by lexicographic order on $(d_j(v), d_m(v))$;
$\mathcal{P}' \leftarrow ()$; $l \leftarrow 0$; $l' \leftarrow 0$; $k' \leftarrow 0$; $k \leftarrow 0$;
$s_j \leftarrow 0$; $s_m \leftarrow 0$; $s'_j \leftarrow 0$; $s'_m \leftarrow 0$;
while $k \leq |\mathcal{C}_1|$ **do**
  while $s_j - s'_j < k \times (|\mathcal{C}_2| - k')$ **or** $(\ s_j - s'_j = k \times (|\mathcal{C}_2| - k')$ **and** $s_m > s'_m)$ **do**
    $k' \leftarrow k' + 1$; $s'_j \leftarrow s'_j + d_j(y_{k'})$; $s'_m \leftarrow s'_m + d_m(y_{k'})$;
  if $s_j - s'_j = k \times (|\mathcal{C}_2| - k')$ **and** $s_m = s'_m$ **then**
    add $\{x_{l+1}, \ldots, x_k\} \cup \{y_{l'+1} \ldots, y_{k'}\}$ at the end of $\mathcal{P}'$; $l \leftarrow k$; $l' \leftarrow k'$;
    if $s_j - s'_j - d_j(y_{k+1}) = k \times (|\mathcal{C}_2| - k' - 1)$ **and** $s_m = s'_m + d_m(y_{k+1})$ **then**
      $k' \leftarrow k' + 1$; $s'_j \leftarrow s'_j + d_j(y_{k'})$; $s'_m \leftarrow s'_m + d_m(y_{k'})$;
      add $\{y_{k'}\}$ at the end of $\mathcal{P}'$; $l' \leftarrow k'$;
  $k \leftarrow k + 1$; $s_j \leftarrow s_j + d_j(x_k)$; $s_m \leftarrow s_m + d_m(x_k)$;
remove $\emptyset$ form $\mathcal{P}'$, if any ;
if $(2,1) \in S$ **then** reverse $\mathcal{P}'$;
return $\mathcal{P}'$

Algorithm 6: Computation of $\mathcal{P}'_S$ when S is non-symmetric