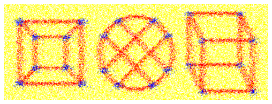

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 21, no. 4, pp. 491–525 (2017)
DOI: 10.7155/jgaa.00426

Bounded, minimal, and short representations of unit interval and unit circular-arc graphs. Chapter II: algorithms

Francisco J. Soulignac [1,2]

[1]CONICET
[2]Universidad Nacional de Quilmes, Buenos Aires, Argentina

Abstract

This is the second and last chapter of a work in which we consider the unrestricted, minimal, and bounded representation problems for unit interval (UIG) and unit circular-arc (UCA) graphs. In the unrestricted version (Rep), a proper circular-arc (PCA) model $\mathcal{M}$ is given and the goal is to obtain an equivalent UCA model $\mathcal{U}$. In the bounded version (BoundRep), $\mathcal{M}$ is given together with some lower and upper bounds that the beginning points of $\mathcal{U}$ must satisfy. In the minimal version (MinUCA), the circumference of the circle and the length of the arcs in $\mathcal{U}$ must be simultaneously as small as possible, while the separation of the extremes is greater than a given threshold. In this chapter we take advantage of the theoretical framework developed in Chapter I to design efficient algorithms for these problems. We show a linear-time algorithm with negative certification for Rep, that can also be implemented to run in logspace. We develop algorithms for different versions of BoundRep that run in linear space and quadratic time. Regarding MinUCA, we first show that the previous linear-time algorithm for MinUIG (i.e., MinUCA on UIG models) fails to provide a minimal model for some input graphs. We fix this algorithm but, unfortunately, it runs in linear space and quadratic time. Then, we apply the algorithms for MinUIG and MinUCA (Chapter I) to find the minimum powers of paths and cycles that contain given UIG and UCA models, respectively.

Submitted:	Reviewed:	Revised:	Accepted:	Final:
February 2016	October 2016	October 2016	February 2017	February 2017
		Published: April 2017		
	Article type: Regular paper		Communicated by: S. Whitesides	

Supported by PICT ANPCyT grants 2013-2205 and 2015-2419, and by PUNQ Grant 1451/15.
E-mail address: francisco.soulignac@unq.edu.ar (Francisco J. Soulignac)

1 Introduction

This is the second and last chapter of a work that is concerned with some representation problems for unit interval and unit circular-arc models. Although the work is envisioned as one unit, a split was required during the reviewing process because of its length. Chapter I [20] provides a thorough motivation to study the representation problems at hand, while it prepares the theoretical ground required for Chapter II. Chapter II, i.e. the present manuscript, deals with the implementation of efficient algorithms for the different problems. Naturally, some information is repeated in both chapters to make them self-contained.

Besides the contributions on each of the representation problems, our main goal is to show that *synthetic graphs* —which were originally defined for proper interval models— provide a convenient tool for studying proper circular-arc models, both from the theoretical and algorithmic points of view. Yet, this goal is achieved only when Chapters I (theory) and II (algorithms) are considered as being parts of a whole. Chapter I provides this unified view as it contains a detailed summary, that we shall not repeat here, of all the contributions of this work. Nor do we repeat the motivations for studying the representation problems, as they follow from Chapter I. Instead, the remainder of this section introduces what are synthetic graphs, what they represent, and why they play an important role in the study of interval and circular-arc models. This presentation is done from a big picture perspective and, so, it complements the introduction of Chapter I, whose focus is on the particular applications. Later, we provide a brief introduction to each representation problem in its corresponding section. We remark that, although some key concepts are briefly introduced in informal terms, the formal definitions are deferred to the subsequent sections.

A *proper circular-arc (PCA) model* is a pair $\mathcal{M} = (C, \mathcal{A})$ where C is a circle and $\mathcal{A}$ is a family of inclusion-free arcs of C in which no pair of arcs in $\mathcal{A}$ cover C. Each arc $A = (s, t) \in \mathcal{A}$ is described by its *extreme points*, being $s(A) = s$ the *beginning point* and $t(A) = t$ the *ending point*. We assume C has a special point, called 0, such that $s(A)$ and $t(A)$ correspond to the lengths of the arcs that go from 0 to $s(A)$ and to $t(A)$, respectively. This allows us to define an ordering $<$ on $\mathcal{A}$ such that $A_1 < A_2$ if and only if $s(A_1) < s(A_2)$. In this work, a *proper interval (PIG) model* is a PCA model in which no arc contains 0. *Unit circular-arc (UCA)* and *unit interval (UIG)* models correspond to the PCA and PIG models in which all the arcs have the same length, respectively. Through this work we also assume that no arc shares an extreme with another arc.

The aim of this Chapter is to provide new algorithms to improve the time and space complexities required to solve different representation problems on UCA models. The main results, together with a high-level description of the problems, are given in Section 1.1. What the problems have in common is that they follow the same generic specification. The input is a PCA model $\mathcal{M}$ with arcs $A_1 < \ldots < A_n$ and a set S of "separation constraints". The output is an UCA model $\mathcal{U}$ with arcs $U_1 < \ldots < U_n$ such that:

(i) $\mathcal{U}$ is *equivalent* to $\mathcal{M}$, i.e., $s(A_i)$ (resp. $t(A_i)$) is the j-th extreme point

that appears in a traversal of the circle of $\mathcal{M}$ from 0 if and only if $s(U_i)$ (resp. $t(U_i)$) is the j-th extreme point that appears when traversing, from 0, the circle of $\mathcal{U}$,

(ii) the beginning points of the arcs in $\mathcal{U}$ satisfy the "separation constraints", and

(iii) $\mathcal{U}$ has some properties of interest that depend on the problem at hand.

A *separation constraint* is an inequality that dictates how far or close must a beginning points $s(U_i)$ be from either 0 or another beginning point of $\mathcal{U}$. For instance, it could specify that $s(U_i) \geq 3$ or $s(U_i) \geq s(U_j) - 7$. Regarding 0 as the beginning point of a fictitious arc U_0, we can define such constraints as being triplets (i, j, δ_{ij}), for $0 \leq i, j \leq n$, that represent the inequality $s(U_j) \geq s(U_i) + \delta_{ij}$.

An *ℓ-IG* model is a UIG model whose arcs have length ℓ. A key fact observed by Pirlot [16] is that, given a PIG model $\mathcal{M}$ and a length ℓ, we can model the existence of an ℓ-IG model $\mathcal{U}$ equivalent to $\mathcal{M}$ by using $O(n)$ separation constraints. For instance, if $A_i \cap A_j = \emptyset$ for $A_i < A_j$, then we can specify that $U_i \cap U_j \neq \emptyset$ with the inequality $s(U_j) \geq s(U_i) + \ell + 1$. Similarly, Klavík et al. [9] use separation constraints that involve 0 to bound the position of the beginning points in $\mathcal{U}$. This construction can be applied to PCA models as well; all we have to do is to add the circumference of the circle into the game. Say that a UCA model is a *(c, ℓ)-CA model* when its circle has circumference c while its arcs have length ℓ. Chapter I shows that $O(n)$ separation constraints, that involve c and ℓ, are enough to express the existence of a (c, ℓ)-CA model $\mathcal{U}$ equivalent to a PCA model $\mathcal{M}$. Once the separation constraints are defined, we can reduce the problem of finding $\mathcal{U}$ to that of finding appropriate values for the beginning points. That is, given a set of separation constraints S whose triplets (i, j, δ_{ij}) are such that $0 \leq i, j \leq n$, the goal is to find the values $p(0)$, $\ldots$, $p(n)$ such that $p(j) \geq p(i) + \delta_{ij}$, for every triplet $(i, j, \delta_{ij}) \in S$. We say that S is *satisfiable* when p exists; in such case p is a *solution* to S.

Let G_S be the weighed (multi)digraph that has a vertex v_i, for $0 \leq i \leq n$, and an edge $v_i \to v_j$ with weight δ_{ij}, for $(i, j, \delta_{ij}) \in S$. It is well known that S is satisfiable if and only if G_S has no cycles of positive weight (cf. [18]). Furthermore, the function p such that $p(i)$ is the weight of the heaviest path from v_0 to v_i is a solution to S. That is, a solution to S can be obtained by solving the shortest path problem on G_S. In a nutshell, the idea to solve the different representation problems on $\mathcal{M}$ is to specify the output by using a set of separation constraints S. From S we build the graph G_S that describe the solutions to S, and we seek one solution by solving different path (or connectivity) problems.

The goal of the *representation problem* for PIG graphs is to find a UIG model $\mathcal{U}$ equivalent to an input PIG graph $\mathcal{M}$. As mentioned above, the representation problem can be reduced to that of finding a solution to a specific set S of $O(n)$ separation constraints [15, 17]. Pirlot [16, 17] refers to G_S as being the *synthetic graph* of $\mathcal{M}$. The name could come from the fact that S is a simplification (i.e., synthesis) of a family with $O(n^2)$ inequalities, or from the fact that the

unweighted version of G_S is a *compact* (i.e., requiring $O(n)$ space) representation of $\mathcal{M}$. The fact that G_S uniquely represents $\mathcal{M}$ is a desired property, because S defines the representation problem. Not surprisingly, the representation problem can be specified by different sets of separation constraints, while other sets of separation constraints arise when different problems are specified. In the latter case, the corresponding graphs are also representations of $\mathcal{M}$, but adorned with certain vertices and edges that are particular to the specific problem. Following Pirlot's terminology, we refer to all these graphs as being *synthetic graphs*.

Synthetic graphs appeared more than two decades ago, and they are covered in detail in a book by Pirlot and Vincke [18, Chapter 4]. Yet, they have gone unnoticed by many researchers in the field of algorithmic graph theory (see Chapter I). The reason for this ignorance could be, perhaps, the fact that Pirlot's article is written in terms of semiorders; its emphasis is on preference modeling and order theory. Subsequent articles on synthetic graph are also written in terms of semiorders. Up to our knowledge, the first graph theoretical article that uses synthetic graphs appeared in 2017 [9]. Yet, the preprint of this manuscript [8], from 2012, has no mention to Pirlot's work. In fact, the authors claim [8, p. 2] that "specific properties of unit interval representations were never investigated [since] it is easier to work with combinatorially equivalent proper interval representations".[1] The published version acknowledges the the present manuscript and its relation to Pirlot's work, but it does not show the strong relation between their techniques and those by Pirlot. In fact, the authors fail to mention that the so-called *left-most* representation was studied by Pirlot, even though this is one of the main problems in the present manuscript (see Section 5). So, besides improving some known algorithms for the different recognition problems, our main meta-contribution is to bring the attention to synthetic graphs, showing that they provide a simpler theoretical ground for understanding PCA models with separation constraints. To highlight this fact, we even re-prove some known theorems and rewrite some known algorithms in terms of synthetic graphs.

1.1 The algorithmic problems: contributions

The next paragraphs describe the main contributions of the present chapter. We recall that these paragraphs use an informal language and omit some important details; the formal specifications of the problems are deferred to Section 2. Remember also that Section I.1 provides a thorough motivation to study each of these problems.

The (unrestricted) representation problem. In the *representation* (Rep) problem a UCA model equivalent to an input PCA model $\mathcal{M}$ must be generated. Of course, Rep is unsolvable when $\mathcal{M}$ is equivalent to no UCA model; a *negative*

[1] N.B.: this assertion was replaced with "specific properties of unit interval representations were not much investigated" in [9]

witness is desired in such a case. In Section 4.1 we provide two algorithms for Rep, one runs in linear time and the other one runs in logspace.

Theorem 2 *There is an algorithm that solves* Rep *in $O(n)$ time.*

Theorem 5 *There is an algorithm that solves* Rep *in logspace.*

Rep is strongly related to the *recognition* problem for UCA graphs, whose goal is to determine if a given graph G is UCA. Again, we seek for a *certifying algorithm* that outputs either a UCA model of G or a witness certifying that G is not UCA. We can solve the recognition problem in two steps. First, we compute a PCA model $\mathcal{M}$ of G. Then, we apply Rep to transform $\mathcal{M}$ into an equivalent UCA model. This algorithm is correct because G is UCA if and only if every PCA model $\mathcal{M}$ representing G graph is equivalent to some UCA model [22]. As for the complexity, we can find a PCA model of G in either linear time [7] or logspace [10]. Thus, the whole algorithm runs in either linear time or logspace. Moreover, as the algorithms in [7] and [10] are certifying, we obtain that the whole recognition algorithm is certifying.

The bounded representation problems. We use the term *bounded representation* to encompass a family of similar representation problems. All these problems are generalizations of Rep, as a UCA model equivalent to an input PCA model $\mathcal{M}$ must be found. The difference is that at least three other input parameters are required for the bounded problems: two lengths $c, \ell \in \mathbb{Q}_{\geq 0}$ and a set of separation constraints S, all of whose triplets contain the 0 point. The goal, then, is to find a (c, ℓ)-CA model $\mathcal{U}$ equivalent to $\mathcal{M}$ whose beginning points satisfy the separation constraints in S. The term *bounded* comes from the fact that all the constraints in S are of the form $s(A) \geq b$ or $s(A) \leq c - b$ for some *bound* $b \in \mathbb{Q}_{\geq 0}$. As in Rep, a negative witness is desired when the problem is unsolvable.

We consider three variants of bounded representation problems: BoundRep, IntBoundRep, and u-Rep. These variants depend on whether c and ℓ are integer, the arcs of $\mathcal{U}$ are required to have integer extremes, and the additional constraints that S could contain. The details, together with a comparison of the problems, appear in Section 2. The main result is that we can solve all the discussed problems in quadratic time and linear space.

Theorem 1 BoundRep*,* IntBoundRep*, and* u-Rep *can be solved in $O(n^2)$ time and $O(n)$ space.*

The minimal representation problems. The *minimal representation* problems also refer to a family of problems. In the general setting, we are given a UCA model $\mathcal{M}$ that satisfies a set S of separation constraints, and the goal is to find an "S-minimal" model equivalent to $\mathcal{M}$. Roughly speaking, a (c, ℓ)-CA model $\mathcal{U}$ satisfying S is *S-minimal* when $c \leq c'$ and $\ell \leq \ell'$ for every (c', ℓ')-CA model that satisfies S. Of course, it is not obvious that such a minimal S-model

should exist. In this work we solve two flavors of minimal representation problems: MinUIG and IntMinUCA; in the former problem both $\mathcal{M}$ and $\mathcal{U}$ are UIG, while in the latter they are UCA. The important observation is that both problems are solvable; see [16] for MinUIG and Chapter I for MinUCA.

The negative result of this article is that the linear-time algorithm by Mitas [15] has a flaw that prevents it from solving MinUIG on some inputs (see Section 5). Fortunately, her algorithm correctly solves Rep and, so, it can be used to find a "short" UIG model when an input PIG model is given. We patch Mitas' algorithm to solve MinUIG; unfortunately, the new algorithm runs in $O(n^2)$ time. Solving MinUIG in linear time remains, thus, an open problem.

Theorem 7 MinUIG *can be solved in $O(n^2)$ time and linear space.*

Powers of paths and cycles. Let C_q^k (resp. P_q^k) be the k-th power of the cycle (path) graph on q vertices. Lin et al. [12] observed that a graph G is a UCA (resp. UIG) if and only if G is an induced subgraph of C_q^k (resp. P_q^k) for some q, k (see also [5] for UIG graphs and [6] for UCA graphs). In the *minimal power of a cycle* MinC_q^k *(resp.* MinP_q^k*)* problem we are given a UCA (resp. UIG) model $\mathcal{M}$ of a graph G, and the goal is to find an equivalent UCA model $\mathcal{U}$ that "implicitly" represents C_q^k (resp. P_q^k), where q and k are as small as possible. In Section 6 we show that MinP_q^k and MinC_q^k are strongly related to MinUIG and IntMinUCA, respectively. As a consequence of this fact, MinC_q^k can be solved in $O(n^4 \log n)$ time and linear space, and MinP_q^k can be solved in $O(n^2)$ time and linear space.

2 Preliminaries

In this article we consider (simple) graphs, (simple) digraphs, and q-digraphs. A *q-digraph*, for $q \in \mathbb{N}$, is a $(q+1)$-tuple $G = (V, E_1, \ldots, E_q)$ such that (V, E_i) is a digraph, for $1 \leq i \leq q$. Clearly, every digraph is a 1-digraph. For the sake of simplicity, we refer to the directed edges in E_i as being *edges* of G, unless otherwise stated. For a (q-di)graph G, we write $V(G)$ and $E(G)$ to denote the sets of vertices and (bag of) edges of G, respectively, while we use n and m to denote $|V(G)|$ and $|E(G)|$, respectively. For any pair $u, v \in V(G)$, we write uv to denote the pair (u, v); note that uv is an unordered pair when G is a graph, while it is an ordered pair when G is a q-digraph. To avoid confusion, we write $u \to v$ as an equivalent of uv when G is a q-digraph. Sometimes we may refer to the pair uv as being the (directed) edge between u and v (*from or starting at u to or ending at v*), regardless of whether $uv \in E(G)$. The *in-* and *out-degrees* of v in a q-digraph are the number of edges of G starting and ending at v, respectively.

A walk W of a (q-di)graph G is a sequence of edges $v_1v_2, v_2v_3 \ldots, v_{k-1}v_k$ of G. Walk W goes *from (or starts at) v_1 to (or ends at) v_k*. We say that W is a circuit when $v_k = v_1$, that W is a *path* when $v_i \neq v_j$ for every $1 \leq i < j \leq k$, and that W is a *cycle* when it is a circuit and $v_1v_2, \ldots, v_{k-2}v_{k-1}$ is a path. If G

contains no cycles, then G is an *acyclic* (q-di)graph. For the sake of notation, we could say that W is a *circuit* when $v_1 \neq v_k$; this means that W, v_kv_1 is a circuit. Moreover, we may write that a sequence of vertices $v_1, \ldots, v_k$ is a *walk* of G to express that some sequence of edges $v_1v_2, \ldots, v_{k-1}v_k$ is a walk of G. Both conventions are ambiguous when G is a q-digraph for $q > 1$, as there could be q edges from v_i to v_{i+1} (or from v_k to v_1 in the former case). In general, the edge represented by v_iv_{i+1} is clear by context. When this is not the case, v_iv_{i+1} refers to any of the edges from v_i to v_{i+1}.

An *edge weighing*, or simply a *weighing*, of a (q-di)graph G is a function $w\colon E(G) \to \mathbb{R}$. The value $w(uv)$ is referred to as the *weight* of uv (with respect to w). For any bag of edges E, the *weight* of E (with respect to an edge weighing w) is $w(E) = \sum_{uv \in E} w(uv)$. We use two distance measures on a (q-di)graph G with a weighing w. For $u, v \in V(G)$, we denote by $\mathbf{d}^*w(G, u, v)$ the maximum among the weights of the walks from u to v, while $\mathbf{d}w(G, u, v)$ denotes the maximum among the weights of the paths starting at u and ending at v. Note that $\mathbf{d}w(G, u, v) < \infty$ for every u, v, while $\mathbf{d}^*w(G, u, v) = \mathbf{d}w(G, u, v)$ when G contains no cycle of positive weight [1]. For a weighing w', we write $(\mathbf{d}w \circ \mathbf{d}w')(G, u, v) = \max\{w(W) \mid W \text{ is a path from } u \text{ to } v \text{ with } w'(W) = \mathbf{d}w'(G, u, v)\}$. In other words, $\mathbf{d}w \circ \mathbf{d}w'$ measures the "w-distance" from u to v when only those paths that impose the maximum "w'-distance" from u to v are considered. For the sake of notation, we omit the parameter G when there are no ambiguities.

A *straight plane* (q-di)graph, or simply a *plane* (q-di)graph, is a (q-di)graph whose vertices are coordinates in the plane and whose edges are non-crossing straight lines. Similarly, a *toroidal* (q-di)graph is a (q-di)graph whose vertices and edges can be placed on the surface of a torus in such a way that no pair of edges intersect.

A *proper circular-arc* (PCA) model $\mathcal{M}$ is a pair $(C, \mathcal{A})$, where C is a circle and $\mathcal{A}$ is a collection of open arcs of C such that no arc contains another arc and no pair of arcs in $\mathcal{A}$ cover C. When traversing the circle C, we always choose the clockwise direction. If s, t are points of C, we write (s, t) to mean the arc of C defined by traversing the circle from s to t, and $|s, t|$ to mean the length of (s, t). Sometimes we refer to $|s, t|$ as being the *separation* from s to t. Points s and t are the *extremes* of (s, t), while s is its *beginning point* and t its *ending point*. For $A \in \mathcal{A}$, we write $A = (s(A), t(A))$. The *extremes* of $\mathcal{A}$ are those of all arcs in $\mathcal{A}$. In this article we assume that no pair of extremes of $\mathcal{A}$ coincide. An ordered pair of extremes s_1s_2 of $\mathcal{M}$ is *consecutive* when there is no extreme $s \in (s_1, s_2)$ (note that s_2s_1 is not consecutive in this case, unless $|\mathcal{A}| = 1$). We assume C has a special point 0 with the property that $s(A_i) = |0, s(A_i)|$ and $t(A_i) = |0, t(A_i)|$, for every $1 \leq i \leq n$. For every pair of points p_1, p_2, we write $p_1 < p_2$ to indicate that p_1 appears before p_2 in a traversal of C from 0. Similarly, we write $A_1 < A_2$ to mean that $s(A_1) < s(A_2)$ for any pair of arcs A_1, A_2 on C.

A *unit circular-arc* (UCA) model is a PCA model $\mathcal{M}$ in which all the arcs have the same length. Let $A_1 < \ldots < A_n$ be the arcs of $\mathcal{M} = (C, \mathcal{A})$, $c, \ell \in \mathbb{Q}_{>0}$, $d, d_s \in \mathbb{Q}_{\geq 0}$, and $d_\ell, d_r\colon \mathcal{A} \to \mathbb{Q}_{\geq 0}$. We say that $\mathcal{M}$ is a $(c, \ell, d, d_s, d_\ell, d_r)$-*CA*

model when:

(unit_1) C has circumference c,

(unit_2) all the arcs of $\mathcal{A}$ have length ℓ,

(unit_3) $|p_1, p_2| \geq d$ for every pair of consecutive extremes $p_1 p_2$,

(unit_4) $|s_1, s_2| \geq d + d_s$ for any pair of beginning points s_1, s_2, and

(unit_5) $d_\ell(A_i) \leq s(A_i) \leq c - d_r(A_i)$ for every $1 \leq i \leq n$.

Intuitively, $\mathcal{M}$ is a UCA model in which consecutive extremes are separated by at least d space, the beginning points are separated by $d + d_s$ space, and $d_\ell(A_i)$ and $d_r(A_i)$ are lower bounds of the separation from 0 to $s(A_i)$ and from $s(A_i)$ to 0, respectively. We simply write that $\mathcal{M}$ is a (c, ℓ, d, d_s)-*CA model* to indicate that $d_\ell = d_r = 0$, and that $\mathcal{M}$ is a (c, ℓ)-CA model to mean that $\mathcal{M}$ is a $(c, \ell, 1, 0)$-CA model. To further simplify the notation, we refer to the tuple $u = (c, \ell, d, d_s, d_\ell, d_r)$ as a *UCA descriptor*, and we say that u is *integer* when c, ℓ, d, d_s, d_ℓ, and d_r are integers. Similarly, a u-CA model $\mathcal{M}$ is *integer* when c, ℓ and all the extremes of $\mathcal{M}$ are integers.

A *proper interval* (PIG) model is a PCA model $\mathcal{M}$ in which no arc crosses 0; if $\mathcal{M}$ is also UCA, then $\mathcal{M}$ is a *unit* interval (UIG) model. Any UIG model $\mathcal{M}$ is a u-CA model for some large enough c; for simplicity, we just write $c = \infty$ in this case. For this reason, we say that $\mathcal{M}$ is an (ℓ, d, d_s)-IG (resp. ℓ-IG) model when $\mathcal{M}$ is a (∞, ℓ, d, d_s)-CA (resp. (∞, ℓ)-CA) model. That is, $\mathcal{M}$ is an (ℓ, d, d_s)-IG model when all the arcs have length ℓ, every pair of consecutive extremes is separated by d space, and every pair of beginning points is separated by $d + d_s$ space.

Each PCA model $\mathcal{M}$ defines a graph $G(\mathcal{M})$ that contains a vertex for each arc of $\mathcal{M}$ where two vertices are adjacent if and only if their corresponding arcs have nonempty intersection. We say that $\mathcal{M}$ *represents* a graph G, and that G *admits* $\mathcal{M}$, when G is isomorphic to $G(\mathcal{M})$. A graph is a *proper circular-arc (PCA)*, *unit circular-arc (UCA)*, *proper interval (PIG)*, or *unit interval (UIG)* graph when it admits a PCA, UCA, PIG, or UIG model, respectively.

Clearly, two PCA models $\mathcal{M}_1 = (C_1, \mathcal{A}_1)$ and $\mathcal{M}_2 = (C_2, \mathcal{A}_2)$ are equal when $C_1 = C_2$ and $\mathcal{A}_1 = \mathcal{A}_2$. We say that $\mathcal{M}_1$ is *equivalent* to $\mathcal{M}_2$ when the extremes of $\mathcal{M}_1$ appear in the same order as in $\mathcal{M}_2$ in the traversals of C_1 and C_2 from their respective 0 points. Formally, $\mathcal{M}_1$ and $\mathcal{M}_2$ are equivalent if there exists $f\colon \mathcal{A}_1 \to \mathcal{A}_2$ such that $e(f(A)) < e'(f(B))$ if and only if $e(A) < e'(B)$, for $e, e' \in \{s, t\}$. By definition, $\mathcal{M}_1$ and $\mathcal{M}_2$ are equivalent whenever they are equal.

In this manuscript we consider several representation problems. In the *(unrestricted) representation* (Rep) problem a UCA model equivalent to an input PCA model $\mathcal{M}$ must be generated. Recall that Rep is solvable and only if $G(\mathcal{M})$ is UCA; a *negative witness* must be provided when $G(\mathcal{M})$ is not UCA. In the u-Rep problem, a (an integer) UCA descriptor u is given together with $\mathcal{M}$, and the goal is to build a (an integer) u-CA model $\mathcal{U}$. Clearly, $\mathcal{U}$ is a solution to Rep;

this time, however, $\mathcal{U}$ need not exists when $G(\mathcal{M})$ is UCA. As before, a negative witness should be provided in this case. Note that $\mathcal{U}$ is required to be integer when u is integer. As proven in Theorem I.1, such an integer model exists when u-Rep is solvable. The *bounded representation* (BoundRep) problem is a slight variation of u-Rep in which a feasible $d > 0$ must be found by the algorithm, as it is not given as input. That is, we are given a PCA model $\mathcal{M} = (C, \mathcal{A})$ together with $c, \ell \in \mathbb{Q}_{>0}$, $d_s \in \mathbb{Q}_{\geq 0}$ and $d_\ell, d_r \colon \mathcal{A} \to \mathbb{Q}_{\geq 0}$, and we have to find a u-CA model equivalent to $\mathcal{M}$ for some UCA descriptor $u = (c, \ell, d, d_s, d_\ell, d_r)$ with $d \in \mathbb{Q}_{>0}$. Note that BoundRep admits a solution if and only if u-Rep is solvable for $d = \frac{1}{B}$, where $B = \prod_{i=1}^{k} b_i$ for the bounds $\frac{a_1}{b_1}, \ldots, \frac{a_k}{b_k}$ of d_ℓ and d_r. Unlike u-Rep, some instances of BoundRep could admit only non-integer solutions (e.g., the path on 3-vertices admits a $(2, 1, \frac{1}{3}, 0)$-CA model but no integer $(2, 1)$-CA models). Thus, it makes sense to study the *integer bounded representation* (IntBoundRep) problem in which all the input values are integers and the output model must be integer as well. As before, IntBoundRep admits a solution if and only if u-Rep is solvable for $d = 1$. We also study the MinUIG, IntMinUCA, MinP$_q^k$, and MinC$_q^k$ problems that are related to *minimal* models. We postpone their definitions to Sections 5 and 6.

2.1 Restrictions on the input models

As it is customary in the literature, in this work we assume that all the arcs of a PCA model $\mathcal{M}$ are open and no two extremes of $\mathcal{M}$ coincide. Also, by definition, no pair of arcs of $\mathcal{M}$ cover the whole circle. For technical reasons, we also assume that $\mathcal{M}$ is not trivial; if $A_1 < \ldots A_n$ are the arcs of $\mathcal{M}$, then $\mathcal{M}$ is *trivial* when either:

1. $s(A_n) < t(A_1)$, or
2. $s(A_i)t(A_i)$ are consecutive for some $1 \leq i \leq n$.

The reasons behind these assumptions are thoroughly explained in Section I.1. In short, all the considered problems are trivial when the input model either is trivial or has two arcs covering the circle, while all the arguments in this work can be easily adapted when PCA models with closed arcs, or whose extremes could coincide, are considered.

2.2 What is linear time/space for PCA models?

As discussed in [19], every PCA model $\mathcal{M}$ can be encoded with $O(n)$ bits, n being the number of arcs in $\mathcal{M}$. Thus, in theory, an algorithm on $\mathcal{M}$ is linear when it applies $O(n)$ operations on bits. However, it is a common practice to assume that $\mathcal{M}$ is implemented with $\Theta(n)$ pointers in such a way that the extremes of an arc can be obtained in $O(1)$ time when the other extreme is given (see [19]). Following this tradition, we state that an algorithm is linear when it performs $O(n)$ operations on pointers of size $\Theta(\log n)$.

3 The bounded representation problems

Pirlot introduced the synthetic graph of a PIG model [16, 17] to represent the separation constraints of its extremes in any equivalent UIG representation. In Section I.3 we extended them to PCA models to reflect the separation constraints in any equivalent UCA model. In this section we solve different flavors of the bounded representation problem by taking advantage of our generalized synthetic graphs.

Let $\mathcal{M} = (C, \mathcal{A})$ be a PCA model with arcs $A_1 < \ldots < A_n$. The *bounded synthetic graph* of $\mathcal{M}$ is the 4-digraph $\mathcal{B}(\mathcal{M})$ (see Figure 1) that has a vertex $v(A_i)$ for each $A_i \in \mathcal{A}$ and a vertex A_0, and whose bag of edges is $E_\sigma \cup E_\nu \cup E_\eta \cup E_\beta$, where:

- $E_\sigma = \{v(A_i) \to v(A_{i+1}) \mid 1 \leq i \leq n,\ A_{n+1} = A_1\}$,
- $E_\nu = \{v(A_i) \to v(A_j) \mid t(A_i)s(A_j)$ are consecutive in $\mathcal{M}\}$,
- $E_\eta = \{v(A_i) \to v(A_j) \mid s(A_i)t(A_j)$ are consecutive in $\mathcal{M}\}$, and
- $E_\beta = \{A_0 \to v(A_i), v(A_i) \to A_0 \mid 1 \leq i \leq n\}$.

The edges in E_σ, E_ν, E_η, and E_β are the *steps*, *noses*, *hollows*, and *bounds* of $\mathcal{B}(\mathcal{M})$, respectively. (We remark that E_σ, E_ν and E_η could have a nonempty intersection. However, $\mathcal{B}(\mathcal{M})$ has no loops as $\mathcal{M}$ is not trivial.) For the sake of simplicity, we drop the parameter $\mathcal{M}$ from $\mathcal{B}(\mathcal{M})$ when no ambiguities are possible. Moreover, we regard the arcs of $\mathcal{M}$ as being the vertices of $\mathcal{B}$, thus we may say that $A_i \to A_j$ is a nose instead of writing that $v(A_i) \to v(A_j)$ is a nose.

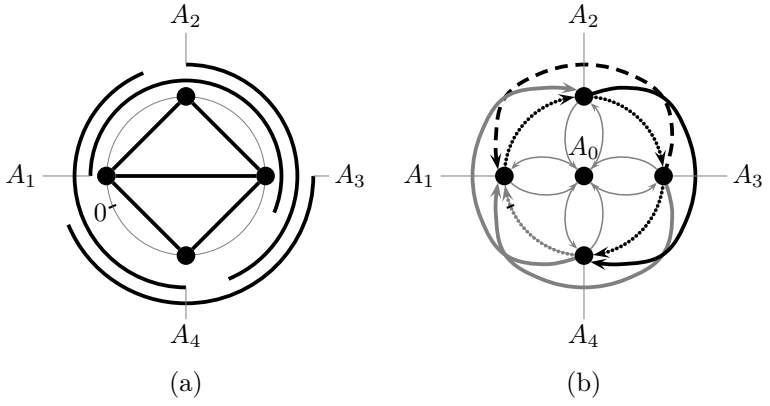


FIGURE 1. *(a) A PCA model $\mathcal{M}$ and the graph $G(\mathcal{M})$ it represents. (b) The synthetic graph of $\mathcal{M}$ where thick solid, dashed, dotted, and thin solid lines represent noses, hollows, steps, and bounds, respectively. Internal edges are black, whereas external edges are gray.*

A step (resp. nose) $A_i \to A_j$ is *internal* when $i < j$, while a hollow is *internal* when $i > j$. Non-internal edges are referred to as *external*; in particular, all the bounds are external. In other words, a step (resp. nose) $A_i \to A_j$ is external if and only if $(s(A_i), s(A_j))$ crosses 0, while a hollow $A_i \to A_j$ is external if and only if $(s(A_j), s(A_i))$ crosses 0.

Each UCA descriptor u implies an edge weighing sep_u of $\mathcal{B}$ whose purpose is to indicate how far or close $s(A_i)$ and $s(A_j)$ must be in any u-CA model equivalent to $\mathcal{M}$, for every edge $A_i \to A_j$ of $\mathcal{B}$. The edge weighing sep_u is such that, for every $1 \leq i, j \leq n$:

(sep_1) $\mathrm{sep}_u(A_i \to A_j) = d + d_s - cq$ if $A_i \to A_j$ is a step,

(sep_2) $\mathrm{sep}_u(A_i \to A_j) = d + \ell - cq$ if $A_i \to A_j$ is a nose,

(sep_3) $\mathrm{sep}_u(A_i \to A_j) = d + cq - \ell$ if $A_i \to A_j$ is a hollow, and

(sep_4) $\mathrm{sep}_u(A_0 \to A_i) = d_\ell(A_i)$ and $\mathrm{sep}_u(A_i \to A_0) = d_r(A_i) - c$,

where $q \in \{0, 1\}$ equals 0 if and only if $A_i \to A_j$ is internal. In turn, the weighing sep_u can be used to define the u-CA model $\mathcal{U}(\mathcal{M}, u)$ with arcs $U_1, \ldots, U_n$ such that $s(U_i) = \mathbf{d}\mathrm{sep}_u(A_0, A_i)$, for every $1 \leq i \leq n$ (we assume arithmetic modulo c). For the sake of notation, we omit the subscript u from sep, and the parameters $\mathcal{M}$ and u from $\mathcal{U}$, when no ambiguities are possible. The following theorem is the main result in Section I.3 that generalizes [16, Proposition 2.5] and [9, Proposition 4.4] which consider only PIG graphs.

Theorem I.1 *The following statements are equivalent for a PCA model $\mathcal{M}$ with arcs $A_1 < \ldots < A_n$ and a (an integer) UCA descriptor u:*

(i) $\mathcal{M}$ is equivalent to a u-CA model.

(ii) $\mathrm{sep}(\mathcal{W}) \leq 0$ *for every cycle $\mathcal{W}$ of $\mathcal{B}$.*

(iii) $\mathcal{U}$ is a (an integer) u-CA model equivalent to $\mathcal{M}$.

Though simple enough, Theorem I.1 allows us to solve u-Rep as follows. First, we build the 4-digraph $\mathcal{B}$ in which every edge $A_i \to A_j$ is weighed with $s_{ij} = \mathrm{sep}_u(A_i \to A_j)$. Then, we invoke the Bellman-Ford shortest path algorithm [1] on $\mathcal{B}$ to obtain $s_i = \mathbf{d}^*\mathrm{sep}(A_0, A_i)$ for every $0 \leq i \leq n$. If Bellman-Ford ends in success, then we output $\mathcal{U}(\mathcal{M}, u)$; otherwise, we output the cycle of positive weight found as the negative witness. Recall that Bellman-Ford ends in success if and only if $\mathcal{B}$ has no cycle of positive weight, in which case $\mathbf{d}\mathrm{sep} = \mathbf{d}^*\mathrm{sep}$ and, so, $\mathcal{U}(\mathcal{M}, u)$ is well defined.

Bellman-Ford computes each value $s_i = s_i^n$ in an iterative manner. At iteration k, the value of s_i is updated to $s_i^k = \max\{s_j^{k-1} + s_{ji} \mid A_j \to A_i \in E(\mathcal{B})\}$ for every $0 \leq i \leq n$. As $\mathcal{B}$ has $O(n)$ edges, a total of $O(n^2)$ arithmetic operations are performed. By (sep_1)–(sep_4), $s_i^k \in \mathbb{N}$ when u is integer, thus $O(1)$ time is required by each operation. However, as it was noted by Klavík et al. [9], we cannot assume $O(1)$ time per operation when u is non-integer.

The inconvenience is that, to compare two fractional values a_1/b_1 and a_2/b_2, we have to multiply them with a common multiple of b_1 and b_2. Thus, a priori, the number of bits used to represent s_j^{k-1} could be large, and the operations required to compute s_i^k could take more than constant time.

It turns out that we can represent s_i^k with $O(1)$ words in a simple manner. The idea is to use a *distance tuple* $t = \langle b, [c], [\ell], [d], [d_s] \rangle$ with $b \in \{d_\ell(A), d_r(A) \mid A \in \mathcal{A}\} \cup \{-\infty\}$, $[c], [\ell] \in \mathbb{Z}$, and $[d], [d_s] \in \mathbb{N}$, in order to represent the rational

$$[t] = b + [c]c + [\ell]\ell + [d]d + [d_s]d_s.$$

So, for instance, we can represent $s_{ij} = \text{sep}(A_i \to A_j)$ as in the following table, where $q \in \{0, 1\}$ equals 0 if and only if $A_i \to A_j$ is internal.

Type	b	$[c]$	$[\ell]$	$[d]$	$[d_s]$
Step	0	$-q$	0	1	1
Nose	0	$-q$	1	1	0
Hollow	0	q	-1	1	0
Bound ($i = 0$)	$d_\ell(A_j)$	0	0	0	0
Bound ($j = 0$)	$d_r(A_i)$	-1	0	0	0

Analogously, we implement s_i^k using a distance tuple for every $0 \leq i, k \leq n$. Just note that if s_0^k is ever updated, then $\mathcal{B}$ has a cycle of positive weight, thus we can immediately halt Bellman-Ford in failure. By doing so we observe, by invariant, that $s_i^k = [t]$ for every iteration k and some distance tuple $t = \langle b, [c], [\ell], [d], [d_s] \rangle$ in which $-k \leq [c], [\ell] \leq k$, $0 \leq [d], [d_s] \leq k$. Figure 2 depicts an execution of Bellman-Ford in which every weight is implemented using a distance tuple.

With the above implementation, each arithmetic operation performed by Bellman-Ford costs $O(1)$ time, as it involves only $O(1)$ of the input values. (We emphasize that using distance tuples in a real implementation is not required. The advantage of distance tuples is that they allow us to conclude that each fractional value can be represented with $O(1)$ words. Of course, this is true even when a traditional implementation of fractional values is employed.) We conclude, therefore, that u-Rep can be solved in $O(n^2)$ time, even when u is non-integer. As far as our knowledge extends, this is the first polynomial algorithm to solve this problem.

By definition, (Int)BoundRep is solvable if and only if u-Rep is solvable for some (integer) UCA descriptor u. The main difference between both problems is that d is an input of u-Rep whereas a feasible d must be found by BoundRep. A simple solution for (Int)BoundRep is to invoke the above algorithm with a small enough value of d. For instance, if $a_1/b_1, \ldots, a_k/b_k$ are the bounds of d_ℓ and d_r, then we can take $d = \frac{1}{B}$ where $B = \prod_{i=1}^{k} b_i$. In other words, we transform every weight of sep into an integer before invoking Bellman-Ford in the algorithm above. This algorithm is efficient when d consumes $O(1)$ bits, e.g., when u is integer. But, it is not efficient in the general case because to compare two distance tuples we need to operate with d.

Weights of the edges implemented with distance tuples

Edge	sep_u	$\langle\ b\ ,[c],[\ell],[d],[d_s]\ \rangle$	Edge	sep_u	$\langle\ b,[c],[\ell],[d],[d_s]\ \rangle$
$A_0 \xrightarrow{\beta} A_1$	$\frac{3}{2}$	$\langle\ \frac{3}{2}\ ,0\ ,0\ ,0\ ,\ 0\ \rangle$	$A_1 \xrightarrow{\sigma} A_2$	$\frac{14}{3}$	$\langle 0,0\ ,0\ ,1\ ,\ 1\ \rangle$
$A_0 \xrightarrow{\beta} A_2$	$\frac{9}{4}$	$\langle\ \frac{9}{4}\ ,0\ ,0\ ,0\ ,\ 0\ \rangle$	$A_2 \xrightarrow{\sigma} A_3$	$\frac{14}{3}$	$\langle 0,0\ ,0\ ,1\ ,\ 1\ \rangle$
$A_0 \xrightarrow{\beta} A_3$	$\frac{27}{8}$	$\langle\ \frac{27}{8},0\ ,0\ ,0\ ,\ 0\ \rangle$	$A_3 \xrightarrow{\sigma} A_4$	$\frac{14}{3}$	$\langle 0,0\ ,0\ ,1\ ,\ 1\ \rangle$
$A_0 \xrightarrow{\beta} A_4$	$\frac{81}{16}$	$\langle\ \frac{81}{16},0\ ,0\ ,0\ ,\ 0\ \rangle$	$A_4 \xrightarrow{\sigma} A_1$	$\frac{-245}{6}$	$\langle 0,\text{-}1\ ,0\ ,1\ ,\ 1\ \rangle$
$A_1 \xrightarrow{\beta} A_0$	$\frac{-647}{16}$	$\langle\ \frac{81}{16},\text{-}1\ ,0\ ,0\ ,\ 0\ \rangle$	$A_2 \xrightarrow{\nu} A_4$	$\frac{125}{6}$	$\langle 0,0\ ,1\ ,1\ ,\ 0\ \rangle$
$A_2 \xrightarrow{\beta} A_0$	$\frac{-337}{8}$	$\langle\ \frac{27}{8},\text{-}1\ ,0\ ,0\ ,\ 0\ \rangle$	$A_3 \xrightarrow{\eta} A_1$	$\frac{-73}{6}$	$\langle 0,0\ ,\text{-}1\ ,1\ ,\ 0\ \rangle$
$A_3 \xrightarrow{\beta} A_0$	$\frac{-173}{4}$	$\langle\ \frac{9}{4}\ ,\text{-}1\ ,0\ ,0\ ,\ 0\ \rangle$	$A_3 \xrightarrow{\nu} A_1$	$\frac{-74}{3}$	$\langle 0,\text{-}1\ ,1\ ,1\ ,\ 0\ \rangle$
$A_4 \xrightarrow{\beta} A_0$	-44	$\langle\ \frac{3}{2}\ ,\text{-}1\ ,0\ ,0\ ,\ 0\ \rangle$	$A_4 \xrightarrow{\nu} A_2$	$\frac{-74}{3}$	$\langle 0,\text{-}1\ ,1\ ,1\ ,\ 0\ \rangle$

$\xrightarrow{x}$ is used to mean that an edge belongs to E_x.

Weights of the vertices implemented with distance tuples

	After Step 1	**After Step 2**	**After Step 3**	
Vertex	$\langle\ b\ ,[c],[\ell],[d],[d_s]\ \rangle$	$\langle\ b,[c],[\ell],[d],[d_s]\ \rangle$	$\langle\ b,[c],[\ell],[d],[d_s]\ \rangle$	s^n
A_0	$\langle\ 0\ ,0\ ,0\ ,0\ ,\ 0\ \rangle$	$\langle 0,0\ ,0\ ,0\ ,\ 0\ \rangle$	$\langle 0,0\ ,0\ ,0\ ,\ 0\ \rangle$	0
A_1	$\langle\ \frac{3}{2}\ ,0\ ,0\ ,0\ ,\ 0\ \rangle$	$\langle\ \frac{3}{2},0\ ,0\ ,0\ ,\ 0\ \rangle$	$\langle\ \frac{3}{2},0\ ,0\ ,0\ ,\ 0\ \rangle$	$\frac{3}{2}$
A_2	$\langle\ \frac{9}{4}\ ,0\ ,0\ ,0\ ,\ 0\ \rangle$	$\langle\ \frac{3}{2},0\ ,0\ ,1\ ,\ 1\ \rangle$	$\langle\ \frac{3}{2},0\ ,0\ ,1\ ,\ 1\ \rangle$	$\frac{37}{6}$
A_3	$\langle\ \frac{27}{8},0\ ,0\ ,0\ ,\ 0\ \rangle$	$\langle\ \frac{9}{4},0\ ,0\ ,1\ ,\ 1\ \rangle$	$\langle\ \frac{3}{2},0\ ,0\ ,2\ ,\ 2\ \rangle$	$\frac{65}{6}$
A_4	$\langle\ \frac{81}{16},0\ ,0\ ,0\ ,\ 0\ \rangle$	$\langle\ \frac{9}{4},0\ ,1\ ,1\ ,\ 0\ \rangle$	$\langle\ \frac{3}{2},0\ ,1\ ,2\ ,\ 1\ \rangle$	27

FIGURE 2. *Execution of Bellman-Ford on the synthetic graph of Figure 1 weighed with* sep_u*, where:*

$$u = \left\langle c\colon \frac{91}{2}, \ell\colon \frac{33}{2}, d\colon \frac{13}{3}, d_s\colon \frac{1}{3}, d_\ell\colon A_i \to \left(\frac{3}{2}\right)^i, d_r\colon A_i \to \left(\frac{3}{2}\right)^{5-i} \right\rangle$$

.

An alternative solution for BOUNDREP is to find any $d < d_*$, where d_* is the maximum such that $\mathcal{M}$ is equivalent to a $(c, \ell, d_*, d_s, d_\ell, d_r)$-CA model. To find d we invoke the algorithm for u-REP, but instead of giving an input number for d, we just think of d as a placeholder for a value lower than or equal to d_*. As before, $s_{ij} = \text{sep}(A_i \to A_j)$ and s_i^k are encoded with distance tuples $[t_{ij}]$ and $[t_i]$, respectively. However, we re-implement the comparison operator to cope with the fact that d is an indeterminate value.

Let $[d_i]$ and $[d_{ij}]$ be the coefficients of t_i and t_{ij} that multiply d, respectively. For a distance tuple $t = \langle b, [c], [\ell], [d], [d_s]\rangle$, let

$$\|t\| = [t] - [d]d = b + [c]c + [\ell]\ell + [d_s]d_s.$$

The main observation is that $\mathbf{d}^*\text{sep}(A_0, A_i) < \mathbf{d}^*\text{sep}(A_0, A_j) + \text{sep}(A_j \to A_i)$ if

and only if

- $\|t_i\| < \|t_{ji}\| + \|s_j\|$, or
- $\|t_i\| = \|t_{ji}\| + \|s_j\|$ and $[d_i] < [d_j] + [d_{ji}]$.

Hence, every arithmetic operation —including comparisons— costs $O(1)$ time, as it involves only $O(1)$ input values.

If Bellman-Ford ends in success, then a value d such that

$$\min\{\mathbf{d}^*\text{sep}(A_0, A_i) - \mathbf{d}^*\text{sep}(A_0, A_j) - \text{sep}(A_j \to A_i) \mid A_j \to A_i \in E(\mathcal{B})\} =$$
$$\min\{[t_i] - [t_j] - [t_{ij}] \mid A_i \to A_j \in E(\mathcal{B})\} =$$
$$\min\{\|t_i\| + d[d_i] - \|t_j\| - d[d_j] - \|t_{ij}\| - d[d_{ij}] \mid A_i \to A_j \in E(\mathcal{B})\} > 0$$

can be obtained in $O(n)$ time. By Theorem I.1, the algorithm is correct as $\text{sep}(\mathcal{W}) \leq 0$ for every cycle $\mathcal{W}$ of $\mathcal{B}$. As for the certification problem, note that d consumes $O(1)$ bits, thus we can output $\mathcal{U}(\mathcal{M})$ in $O(n)$ time. Moreover, any cycle of positive weight found by Bellman-Ford can be used for negative certification.

Theorem 1 BoundRep, IntBoundRep, *and* u-Rep *can be solved in* $O(n^2)$ *time and* $O(n)$ *space.*

4 The unrestricted representation problem

In 1974, Tucker showed a characterization by forbidden subgraphs of those PCA graphs that are UCA [22]. His proof yields an effective method to transform a PCA model $\mathcal{M}$ into an equivalent UCA model $\mathcal{U}$. Unfortunately, the extremes of $\mathcal{U}$ are not guarantied to be of a polynomial size and, thus, the corresponding representation algorithm cannot be regarded as polynomial. The representation problem remained unsolved until 2008, when Lin and Szwarcfiter designed the algorithm LS that transforms any PCA model into an equivalent UCA model in linear time [13]. The LS algorithm, however, gives no output when the input graph is not UCA. A year later, Kaplan and Nussbaum [7] devised the algorithm KN that solves the problem of finding a forbidden induced subgraph in linear time.

The pair (LS, KN) can be regarded as a certifying algorithm. The advantage of certifying algorithms [14] over their non-certifying counterparts is that they provide a *witness* guaranteeing the validity of the YES-NO answer. The end user can *authenticate* the witness to be confident that the answer is correct, even in the presence of an incorrect implementation. However, an erroneous implementation of LS could claim that a UCA graph G admits no UCA models, while a correct implementation of KN claims that G is UCA. In such a case, no witness is obtained at all, defeating the purpose of a certifying algorithm. Is for this reason that Kaplan and Nussbaum [7] leave open the problem of finding a *unified* certifying algorithm. Regarding the space complexity, Köbler et al. [10] mention that the representation problem in logspace is open as well.

In this section we design two unified certifying algorithms for the representation problem; one runs in linear time and the other in logspace. Our algorithms depend on Theorem I.2 that brings Tucker's characterization to the framework of synthetic graphs. One of the main features of Theorem I.2 is that it exhibits other equivalences that can be used for positive and negative certification. In particular, it shows how to obtain an integer (c, ℓ)-CA model equivalent to $\mathcal{M}$ with c and ℓ polynomial in n.

4.1 Efficient Tucker's characterization

The cycles of $\mathcal{B}$ with maximum sep-values play a fundamental role when deciding if $\mathcal{M}$ admits an equivalent u-CA model. The key idea of the representation algorithm is to take $\mathcal{U}^* = \mathcal{U}(c^*, \ell^*)$ as in Theorem I.1 for some appropriate values c^* and ℓ^*. There are two reasons why c^* and ℓ^* are appropriate: first, $\mathcal{U}^*$ is a UCA model equivalent to $\mathcal{M}$ whenever $G(\mathcal{M})$ is UCA; second, even though the best algorithm we know requires $O(n^2)$ time to compute $\mathcal{U}(c, \ell)$ for general values of c and ℓ, we can compute $\mathcal{U}^*$ in $O(n)$ time. Before we define c^* and ℓ^*, we need to introduce some concepts to describe the sep-values of the boundless synthetic graph. The *(boundless) synthetic graph* of $\mathcal{M}$ is just $\mathcal{S}(\mathcal{M}) = \mathcal{B}(\mathcal{M}) \setminus A_0$; for the sake of simplicity, we drop the parameter $\mathcal{M}$ as usual. (Note that $\mathcal{S}$ is a 3-digraph as it contains no bounds.)

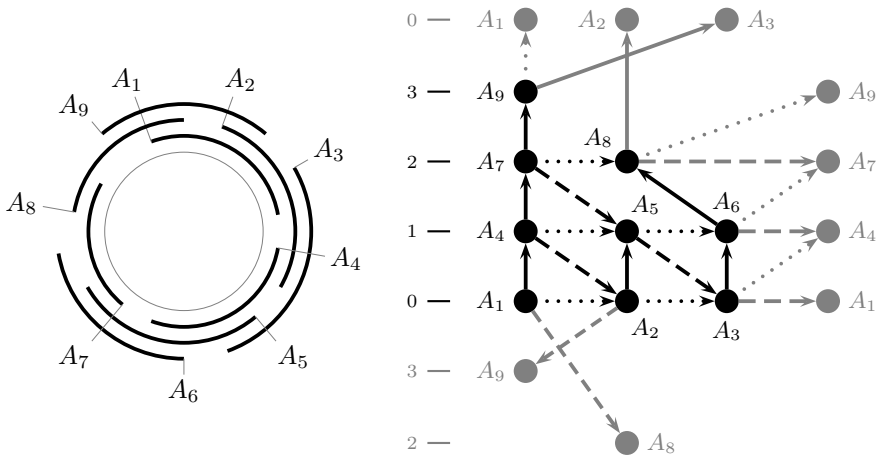
Let $\mathcal{M} = (C, \mathcal{A})$ be a PCA model with arcs $A_1 < \ldots < A_n$. The *height* $h(A_i)$ of A_i $(1 \leq i \leq n)$ is recursively defined as follows:

$$h(A_i) = \begin{cases} 0 & \text{if } s(A_i) < t(A_1) \\ 1 + h(A_j) & \text{otherwise, where } A_j = \max\{A_j \mid t(A_j) < s(A_i)\}. \end{cases}$$

The *height* of $\mathcal{M}$ is defined as $h(\mathcal{M}) = h(A_n)$; note that $h(\mathcal{M}) \geq 1$ (because $\mathcal{M}$ is not trivial). For the sake of notation, we drop the parameter $\mathcal{M}$ as usual. Figure 3 depicts the synthetic graph of a PCA model with the vertices drawn in levels according to their height.

Following Section I.3.3, we refer to a nose (resp. step, hollow) $A_i \to A_j$ as being a δ-nose (resp. δ-step, δ-hollow) to indicate that $h(A_j) - h(A_i) = \delta$. For every walk $\mathcal{W}$ of $\mathcal{S}$, we write $\nu_\delta(\mathcal{W})$, $\eta_\delta(\mathcal{W})$, and $\sigma_\delta(\mathcal{W})$ to indicate the number of δ-noses, δ-hollows, and δ-steps of $\mathcal{W}$, respectively. As usual, we do not write the parameter $\mathcal{W}$ when it is clear from context.

It is not hard to see (check Figure 3 and Section I.3.3) that $\mathcal{S}$ has three kinds of noses —namely 1-, $(-h)$-, and $(1-h)$-noses—, three kinds of steps —namely 0-, 1-, and $(-h)$-steps—, and four kinds of hollows —namely 0-, (-1)-, h-, and external $(h-1)$-hollows. A priori, we need to differentiate between internal 0-hollows and external $(h-1)$-hollows when $h = 1$. However, we will refer to $A_i \to A_j$ as being an $(h-1)$-*hollow* to mean that $A_i \to A_j$ is an **external** $(h-1)$-hollow; no confusions are possible because $\mathcal{M}$ has no external 0-hollows when $h = 1$ (otherwise A_i and A_j would cover the circle of $\mathcal{M}$). Note that $h(A_j) = 0$ when $A_i \to A_j$ is a $(-h)$- or $(1-h)$-nose; $h(A_i) = 0$ when $A_i \to A_j$ is a h- or $(h-1)$-hollow; and $h(A_1) = 0$ for the unique $(-h)$-step $A_n \to A_1$.



(a) A PCA model $\mathcal{M}$. **(b)** The synthetic graph $\mathcal{S}(\mathcal{M})$.

FIGURE 3. *(Boundless) synthetic graph $\mathcal{S}$ of a PCA model $\mathcal{M}$. Each gray vertex corresponds to a black vertex (we separate them for the sake of exposition) and each edge is drawn only once. The edges are solid, dashed, and dotted, according to whether they are noses, hollows, and steps, respectively. The height of $\mathcal{M}$ is $h = 3$ and each vertex is drawn in a row that corresponds to its height; the height is indicated to the left. Note that there are* 1*-,* $(-h)$*-, and* $(1-h)$*-noses,* 0*-,* (-1)*-,* h*-, and* $(h-1)$*-hollows, and* 0*-,* 1*-, and* $(-h)$*-steps.*

"Nose" and "hollow" walks are of particular interest for two coincident reasons. On the one hand, they correspond to what Tucker calls by the names of (a, b)-*independents* and (x, y)-*circuits* (see Section I.5). On the other hand, as it happens with (a, b)-independents and (x, y)-circuits, their "ratios" impose lower and upper bounds on the length of any UCA model equivalent to $\mathcal{M}$. A walk of $\mathcal{S}$ is a *nose walk* when it contains no hollows, while it is a *hollow walk* when it contains no noses and $\eta_h + \eta_{h-1} \geq \sigma_{-h}$. Note that a walk is both a nose and a hollow walk only if all its edges are steps; in general, walks that contain only steps are referred to as *step walks*.

The *ratio* of nose walk $\mathcal{W}_N$ is the value $r(\mathcal{W}_N) = \frac{\nu_{-h} - \sigma_1}{\nu_{1-h} + \nu_{-h} + \sigma_{-h}}$, while the *nose ratio* of $\mathcal{M}$ is $r(\mathcal{M}) = \max\{r(\mathcal{W}_N) \mid \mathcal{W}_N \text{ is a nose cycle of } \mathcal{M}\}$. Similarly, $R(\mathcal{W}_H) = \frac{\eta_0 + \eta_h + \sigma_1}{\eta_h + \eta_{h-1} - \sigma_{-h}}$ is the *ratio* of a hollow walk $\mathcal{W}_H$, while $R(\mathcal{M}) = \min\{R(\mathcal{W}_H) \mid \mathcal{W}_H \text{ is a hollow cycle of } \mathcal{M}\}$ is the *hollow ratio* of $\mathcal{M}$. The following observation describes the bounds on c implied by both ratios; note that, as usual, we omit the parameter $\mathcal{M}$ from r and R.

Lemma I.2 *For every u-CA model,*

$$(\ell + d)(h + r) \leq c \leq (\ell - d)(h + R) \tag{1}$$

By (1), $\mathcal{M}$ is equivalent to a u-CA model only if $c = (\ell + d)(h + r) + e$ for some $e \geq 0$. In general, $e = c - (\ell + d)(h + r)$ is referred to as the *extra space*.

The next lemma describes the value of $\mathrm{sep}(\mathcal{W})$ in terms of ℓ and e, for every walk $\mathcal{W}$ of $\mathcal{S}$.

Lemma 1 (see (5)–(8), Ch. I) *For every CA descriptor u,*

$$\mathrm{sep}(\mathcal{W}) = (\ell + d)(h(A_j) - h(A_i) + \mathrm{len}(\mathcal{W})) + e\,\mathrm{ext}(\mathcal{W}) + \mathrm{const}(\mathcal{W}, d, d_s) \quad (2)$$

where:

$$\mathrm{len}(\mathcal{W}) = \nu_{-h} - \eta_h - \sigma_1 - \eta_0 + r\,\mathrm{ext}(\mathcal{W}) \quad (3)$$

$$\mathrm{ext}(\mathcal{W}) = \eta_h + \eta_{h-1} - \nu_{-h} - \nu_{1-h} - \sigma_{-h}, \text{ and} \quad (4)$$

$$\mathrm{const}(\mathcal{W}, d, d_s) = 2d(\eta_{-1} + \eta_0 + \eta_h + \eta_{h-1}) + (d + d_s)(\sigma_1 + \sigma_0 + \sigma_{-h}). \quad (5)$$

The values $\mathrm{len}(\mathcal{W})$, $\mathrm{ext}(\mathcal{W})$, and $\mathrm{const}(\mathcal{W}, d, d_s)$ are the *length*, *extra*, and *constant factors* of $\mathcal{W}$, and $\mathcal{W}$, d, and d_s are omitted as usual. One of the advantages of expressing sep as a polynomial with indeterminates ℓ and e and coefficients len, ext and const is that, obviously, the factors depend only on the structure of $\mathcal{S}$ and not on the weighing function sep. Thus, $\mathrm{ext}(\mathcal{W})$ is the weight of $\mathcal{W}$ when the following edge weighing ext (the overloaded notation is intentional) of $\mathcal{S}$ is considered:

$$\mathrm{ext}(A_i \to A_j) = \begin{cases} 1 & \text{if } A_i \to A_j \text{ is an external hollow} \\ -1 & \text{if } A_i \to A_j \text{ is an external nose or step} \\ 0 & \text{otherwise} \end{cases}$$

We can compute $\mathrm{len}(\mathcal{W})$ and $\mathrm{const}(\mathcal{W}, d, d_s)$ in a similar fashion with the corresponding edge weighings len and const_{d,d_s}.

Recall that our goal is to find $\mathcal{U}^* = \mathcal{U}(c^*, \ell^*)$ for some appropriate values c^* and ℓ^* for which we can assure that $\mathcal{U}^*$ is a UCA model equivalent to $\mathcal{M}$ when $G(\mathcal{M})$ is UCA. By Theorem I.1, c^* and ℓ^* must be such that $\mathrm{sep}(\mathcal{W}) \leq 0$ for every cycle $\mathcal{W}$ of $\mathcal{S}$. In particular, sep must be non-positive for nose and hollow cycles, which impose the lower and upper bounds described by (1), respectively. The reason to consider only nose and hollow cycles is that they have the largest sep-values when c^* and ℓ^* are large enough (see Chapter I).

Lemma I.3 *For any walk $\mathcal{W}$ of $\mathcal{S}$ there exists either a nose or hollow walk $\mathcal{W}'$ of $\mathcal{S}$ starting and ending at the same vertices as $\mathcal{W}$ such that* $\mathrm{len}(\mathcal{W}) \leq \mathrm{len}(\mathcal{W}')$ *and* $\mathrm{ext}(\mathcal{W}) \leq \mathrm{ext}(\mathcal{W}')$.

Our goal is not only to compute $\mathcal{U}^*$, but to do it fast. The invocation to Bellman-Ford is one of the bottlenecks in our algorithm to compute $\mathcal{U}(c, \ell)$. The reason why Bellman-Ford is applied resides in the fact that $\mathcal{S}$ is not an acyclic graph. The advantage of c^* and ℓ^* is that we can remove all the cycles of $\mathcal{S}$ without affecting the distances from A_1. Thus, we can avoid Bellman-Ford and, consequently, we decrease the time complexity to $O(n)$. With this in mind, say that an edge $A_i \to A_j$ of $\mathcal{S}$ is *redundant* when either

(red$_1$) $\mathbf{d}\text{len}(A_1, A_j) > \mathbf{d}\text{len}(A_1, A_i) + \text{len}(A_i \to A_j)$, or

(red$_2$) $\mathbf{d}\text{len}(A_1, A_j) = \mathbf{d}\text{len}(A_1, A_i) + \text{len}(A_i \to A_j)$ and
$(\mathbf{d}\text{ext} \circ \mathbf{d}\text{len})(A_1, A_j) > (\mathbf{d}\text{ext} \circ \mathbf{d}\text{len})(A_1, A_i) + \text{len}(A_i \to A_j)$.

Roughly speaking, $A_i \to A_j$ is redundant when it plays no role in the separation between $s(A_1) = 0$ and $s(A_j)$ for large values of ℓ and not-so-large values of e. (Recall that $(\mathbf{d}\text{ext} \circ \mathbf{d}\text{len})$ is the ext-distance restricted only to those paths with maximum len-distance.) The *reduction* of $\mathcal{S}(\mathcal{M})$ is the 3-digraph $\mathcal{R}(\mathcal{M})$ obtained after removing all the redundant edges of $\mathcal{S}(\mathcal{M})$; as usual, we omit the parameter $\mathcal{M}$. Theorem I.2 includes Tucker's characterization as equivalence (i) $\Leftrightarrow$ (ii).

Theorem I.2 *Let $\mathcal{M}$ be a PCA model with arcs $A_1 < \ldots < A_n$, and $r_1, r_2 \in \mathbb{N}$ be such that $r = r_1/r_2$. Then, the following statements are equivalent:*

(i) *$\mathcal{M}$ is equivalent to a UCA model.*

(ii) *$r < R$.*

(iii) *$\text{len}(\mathcal{W}) < 0$ for every hollow cycle $\mathcal{W}$ of $\mathcal{S}$.*

(iv) *either $\text{len}(\mathcal{W}) < 0$ or $\text{len}(\mathcal{W}) = 0$ and $\text{ext}(\mathcal{W}) < 0$, for each cycle $\mathcal{W}$ of $\mathcal{S}$.*

(v) *$\mathcal{R}$ is acyclic.*

(vi) *$\mathbf{d}^*\text{sep}_{(c^*,\ell^*)}(\mathcal{B}, A_0, A_i) = \mathbf{d}\text{sep}_{(c^*,\ell^*)}(\mathcal{R}, A_1, A_i)$ for every $1 \leq i \leq n$, where $c^* = (\ell^* + 1)(h + r) + e$, $(\ell^* + 1) = r_2 e^2$, and $e = 4n$.*

(vii) *$\mathcal{U}(c^*, \ell^*)$ is an integer (c^*, ℓ^*)-CA model equivalent to $\mathcal{M}$ for c^* and ℓ^* as in (vi).*

We can already foresee the algorithmic consequences on Rep that Theorem I.2 has when combined with Theorem I.1. For any input PCA model $\mathcal{M}$, we solve u-Rep for the UCA descriptor $u = (c^*, \ell^*)$ implied by statement (vi). As a byproduct, we either obtain a UCA model equivalent to $\mathcal{M}$ or a cycle of $\mathcal{S}$ that can be used for negative certification. The algorithm costs $O(n^2)$ time, plus the time and space required so as to compute $r(\mathcal{M})$. In Section 4.3 we show an $O(n)$ time variant of this algorithm, taking advantage of the reduction of $\mathcal{S}$. However, we first discuss how r can be found.

4.2 The algorithm by Kaplan and Nussbaum

Translated to synthetic graphs, Tucker's characterization (equivalence (i) $\Leftrightarrow$ (ii) of Theorem I.2) states that $\mathcal{M}$ is equivalent to no UCA model if only if $\mathcal{S}$ has nose and hollow cycles $\mathcal{W}_N$ and $\mathcal{W}_H$ such that $r(\mathcal{W}_N) \geq R(\mathcal{W}_H)$. The original proof by Tucker does not show how to obtain such cycles. More than thirty years later, Durán et al. [4] described the first polynomial algorithm to compute these cycles with a rather complex implementation. A few years later, Kaplan

and Nussbaum [7] improved this algorithm to run in $O(n)$ time while simplifying the implementation. The purpose of this section is to translate the algorithm by Kaplan and Nussbaum in terms of the synthetic graph. The proof of correctness is simple, short, and rather intuitive, while the implementation is quite similar to the one given by Kaplan and Nussbaum.

The main concept of this section is that of greedy cycles. For any nose (resp. hollow) walk $\mathcal{W}_N = B_1, \ldots, B_k$ of $\mathcal{S}$, we say that B_i is *greedy (in $\mathcal{W}_N$)* when either the edge $B_i \to B_{i+1}$ of $\mathcal{W}_N$ is a nose (resp. hollow) or no nose (resp. hollow) of $\mathcal{S}$ starts at B_i. A nose (resp. hollow) cycle is *greedy* when all its vertices are greedy. In other words, $\mathcal{W}_N$ is greedy when noses (resp. hollows) are preferred over steps. The main idea of Durán et al., which was somehow implicit in [22], is to observe that $\mathcal{S}$ contains a greedy nose (resp. hollow) cycle of highest (resp. lowest) ratio. Then, they compute the unique greedy nose (resp. hollow) cycle starting at a vertex A, for every $A \in V(\mathcal{S})$, and keep the one with highest (resp. lowest) ratio. Note that each greedy nose (resp. hollow) cycle $B_1, \ldots, B_k$ is found k times, once for each starting vertex B_i. Kaplan and Nussbaum, instead, compute each greedy nose (resp. hollow) cycle only once by taking only one vertex as the starting point.

The next lemma has a new proof that $\mathcal{S}$ contains a greedy nose cycle of highest ratio.

Lemma 2 (see also [4, 7, 22]) *For any nose cycle $\mathcal{W}_N$ of $\mathcal{S}$ there exists a greedy nose cycle $\mathcal{W}'_N$ of $\mathcal{S}$ such that $r(\mathcal{W}_N) \leq r(\mathcal{W}'_N)$.*

Proof: The proof is trivial when $\mathcal{W}_N$ is greedy. When $\mathcal{W}_N$ is not greedy, we can transform it into a greedy nose cycle by traversing $\mathcal{W}_N$ from any vertex while applying the following operation when a non-greedy vertex B_1 is found, until no more non-greedy vertices remain. Let $B_1 \to B$ be the nose from B_1 and $\mathcal{W} = B_1, \ldots, B_i$ be the shortest subpath of $\mathcal{W}_N$ such that either $B_i = B$ or $B_{i-1} \to B_i$ is a nose. The operation transforms $\mathcal{W}_N$ into $\mathcal{W}'_N$ by replacing $\mathcal{W}$ with $\mathcal{W}'$, where $\mathcal{W}'$ is the path formed by the nose $B_1 \to B$ followed by the step path from B to B_i. Thus, it suffices to prove that $r(\mathcal{W}) \leq r(\mathcal{W}')$. Moreover, if we let $r_1 = \nu_{-h} - \sigma_1$ and $r_2 = \nu_{1-h} + \nu_{-h} + \sigma_{-h}$, then $r_j(\mathcal{W}'_N) = r_j(\mathcal{W}_N) - r_j(\mathcal{W}) + r_j(\mathcal{W}')$ for $j \in \{1, 2\}$. So, as $r = r_1/r_2$ by definition, it is enough to show that $r_1(\mathcal{W}') \geq r_1(\mathcal{W})$ and $r_2(\mathcal{W}') \leq r_2(\mathcal{W})$.

If $B = B_i$ then $\mathcal{W}$ has at least one 1-step or $(-h)$-step, while $\mathcal{W}' = B_1 \to B$. Thus $r_1(\mathcal{W}) \leq 0 \leq r_1(\mathcal{W}')$ and $r_2(\mathcal{W}') \leq r_2(\mathcal{W})$, and the lemma follows. Otherwise, if $B \notin \mathcal{W}$, then B_1, B_{i-1}, B, B_i appear in this order in $\mathcal{S}$ when its steps are traversed from B_1. It is not hard to see that $r_i(\mathcal{W}) = r_i(\mathcal{W}')$ when $B_1 \to B$ and $B_{i-1} \to B_i$ have equal jumps. This leaves us with only four possible combinations for the heights of B_1, B_{i-1}, B_i, and B when the jumps differ, and in all such cases the lemma is true (see the table below).

$h(B_1)$	$h(B_{i-1})$	$h(B)$	$h(B_i)$	$r_1(\mathcal{W})$	$r_1(\mathcal{W}')$	$r_2(\mathcal{W})$	$r_2(W')$
$h-2$	$h-1$	$h-1$	0	-1	-1	1	1
$h-1$	$h-1$	h	0	0	0	1	1
$h-1$	h	h or 0	0	0	0	1	1
h	0	0	1	0	0	1	1

□

The proof that $\mathcal{S}$ contains a greedy hollow cycle with lowest ratio is similar, and we omit it as it is not required by our algorithm. Moreover, an analogous proof is given in Lemma 4.

Lemma 3 (see [7] and Section I.5) *For any hollow cycle $\mathcal{W}_H$ of $\mathcal{S}$ there exists a greedy hollow cycle $\mathcal{W}'_H$ of $\mathcal{S}$ such that $R(\mathcal{W}_H) \geq R(\mathcal{W}'_H)$.*

The algorithm to compute a nose (resp. hollow) cycle with highest (resp. lowest) ratio follows easily from Lemma 2 (resp. Lemma 3). Just note that if an edge $A_i \to A_j$ belongs to a greedy nose (resp. hollow) cycle, then either $A_i \to A_j$ is a nose (resp. hollow), or there are no noses (resp. hollow) from A_i in $\mathcal{S}$. Then, $\mathcal{W}$ is a greedy nose (resp. hollow) cycle of $\mathcal{S}$ if and only if $\mathcal{W}$ is a cycle of the digraph $\mathcal{S}_N$ that is obtained by keeping only the noses (resp. hollows) of $\mathcal{S}$ and the steps that go from vertices with no noses (resp. hollows). Since all the vertices in $\mathcal{S}_N$ have out-degree 1, we can obtain all the greedy cycles in $O(n)$ time. Then, by Lemmas 2 and 3, r and R can be computed in $O(n)$ time. Furthermore, if $r \geq R$, then a nose and a hollow cycles $\mathcal{W}_N$ and $\mathcal{W}_H$ with $r(\mathcal{W}_N) = r$ and $R(\mathcal{W}_H) = R$ are obtained as a byproduct. If required, these cycles can be transformed into an (a, b)-independent and an (x, y)-circuit of $\mathcal{M}$ in $O(n)$ time, thus obtaining the same output as given by Kaplan and Nussbaum (see Section I.5).

4.3 Efficient construction of UCA models

Durán et al. [4] ask if there exists an integer (c, ℓ)-CA model equivalent to a PCA model $\mathcal{M}$ such that c and ℓ are bounded by a polynomial in n. If affirmative, they also inquire whether such a model can be found in $O(n)$ time. Both questions were affirmatively answered by Lin and Szwarcfiter [13], who showed how to reduce the problem of finding a (c, ℓ)-CA model to a circulation problem. Their algorithm and its correctness have nothing to do with Theorem I.2 and it is not easy to see how a forbidden subgraph can be obtained for certification (that is, without invoking a second recognition algorithm, as the one by Kaplan and Nussbaum). Kaplan and Nussbaum [7] ask for a *unified* certification algorithm. We provide such an algorithm in this section.

Our algorithm is based on the equivalence (i) $\Leftrightarrow$ (v) $\Leftrightarrow$ (vi) of Theorem I.2. That is, the algorithm just checks whether $\mathcal{R}(\mathcal{M})$ is acyclic. If affirmative, then $\mathcal{U}^* = \mathcal{U}(c^*, \ell^*)$ is the positive witness by Theorem I.2, for c^* and ℓ^* as in statement (vi). Otherwise, a nose cycle with ratio $r(\mathcal{M})$ combined with a

cycle of $\mathcal{R}(\mathcal{M})$ form the negative witness. Of course, testing if $\mathcal{R}$ is acyclic and finding a cycle in $\mathcal{R}$ both cost $O(n)$ time. When $\mathcal{R}$ is acyclic, we can compute $\mathbf{d}$sep in $O(n)$ time using $\mathcal{R}$ in order to build $\mathcal{U}^*$. Hence, the difficulty of the algorithm is in finding $\mathcal{R}$.

By definition, $\mathcal{R}$ is obtained by removing the redundant edges of $\mathcal{S}$; this can be done in $O(n)$ time once the redundant edges of $\mathcal{S}$ are found. In turn, recall that an edge $A_i \to A_j$ is redundant if and only if

(red_1) $\mathbf{d}\text{len}(A_1, A_j) > \mathbf{d}\text{len}(A_1, A_i) + \text{len}(A_i \to A_j)$, or

(red_2) $\mathbf{d}\text{len}(A_1, A_j) = \mathbf{d}\text{len}(A_1, A_i) + \text{len}(A_i \to A_j)$ and
$(\mathbf{d}\text{len} \circ \mathbf{d}\text{ext})(A_1, A_j) > (\mathbf{d}\text{len} \circ \mathbf{d}\text{ext})(A_1, A_i) + \text{ext}(A_i \to A_j)$.

Thus, to locate the redundant edges we should find a path $\mathcal{W}_i$ from A_1 to A_i such that $\text{len}(\mathcal{W}_i) = \mathbf{d}\text{len}(A_1, A_i)$ and $\text{ext}(\mathcal{W}_i) = (\mathbf{d}\text{len} \circ \mathbf{d}\text{ext})(A_1, A_i)$ for every $1 \leq i \leq n$. By Lemma I.3, $\mathcal{W}_i$ is either a nose or hollow path. That is, $\mathcal{W}_i \in \{\mathcal{W}_{N,i}, \mathcal{W}_{H,i}\}$, where $\mathcal{W}_{N,i}$ is a nose path such that

(gn_1) $\text{len}(\mathcal{W}_{N,i}) = \max\{\text{len}(\mathcal{W}) \mid \mathcal{W} \text{ is a nose path from } A_1 \text{ to } A_i\}$, and

(gn_2) $$\text{ext}(\mathcal{W}_{N,i}) = \max\left\{\text{ext}(\mathcal{W}) \;\middle|\; \begin{array}{l}\mathcal{W} \text{ is a nose path from } A_1 \text{ to } A_i \\ \text{with } \text{len}(\mathcal{W}) = \text{len}(\mathcal{W}_{N,i})\end{array}\right\},$$

while $\mathcal{W}_{H,i}$ is defined analogously by replacing noses with hollows. The remainder of this section is devoted to the problems of finding $\mathcal{W}_{N,i}$ and $\mathcal{W}_{H,i}$.

Say that a nose (resp. hollow) walk $\mathcal{W} = B_1, \ldots, B_k$ is *greedy* when there exists $B_i \in \mathcal{W}$ such that B_j is greedy in $\mathcal{W}$ for every $1 \leq j \leq i$, while the subwalk $B_i, \ldots, B_k$ of $\mathcal{W}$ is a step walk. In other words, $\mathcal{W}$ is greedy when noses (resp. hollows) are preferred until some point in which only steps follow. It turns out that $\mathcal{W}_{N,i}$ and $\mathcal{W}_{H,i}$ are greedy paths. The proof of this fact is analogous to those of Lemmas 2 and 3, yet we include it for the sake of completeness.

Lemma 4 *For any nose (resp. hollow) walk $\mathcal{W}_N$ of $\mathcal{S}$ there exists a greedy nose (resp. hollow) walk $\mathcal{W}'_N$ of $\mathcal{S}$ joining the same vertices such that either* $\text{len}(\mathcal{W}_N) < \text{len}(\mathcal{W}'_N)$ *or* $\text{len}(\mathcal{W}_N) = \text{len}(\mathcal{W}'_N)$ *and* $\text{ext}(\mathcal{W}_N) \leq \text{ext}(\mathcal{W}'_N)$.

Proof: The proof is by induction on $|\mathcal{W}_N| - p$ and $|\mathcal{W}_N|$, where p is the position of the first non-greedy vertex of $\mathcal{W}_N$. The base case in which p is greater than the position of the last nose (resp. hollow) of $\mathcal{W}_N$ is trivial. Suppose, then, that B_1 is the first non-greedy vertex of $\mathcal{W}_N$ and that B_1 appears before the last nose (resp. hollow) of $\mathcal{W}_N$. Let $B_1 \to B$ the the nose (resp. hollow) from B_1 and consider the following cases.

Case 1: $\mathcal{W}_N$ is a nose walk. Let $\mathcal{W} = B_1, \ldots, B_i$ be the shortest subpath of $\mathcal{W}_N$ such that either $B_{i-1} \to B_i$ is a nose or $B_i = B$, $\mathcal{W}'$ be the path formed by the nose $B_1 \to B$ followed by the step path from B to B_i, and $\mathcal{W}'_N$ be the nose walk obtained from $\mathcal{W}_N$ by replacing $\mathcal{W}$ by $\mathcal{W}'$. Clearly, the position of the first non-greedy vertex of $\mathcal{W}'_N$ is greater than p. Thus, by induction, it suffices to show that $w(\mathcal{W}') \geq w(\mathcal{W})$ (for

$w \in \{\text{len}, \text{ext}\}$), because $w(\mathcal{W}'_N) = w(\mathcal{W}_N) - w(\mathcal{W}) + w(\mathcal{W}')$. If $B_i = B$, then either $\mathcal{W}$ contains at least one 1-step or $B_1 \to B_i$ is a $(-h)$-nose, thus $\text{len}(\mathcal{W}') > \text{len}(\mathcal{W})$. Otherwise, B_1, B_{i-1}, B, B_i appear in this order in $\mathcal{S}$ when the steps are traversed from B_1. As in Lemma 2, it is not hard to see that $w(\mathcal{W}') = w(\mathcal{W})$ when $B_1 \to B$ and $B_{i-1} \to B_i$ have equal jumps, while, as in Lemma 2, only four cases remain otherwise. All these cases are examined in the tables below.

h				len		ext	
B_1	B_{i-1}	B	B_i	$\mathcal{W}$	$\mathcal{W}'$	$\mathcal{W}$	$\mathcal{W}'$
$h-1$	$h-1$ or h	h	0	$-r$	$-r$	-1	-1
$h-2$	$h-1$	$h-1$	0	$-1-r$	$-1-r$	-1	-1
$h-1$	h	0	0	$-r$	$-r$	-1	-1
h	0	0	1	$-r$	$-r$	-1	-1

Case 2: $\mathcal{W}_N$ is a hollow walk. Let $\mathcal{W} = B_1, \ldots, B_i$ be the shortest subwalk of $\mathcal{W}_N$ such that $B_{i-1} \to B_i$ is a hollow. If $B_i = B_j$ for some $1 \leq j < i$, then the subwalk $\mathcal{W}_{ji} = B_j, \ldots, B_i$ of $\mathcal{W}$ is a cycle with exactly one 1-step or 0-hollow, thus $\text{len}(\mathcal{W}_{ji}) < 0$. So, the proof follows by induction on the hollow walk $\mathcal{W}'_N = \mathcal{W}_N \setminus \mathcal{W}_{ji}$ because $\text{len}(\mathcal{W}'_N) = \text{len}(\mathcal{W}_j) + \text{len}(\mathcal{W}_{ji})$ and the position of the first non-greedy hollow is at least p. If $B_i \notin \{B_1, \ldots, B_{i-1}\}$, then B, B_i, B_1, B_{i-1} appear in this order in a traversal of the steps of $\mathcal{S}$ from B. Let $\mathcal{W}'$ be the hollow path formed by $B_1 \to B$ followed by the step path from B to B_i and observe that, as in Case 1, it suffices to prove that $w(\mathcal{W}') = w(\mathcal{W})$ for $w \in \{\text{len}, \text{ext}\}$. Moreover, both equalities hold when $B_1 \to B$ and $B_{i-1} \to B_i$ have equal jumps. The equalities hold also when either $B_1 \to B$ or $B_{i-1} \to B_i$ is a 0-hollow. Indeed, if $B_1 \to B$ is a 0-hollow, then $h(B) = h(B_i) = h(B_1)$ and $h(B_{i-1}) \neq h(B)$, while if $B_{i-1} \to B_i$ is a 0-hollow, then $h(B_i) = h(B_1) = h(B_{i-1})$ and $h(B) \neq h(B_{i-1})$. In both of these cases, $\text{len}(\mathcal{W}) = \text{len}(\mathcal{W}') = -1$ and $\text{ext}(\mathcal{W}) = \text{ext}(\mathcal{W}') = 0$. Finally, when neither $B_1 \to B$ nor $B_{i-1} \to B_i$ are 0-hollows and $B_1 \to B$ and $B_{i-1} \to B_i$ have different jumps, we are left with only six cases, as in the tables below.

h				len		ext	
B_1	B_{i-1}	B	B_i	$\mathcal{W}$	$\mathcal{W}'$	$\mathcal{W}$	$\mathcal{W}'$
0	1	h or $h-1$	0	-1	-1	0	0
0	0	$h-1$	h	$-1+r$	$-1+r$	1	1
h	0	$h-1$	h	-1	-1	0	0
h	0	$h-1$	$h-1$	0	0	0	0
$h-1$	0	$h-2$	$h-1$	-1	-1	0	0

□

Recall that our problem is to find the len and ext values for the path $\mathcal{W}_{N,i}$ satisfying (gn_1) and (gn_2), for every $1 \leq i \leq n$. We solve this problem in

two phases. Clearly, $\mathcal{S}$ has a unique greedy nose path $\mathcal{N}$ beginning at A_1 that is maximal and ends with a nose. The first phase consist of traversing $\mathcal{N} = B_1, \ldots, B_k$ while $p(B_i) = (\text{len}(\mathcal{N}_i), \text{ext}(\mathcal{N}_i))$ is computed and stored for every subpath $\mathcal{N}_i = B_1, \ldots, B_i$ of $\mathcal{N}$. In the second phase each step $A_{i-1} \to A_i$ of $\mathcal{S}$ is traversed while $q(A_i) = (\text{len}(\mathcal{W}_{N,i}), \text{ext}(\mathcal{W}_{N,i}))$ is computed and stored. By Lemma 4, $\mathcal{W}_{N,i}$ is a greedy nose path starting at A_1, thus $\mathcal{W}_{N,i}$ is equal to a subpath of $\mathcal{N}$ plus a (possibly empty) step path. Consequently, there are only two possibilities for the last edge of $\mathcal{W}_{N,i}$ according to whether $A_i \in \mathcal{N}$ or not. If $A_i \notin \mathcal{N}$, then the last edge of $\mathcal{W}_{N,i}$ must be the step $A_{i-1} \to A_i$, thus $q(A_i) = q(A_{i-1}) + (\text{len}(A_{i-1} \to A_i), \text{ext}(A_{i-1} \to A_i))$. Otherwise, the last edge could be the step $A_{i-1} \to A_i$ or the nose $A_j \to A_i$. In the latter case $\mathcal{W}_{N,i}$ is the subpath $\mathcal{N}_i$ of $\mathcal{N}$. Thus, we can compute $\mathcal{W}_{N,i}$ by simply comparing the values $p(A_i)$ and $q(A_{i-1}) + (\text{len}(A_{i-1} \to A_i), \text{ext}(A_{i-1} \to A_i))$. Note that both traversals cost $O(n)$ time. The problem of finding $\mathcal{W}_{H,i}$ is analogous and it also costs $O(n)$ time.

Theorem 2 *There is a unified certifying algorithm that solves* Rep *in* $O(n)$ *time.*

4.4 Logspace construction of UCA models

Köbler et al. [10] ask whether it is possible to solve Rep in (deterministic) logspace. In this section we provide an affirmative answer to this question by showing that the algorithm of the previous section can be implemented to run in logspace. Before doing so, we briefly discuss the logspace recognition of UCA graphs, for the sake of completeness.

As proven by Tucker [22], two steps are enough to recognize if a given graph G is UCA: first check that G is PCA, and then verify if any of its PCA models admits an equivalent UCA model. Köbler et al. [10] show a logspace algorithm for the recognition of PCA graphs that outputs a PCA model $\mathcal{M}$ when G is PCA. We implement the algorithm by Kaplan and Nussbaum so that it runs in logspace when $\mathcal{M}$ is given. Let $\mathcal{S}_N$ be the subgraph of $\mathcal{S}$ obtained by removing all its hollows and all its steps that go from a vertex in which a nose starts. All the vertices of $\mathcal{S}_N$ have out-degree 1. So, the nose ratio of each cycle $\mathcal{W}$ of $\mathcal{S}_N$ can be obtained in logspace by traversing $\mathcal{W}$ from each of its vertices. By Lemma 2, r is the maximum among such ratios. Then, taking into account that $\mathcal{S}_N$ can be easily computed in logspace from $\mathcal{M}$, we conclude that r is obtainable in logspace. An analogous algorithm can be used to compute the hollow ratio R of $\mathcal{M}$ in logspace. By Theorem I.2, $\mathcal{M}$ is equivalent to a UCA model if and only if $r < R$. The algorithm can output, also in logspace, the nose and hollow cycles with ratios r and R, respectively. These cycles provide a negative witness that $\mathcal{M}$ is equivalent to no UCA models when $r \geq R$.

The logspace representation algorithm can be divided in two phases. In the first phase, $\mathcal{R}$ is built, while, in the second phase, the UCA model is obtained from a topological sort of its vertices. Clearly, the problem of computing $\mathcal{R}$ can be logspace reduced to querying which of the edges of $\mathcal{S}$ are redundant. By

(red_1) and (red_2), the problem of testing if $A_i \to A_j$ is redundant is logspace reduced to that of finding $\mathbf{d}\text{len}(A_1, A_i)$ and $(\mathbf{d}\text{len} \circ \mathbf{d}\text{ext})(A_1, A_i)$. As stated in the previous section, this problem is reduced to that of computing $\text{len}(\mathcal{W}_{N,i})$, $\text{ext}(\mathcal{W}_{N,i})$, $\text{len}(\mathcal{W}_{H,i})$, and $\text{ext}(\mathcal{W}_{H,i})$ where $\mathcal{W}_{N,i}$ is the greedy nose path from A_1 to A_i defined by (gn_1) and (gn_2), while $\mathcal{W}_{H,i}$ is defined analogously. By Lemma 2, $\mathcal{W}_{N,i} = \mathcal{N}_{a,b}$ for some $a, b < n$, where $\mathcal{N}_{a,b}$ is the walk obtained by first traversing a edges of the unique maximal greedy path $\mathcal{N}$ that begins at A_1, and then traversing b steps. So, by keeping the counters a and b, we can compute the maximum among the values of $\text{len}(\mathcal{N}_{a,b})$ and $\text{ext}(\mathcal{N}_{a,b})$ for the paths $\mathcal{N}_{a,b}$ ending at A_i with $a, b < n$. Such a computation requires logspace, thus $\mathcal{R}$ can be obtained in logspace.

Once $\mathcal{R}$ is built, we could compute $\mathcal{U}(c, \ell)$ by finding $\mathbf{d}\text{sep}(\mathcal{R}, A_1, A_i)$ for every $1 \leq i \leq n$. There is a major inconvenience with this approach: finding the longest path between two vertices of an acyclic digraph is a complete problem for the class of non-deterministic logspace problems. To deal with this issue we could observe that $\mathcal{R}$ is not only an acyclic digraph but also one with a rather particular structure.

Theorem 3 *If $\mathcal{M}$ is a PCA model, then $\mathcal{S}$ is a toroidal digraph.*

Proof: As the theorem is implied by [15], we defer its proof to Section 5 (see Corollary 1). □

Toroidal acyclic digraphs are much simpler than general acyclic digraphs. Yet, up to this date, the best algorithms to compute their longest paths run in unambiguous logspace [11]. For this reason, the UCA model computed in the second phase is a variant of $\mathcal{U}(c^*, \ell^*)$. The key idea is to observe that the reachability problem for toroidal digraphs that have a unique vertex with in-degree 0 can be solved in logspace [21]. That is, for $A_i, A_j \in V(\mathcal{R})$, the *reachability algorithm* in [21] outputs YES when there is a path from A_i to A_j in $\mathcal{R}$. Then, we can compute $\text{reach}(A_j) = |\{A_i \in V(\mathcal{R}) \mid \text{there is a path from } A_i \text{ to } A_j \text{ in } \mathcal{R}\}|$ in logspace for any given $A_j \in V(\mathcal{R})$. The representation algorithm takes advantage of this fact by replacing const with the easier-to-find reach. That is, the constructed UCA model $\mathcal{U}_{\text{reach}}$ is the (c^*, ℓ^*)-CA model with arcs $U_1, \ldots, U_n$ such that $c^* = (\ell^* + 1)(h + r) + e$ for $e = 4n$, $\ell^* + 1 = r_2 e^2$, and

$$s(U_j) = (\ell + 1)(h(A_j) + \mathbf{d}\text{len}(A_1, A_j)) + e(\mathbf{d}\text{ext} \circ \mathbf{d}\text{len})(A_1, A_j) + 2\,\text{reach}(A_j) \tag{6}$$

for every $1 \leq j \leq n$, where $r = r_1/r_2$ for $r_1, r_2 \in \mathbb{N}$. Observe that, as in Theorem I.2 (vi), c^*, ℓ^*, and $s(U_j)$ are integers. By the previous discussion, $\mathcal{U}_{\text{reach}}$ is obtainable in logspace. The fact that $\mathcal{U}_{\text{reach}}$ is equivalent to $\mathcal{M}$ follows from the next theorem.

Theorem 4 *Let $\mathcal{M}$ be a PCA model. Then, $\mathcal{R}$ is acyclic if and only if $\mathcal{U}_{\text{reach}}$ is an integer UCA model equivalent to $\mathcal{M}$.*

Proof: Suppose first that $\mathcal{R}$ is acyclic and let $U_1, \ldots, U_n$ be the arcs of $\mathcal{U}_{\text{reach}}$ as in its definition. Then, it suffices to show that:

(a) $s(U_j) \geq s(U_i) + \ell + 1 - cq$ for every nose $U_i \to U_j$ of $\mathcal{S}$,

(b) $s(U_j) \geq s(U_i) - \ell + 1 + cq$ for every hollow $U_i \to U_j$ of $\mathcal{S}$, and

(c) $s(U_j) \geq s(U_i) + 1 - cq$ for every step $U_i \to U_j$.

where $q \in \{0, 1\}$ equals 0 if and only if $A_i \to A_j$ is internal. Take any edge $A_i \to A_j$ and, for the sake of notation, let:

- $\Delta x(i,j) = x(A_j) - x(A_i)$ for $x \in \{h, \text{reach}\}$,
- $\Delta\,\text{len}(i,j) = \mathbf{d}\text{len}(A_1, A_j) - \mathbf{d}\text{len}(A_1, A_i) - \text{len}(A_i \to A_j)$, and
- $\Delta\,\text{ext}(i,j) = (\mathbf{d}\text{ext} \circ \mathbf{d}\text{len})(A_1, A_j) - (\mathbf{d}\text{ext} \circ \mathbf{d}\text{len})(A_1, A_i) - \text{ext}(A_i \to A_j)$.

By definition (6),

$$\begin{aligned} s(U_j) =& (\ell+1)(h(A_j) + \mathbf{d}\text{len}(A_1, A_j)) + e(\mathbf{d}\text{ext} \circ \mathbf{d}\text{len})(A_1, A_j) + 2\,\text{reach}(A_j) \\ =& (\ell+1)(h(A_i) + \Delta h(i,j) + \mathbf{d}\text{len}(A_1, A_i) + \text{len}(A_i \to A_j) + \Delta\text{len}(i,j)) + \\ & e((\mathbf{d}\text{ext} \circ \mathbf{d}\text{len})(A_1, A_i) + \text{ext}(A_i \to A_j) + \Delta\,\text{ext}(i,j)) + \\ & 2\,\text{reach}(A_i) + 2\Delta\,\text{reach}(i,j) \\ =& s(U_i) + (\ell+1)(\Delta h(i,j) + \text{len}(A_i \to A_j)) + e(\text{ext}(A_i \to A_j)) + \varepsilon \qquad \text{(i)} \end{aligned}$$

where $\varepsilon = (\ell+1)\Delta\,\text{len}(i,j) + e\Delta\,\text{ext}(i,j) + 2\Delta\,\text{reach}(i,j)$.

Note that $\varepsilon \geq 2$. Indeed, if $A_i \to A_j$ is redundant, then either $\Delta\,\text{len}(i,j) > 0$ or $\Delta\,\text{len}(i,j) = 0$ and $\Delta\,\text{ext}(i,j) > 0$; thus $\varepsilon \geq 2$ because len and ext are large enough (see Chapter I, Theorem I.2 (v) $\Rightarrow$ (vi)). If $A_i \to A_j$ is not redundant, then $A_i \to A_j$ is an edge of $\mathcal{R}$ and $\Delta\,\text{reach}(i,j) > 0$, while $\Delta\,\text{len}(i,j) = \Delta\,\text{ext}(i,j) = 0$. Consequently, $\varepsilon \geq 2$ regardless of whether $A_i \to A_j$ is redundant or not. Then, (a)–(c) follow by inspection, considering the 10 possible kinds of edges that are present in $\mathcal{S}$. For the sake of completeness, the table below sums up all these cases (where $\Delta = \Delta h(i,j)$, len $= \text{len}(A_i \to A_j)$, ext $= \text{ext}(A_i \to A_j)$); recall that $c = h(\ell+1) + r(\ell+1) + e$.

Type $A_i \to A_j$	q	Δ	len	ext	(i) $- \varepsilon$
1-nose	0	1	0	0	$s(U_i) + \ell + 1$
$(1-h)$-nose	1	$-h+1$	$-r$	-1	$s(U_i) + \ell + 1 - c$
$(-h)$-nose	1	$-h$	$1-r$	-1	$s(U_i) + \ell + 1 - c$
(-1)-hollow	0	-1	0	0	$s(U_i) - \ell - 1$
0-hollow	0	0	-1	0	$s(U_i) - \ell - 1$
$(h-1)$-hollow	1	$h-1$	r	1	$s(U_i) - \ell - 1 + c$
h-hollow	1	h	$r-1$	1	$s(U_i) - \ell - 1 + c$
0-step	0	0	0	0	0
1-step	0	1	-1	0	0
$(-h)$-step	1	$-h$	$-r$	-1	$s(U_i) - c$

The converse follows from Theorem I.2 (i) $\Rightarrow$ (v). □

The main theorem of this section then follows.

Theorem 5 *There is an algorithm that solves* Rep *in logspace.*

5 Minimal UIG models

In Section I.6 we define and study the minimal representation problem for UCA graphs. Minimal UCA models are a natural generalization of minimal UIG models as defined by Pirlot [16]. According to Pirlot, an (ℓ, d, d_s)-IG model with arcs $A_1 < \ldots < A_n$ is (∞, d, d_s)-*minimal* when

(min-uig$_1$) $\ell \leq \ell'$, and

(min-uig$_2$) $s(A_i) \leq s(A'_i)$ for every $1 \leq i \leq n$,

for every equivalent (ℓ', d, d_s)-IG model. Similarly, we say that a (c, ℓ, d, d_s)-CA model $\mathcal{M}$ is (d, d_s)-*minimal* when

(min-uca$_1$) $\ell \leq \ell'$, and

(min-uca$_2$) $c \leq c'$,

for every equivalent (c', ℓ', d, d_s)-CA model. Pirlot [16] proved that every UIG model is equivalent to an (∞, d, d_s)-minimal UIG model, for every $d, d_s \in \mathbb{Q}$. The analogous fact that every UCA model is equivalent to a minimal UCA model was proven in Chapter I.

Theorem I.5 *Every UCA graph admits a (d, d_s)-minimal UCA model for all $d, d_s \in \mathbb{Q}$. Furthermore, if $\mathcal{M}$ is equivalent to a $(c, \ell + y, d, d_s)$- and a $(c + x, \ell, d, d_s)$-CA model for $x, y \geq 0$, then $\mathcal{M}$ is also equivalent to a $(c + a, \ell + b, d, d_s)$-CA model, for every $0 \leq a \leq x$ and $0 \leq b \leq y$.*

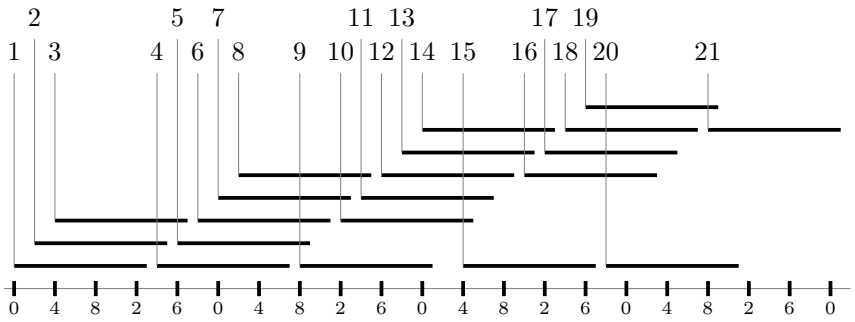
In the light of Theorem I.5, it makes sense to study the *minimal UCA representation* (MinUCA) problem in which $\mathcal{M}$ and $d, d_s \in \mathbb{Q}_{\geq 0}$ are given as input and a (d, d_s)-minimal model equivalent to $\mathcal{M}$ must be found. Unfortunately, we do not know how to solve MinUCA, even if we restrict the problem to $d, d_s \in \mathbb{N}$. The major issue is that we cannot prove that a (d, d_s)-minimal model is integer when $d, d_s \in \mathbb{N}$. The situation is quite different in the UIG case, as every (∞, d, d_s)-minimal model is integer when $d, d_s \in \mathbb{N}$ (see [16] and Corollary I.1). Is for this reason that we define $(\mathbb{N}, d, d_s)$-*minimal* models as those integer UCA models that satisfy (min-uca$_1$) and (min-uca$_2$) for every integer (c', ℓ', d, d_s)-CA model. The IntMinUCA problem is to compute a $(\mathbb{N}, d, d_s)$-minimal model equivalent to a UCA model $\mathcal{M}$ when $\mathcal{M}$ and $d, d_s \in \mathbb{N}$ are given as input.

Theorem I.6 IntMinUCA *is can be solved in $O((d + d_s)n^4 \log(n(d + d_s)))$ time, for every $d, d_s \in \mathbb{N}$.*

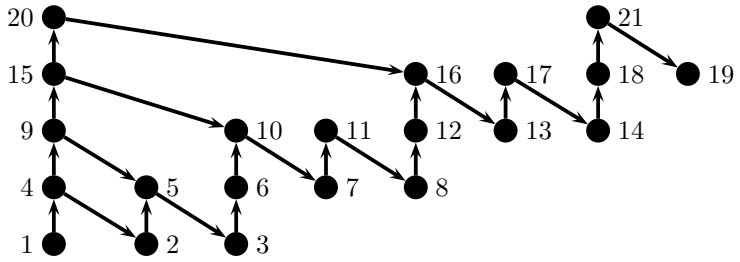
We now consider the *minimal UIG representation* (MinUIG) problem in which $\mathcal{M}$ and $d, d_s \in \mathbb{Q}_{\geq 0}$ are given as input and an (∞, d, d_s)-minimal UIG model equivalent to $\mathcal{M}$ must be generated. As stated above, Pirlot [16] proved that every UIG model $\mathcal{M}$ is equivalent to an (∞, d, d_s)-minimal model $\mathcal{M}^*$. However, it was Mitas [15] who showed that such a model can be found in linear time by transforming $\mathcal{M}$ into $\mathcal{M}^*$. Unfortunately, her proof has a flaw

that invalidates the minimality arguments. Though $\mathcal{M}^*$ is equivalent to $\mathcal{M}$, it need not be (∞, d, d_s)-minimal. On the other hand, the algorithm implied by Theorem I.6 can be implemented to run in $O(n^2 \log n)$ time when applied to $\mathcal{M}$. We shall not prove this fact; instead, we concentrate our efforts on fixing Mitas' algorithm. In the remainder of this section we briefly describe Mitas' algorithm and its counterexample, and propose the patch. The obtained algorithm runs in $O(n^2)$ time and linear space, and works for every $d, d_s \in \mathbb{Q}$. Solving MinUIG in linear time remains an open problem, as far as our knowledge extends.

Let $\mathcal{M}$ be a PIG model with arcs $A_1 < \ldots < A_n$. By definition, no arc of $\mathcal{M}$ crosses 0, thus $\mathcal{S}$ has no external hollows. Similarly, external noses and steps play no role in $\mathcal{S}$ when dealing with the (ℓ, d, d_s)-IG models equivalent to $\mathcal{M}$, as $c = \infty$. Therefore, we assume that $\mathcal{S}$ has only five types of edges, namely 1-noses, 0- and 1-steps, and 0- and (-1)-hollows. Mitas identifies two special vertices of $\mathcal{S}$ for each height value. A *leftmost* vertex is a vertex A_i such that either $i = 1$ or $h(A_i) = h(A_{i-1}) + 1$, while a *rightmost* vertex is a vertex A_j such that either $j = n$ or $h(A_{j+1}) = h(A_j) + 1$.



(a) A 13-UIG model equivalent to $\mathcal{M}$.



(b) The canonical drawing of $\mathcal{T}(\mathcal{M})$; 0-steps are not shown for the sake of exposition.

FIGURE 4. *Counterexample to* (8)*:* $\mathbf{d}\mathrm{sep}(\mathcal{T}, A_1, A_{19}) - \mathbf{d}\mathrm{sep}(\mathcal{T}, A_1, A_{15}) = 14$ *(b), thus Mitas' algorithm computes a* 14*-UIG model (for* $d = 1$ *and* $d_s = 0$*). However, the maximum cycle of* $\mathcal{S}$ *has length* 13 *and, consequently, a* 13*-UIG model equivalent to* $\mathcal{M}$ *exists (a).*

Suppose $\mathcal{M}$ is equivalent to an (ℓ, d, d_s)-IG model and let $\mathcal{U} = \mathcal{U}(\infty, \ell, d, d_s)$

be as in Section 3, but replacing $\mathcal{B}$ and A_0 with $\mathcal{S}$ and A_1, respectively. That is, the arc U_i corresponding to A_i begins at $s(U_i) = \mathbf{d}\text{sep}(A_1, A_i)$ for every $1 \leq i \leq n$. For $x \in \{1, \ldots, h\}$, let A_i and A_j be leftmost and rightmost vertices with $h(A_i) = h(A_j) = x$, and $\mathcal{W}_x$ be a path from A_i to A_j. By definition of $\mathcal{U}$, it follows that $s(U_j) - s(U_i) \geq \text{sep}(\mathcal{W}_x)$. That is,

$$\mathbf{d}\text{sep}(A_1, A_j) - \mathbf{d}\text{sep}(A_1, A_i) \geq \text{sep}(\mathcal{W}_x). \tag{7}$$

Mitas' key idea is to take ℓ so that (7) holds with equality when $\text{sep}(\mathcal{W}_x)$ is maximum (recall sep depends on ℓ). The flaw, however, is that she discards 0-hollows and 1-steps before solving (7).

To make the above statement more precise, let $\mathcal{T}(\mathcal{M})$ be the digraph obtained from $\mathcal{S}(\mathcal{M})$ by removing all the 0-hollows and 1-steps, and $\mathcal{W}_x^{\mathcal{T}}$ be a path from A_i to A_j in $\mathcal{T}$ (as usual we drop the parameter $\mathcal{M}$ from $\mathcal{T}$). Mitas claims [15, Theorem 5] that

$$\mathbf{d}\text{sep}(\mathcal{T}, A_1, A_j) - \mathbf{d}\text{sep}(\mathcal{T}, A_1, A_i) = \text{sep}(\mathcal{W}_x^{\mathcal{T}}). \tag{8}$$

when $\text{sep}(\mathcal{W}_x^{\mathcal{T}})$ is maximum and ℓ is minimum. Figure 4 shows a counterexample to this fact. The problem is that $\mathbf{d}\text{sep}(\mathcal{S}, A_1, A_i)$ is greater than $\mathbf{d}\text{sep}(\mathcal{T}, A_1, A_i)$ when every maximum path from A_1 to A_i contains the 0-hollow or 1-step ending at A_i.

Equation (8) is fundamental for keeping the time and space complexity low. The main observation is that $\mathcal{T}$ is acyclic whereas $\mathcal{S}$ is not.

Lemma 5 ([15]) *Digraph $\mathcal{T}(\mathcal{M})$ is acyclic, for any UIG model $\mathcal{M}$.*

Given that $\mathcal{T}$ is acyclic, we can compute the *column* that every arc of $\mathcal{M}$ occupies in a pictorial description of $\mathcal{T}$. The *column* of A_1 is $c(A_1) = 0$, while, for every $1 < i \leq n$, the *column* of A_i is:

$$c(A_i) = \max \left\{ \begin{array}{c|l} c(N) + \varepsilon & N \to A_i \text{ is a 1-nose} \\ c(H) + 1 & H \to A_i \text{ is a 0-step} \\ c(S) + 1 & S \to A_i \text{ is a } -1\text{-hollow} \end{array} \right\} \tag{9}$$

for a small enough ε (say $\varepsilon \ll 1/n$); obviously, if A_i is not the end of a nose (resp. hollow, step), then the corresponding value in the above equation is 0. It is easy to see, by the existence of 0-steps, that $c(I_i) \leq c(I_k) \leq c(I_j)$ when A_i and A_j are the leftmost and rightmost with height x, for every $A_i < A_k < A_j$. In Figure 4, each vertex A of $\mathcal{S}$ occupies the coordinate $(c(A), h(A))$ on the plane, for some imperceptible ε, while each directed edge is a straight arrow. This pictorial description, which we call the *canonical drawing* of $\mathcal{T}$, was proposed by Mitas and it is quite useful for simplifying some geometrical arguments. The reason is that this drawing is a plane digraph; we include a proof of this fact as it is not completely explicit in [15].

Theorem 6 (see [15, 18]) *The canonical drawing of $\mathcal{T}$ is a plane digraph.*

Proof: Suppose, to obtain a contradiction, that the canonical drawing of $\mathcal{T}$ is not a plane graph. Then, there are two crossing straight lines that correspond to the edges $A_i \to A_j$ and $A_x \to A_y$ with $h(A_i) \leq h(A_x)$. By definition, $\mathcal{T}$ has only 1-noses, (-1)-hollows and 0-steps, while every vertex A is positioned in $(c(A), h(A))$. Hence, it follows that $A_i \to A_j$ is a 1-nose, $A_x \to A_y$ is a (-1)-hollow, and $A_i < A_y < A_x < A_j$. But this configuration is impossible because it implies that $t(A_i)s(A_j)$ and $s(A_x)t(A_y)$ are consecutive, while $t(A_i) < t(A_y)$ and $s(A_x) < s(A_j)$. □

Corollary 1 (Theorem 3) *If $\mathcal{M}$ is a PCA graph, then $\mathcal{S}$ is a toroidal digraph.*

Proof: A torus can be obtained from a rectangle by first pasting its north and south borders together, and then pasting the east end of the obtained cylinder with its west end. Thus, it suffices to show how to draw $\mathcal{S}$ into a rectangle allowing some edges to escape from the north (resp. east) into the south (resp. west). Let $\mathcal{T}$ be obtained from $\mathcal{S}$ by removing all the external edges, plus 1-steps and 0-hollows. To draw $\mathcal{S}$, first copy the canonical drawing of $\mathcal{T}$ into the rectangle. Then, draw all the 0-hollows and 1-steps so that they escape through the east, all the h-hollows and $(-h)$-noses so as to run through the north, and the $(-h)$-step and all the $(h-1)$-hollows and $(1-h)$-noses by going through the north first and then through the east. It is not hard to see that such a drawing is always possible. □

In the next lemma we take advantage of the canonical drawing to prove that every cycle of $\mathcal{S}$ contains exactly one 0-hollow or 1-step. Pirlot also studies the shape of the cycles of $\mathcal{S}$ [16], but without taking advantage of Mitas' canonical drawing. For the next lemma, recall that $\text{len}(\mathcal{W}) = -\eta_0(\mathcal{W}) - \sigma_1(\mathcal{W})$ for any cycle $\mathcal{W}$.

Lemma 6 ([16, Proposition 2.11]) *If $\mathcal{M}$ is a PIG model, then* $\text{len}(\mathcal{W}) = -1$ *for any cycle $\mathcal{W}$ of $\mathcal{S}$.*

Proof: Note that $\text{len} = -\eta_0 - \sigma_1$ (3), hence, by Lemma 5, $\text{len} < 0$. Suppose, to obtain a contradiction, that $\text{len} < -1$. Then, $\mathcal{W}$ has a subpath $\mathcal{W}_{1,j} = B_1, \ldots, B_j$ with no 0-hollows nor 1-steps such that $B_0 \to B_1$ and $B_j \to B_{j+1}$ each is either a 0-hollow or a 1-step of $\mathcal{W}$. Among all such possible paths, take $\mathcal{W}$ so that $h(B_1)$ is maximum. Note that $B_0 \neq B_j$, thus $\mathcal{W}$ has another path $\mathcal{W}_{-k,0} = B_{-k}, \ldots, B_0$ such that B_{-k} is its unique leftmost vertex. By the maximality of $h(B_1)$, it follows that $h(B_0) \geq h(B_1) - 1 \geq h(B_{-k})$ while, since $h(B_j) \leq h(B_{j+1}) \leq h(B_1) - 1 \leq h(B_0)$ and $B_0 \neq B_j$, it follows that $h(B_0) > h(B_j)$.

Call Gr^+ to the curve that results by traversing $\mathcal{W}_{1,j}$ in the canonical drawing of $\mathcal{T}$. Note that Gr^+ is indeed the graph of a continuous function on $\mathbb{R} \to \mathbb{R}$ because $c(B_{i+1}) > c(B_i)$ for every $1 \leq i < j$ by (9). Similarly, the curve Gr^- that results by traversing $\mathcal{W}_{-k,0}$ in the canonical drawing of $\mathcal{T}$ is also the graph of a continuous function. Since $h(B_{i+1}) = h(B_i) \pm 1$ for every $i \in \{-k, \ldots, j\} \setminus 0$, it follows that Gr^+ contains a vertex with height x for every $h(B_j) \leq x \leq h(B_1)$

and Gr^- contains a vertex with height x for every $h(B_{-k}) \leq x \leq h(B_0)$. Then, taking into account that B_1 and B_{-k} are leftmost vertices with $h(B_1) > h(B_{-k})$ and B_j and B_0 are rightmost vertices with $h(B_0) > h(B_j)$, we obtain that Gr^+ and Gr^- intersect. Hence, by Theorem 6, $\mathcal{W}_{1,j}$ and $\mathcal{W}_{-k,0}$ have a nonempty intersection, which implies that $\mathcal{W}$ is not a cycle. □

By Lemma 6 and (2), $\mathrm{sep}(\mathcal{W}) = \mathrm{const}(\mathcal{W}, d, d_s) - \ell - 1$ for every cycle $\mathcal{W}$. Then, by Theorem I.1 and Lemma 6, the minimum ℓ^* such that $\mathcal{M}$ is equivalent to an (ℓ^*, d, d_s)-IG model satisfies

$$\begin{aligned}
\ell^* + 1 &= \max\{\mathrm{const}(\mathcal{W}, d, d_s) \mid \mathcal{W} \text{ is a cycle of } \mathcal{S}\} \\
&= \max\left\{\mathrm{const}(\mathcal{W}, d, d_s) \;\middle|\; \begin{array}{l} \mathcal{W} \text{ is a path } B_1, \ldots, B_j \text{ of } \mathcal{T} \\ \text{for a 1-step or 0-hollow } B_j \to B_1 \end{array}\right\} \\
&= \max\{\mathbf{d}\mathrm{const}_{d,d_s}(\mathcal{T}, A_i, A_j) \mid A_j \to A_i \text{ is either a 1-step or 0-hollow}\}.
\end{aligned}$$

Since $\mathcal{T}$ is acyclic, we can compute $\mathbf{d}\mathrm{const}_{d,d_s}(\mathcal{T}, A_i, A_j)$ in $O(n)$ time and space for any given 1-step or 0-hollow $A_j \to A_i$ of $\mathcal{S}$. Then, ℓ^* is obtained in $O(hn)$ time.

Once ℓ^* has been obtained, $\mathcal{U} = \mathcal{U}(\infty, \ell^*, d, d_s)$ can be constructed in $O(n^2)$ time and linear space as in Section 3. We claim that $\mathcal{U}$ is a (∞, d, d_s)-minimal model. Indeed, $\mathcal{U}$ satisfies (min-uig$_1$) by the minimality of ℓ^*. To see that $\mathcal{U}$ satisfies (min-uig$_2$), consider any path $\mathcal{W}$ of $\mathcal{S}$ from A_1 to A_j. Note that $h(A_j) \geq \sigma_1 + \eta_0 = -\mathrm{len}$ because no leftmost vertex is traversed twice by $\mathcal{W}$. Therefore, by (2),

$$\begin{aligned}
\mathrm{sep}_{(\infty,\ell,d,d_s)} &= (\ell+1)(h(A_j) + \mathrm{len}) + \mathrm{const} \\
&\geq (\ell^*+1)(h(A_j) + \mathrm{len}) + \mathrm{const} = \mathrm{sep}_{(\infty,\ell^*,d,d_s)}
\end{aligned}$$

for any $\ell \geq \ell^*$. Consequently, since $s(A_j) \geq \mathbf{d}\mathrm{sep}_{(\infty,\ell,d,d_s)}(A_1, A_j)$ in any (ℓ, d, d_s)-UIG model equivalent to $\mathcal{U}$, it follows that $\mathcal{U}$ satisfies (min-uig$_2$) as well. We conclude that $O(n^2)$ time and linear space suffices to solve MinUIG.

Theorem 7 MinUIG *can be solved in $O(n^2)$ time and linear space, for any $d, d_s \in \mathbb{Q}$.*

In 2015, while this manuscript was under review, Durán et al. [3] presented an $O(n^3)$ time algorithm to solve MinUCA when the input model is PIG and $d, d_s \in \{0, 1\}$. Even though we consider MinUCA as being a generalization of MinUIG to UCA models, the fact is that MinUCA is simpler than MinUIG when the input model is PIG. In other words, any solution to MinUIG on PIG models is also a solution to MinUCA, but the converse is not true. Indeed, (min-uig$_2$) implies (min-uca$_2$) but (min-uca$_2$) does not imply (min-uig$_2$). (In formal terms, (d, d_s)-minimal UIG models need not be (∞, d, d_s)-minimal.) Although Durán et al. acknowledge our algorithm to solve MinUIG (a preprint of this manuscript is available since 2014), they state that the output model $\mathcal{U}$ of our algorithm satisfies (min-uig$_1$), but they omit to say that our algorithm also ensures (min-uig$_2$) on $\mathcal{U}$. Moreover, they do not even mention that

our algorithm guarantees (min-uca$_2$). We remark that not only our algorithm solves MinUCA on PIG models faster, it also solves the more general MinUIG problem without requiring d and d_s to be integer.

6 Powers of paths and cycles

Powers of paths and cycles are intimately related to UIG and UCA graphs, respectively. For any graph G, its *k-th power* G^k is the graph obtained from G by adding an edge between v and w whenever there is a path in of length at most k joining them. In this section we write P_q and C_q to denote the path and cycle graphs with q vertices, respectively. Lin et al. [12] noted that G is a UCA (resp. UIG) graph if and only if G is an induced subgraph of C_q^k (resp. P_q^k) for some q, k (see also [5] for UIG graphs and [6] for UCA graphs).

Costa et al. [2] propose a specialized $O(n^2)$ time and space algorithm that, given a PIG model $\mathcal{M}$, finds a UIG model $\mathcal{U}$ representing a power of a path $P_{q(k)}^k$ in such a way that $\mathcal{M}$ is equivalent to some induced submodel $\mathcal{U}'$ of $\mathcal{U}$ and $k, q(k)$ are as small as possible. The reason for writing $q(k)$ dependent on k is to be as truthful to [2] as we can; they always write the number of vertices as a function on the power. This is not important, though, as we know that q is independent of k by Pirlot's minimality Theorem [16]. Mitas' algorithm could have been applied to obtain k and q in $O(n)$ time and space, under the assumption that it is correct. Interestingly, Pirlot's Theorem and Mitas' algorithm predate [2] for at least fifteen years. Moreover, [19, Section 9], which is referenced within [2], mentions that Mitas' algorithm could be adapted to work when the input is a PIG model. The purpose of this section is to apply the minimization algorithms to find powers of paths and cycles supergraphs.

Let $\mathcal{C}_q^k$ (resp. $\mathcal{P}_q^k$) be the $(2q, 2k+1)$-CA (resp. $(2k+1)$-IG) model that has an arc with beginning point $2i$ for every $0 \leq i < q$, and $\bullet = \mathbb{N}$ (resp. $\bullet = \infty$). It is not hard to see that $\mathcal{C}_q^k$ (resp. $\mathcal{P}_q^k$) is a $(\bullet, 1, 0)$- and $(\bullet, 1, 1)$-minimal model representing C_q^k (resp. P_q^k). We say that a (c, ℓ)-CA (resp. ℓ-IG) model $\mathcal{M}^*$ is *$\bullet$-completable* when $\mathcal{M}^*$ can be obtained by removing arcs from $\mathcal{C}_q^k$ (resp. $\mathcal{P}_q^k$) for some $k, q \geq 0$. In such case, $\mathcal{C}_q^k$ (resp. $\mathcal{P}_q^k$) is referred to as the *$\bullet$-completion* of $\mathcal{M}^*$, while $\mathcal{M}^*$ is said to be a *$(\bullet, k, q)$-extension* of $\mathcal{M}$ for every UCA (resp. UIG) model $\mathcal{M}$ equivalent to $\mathcal{M}^*$. Note that $\mathcal{M}^*$ is $\bullet$-completable if and only if:

(ext_1) ℓ is odd,

(ext_2) c is even (resp. $c = \infty$), and

(ext_3) all its beginning points are even (thus $\mathcal{M}^*$ is a $(c, \ell, 1, 1)$-CA model).

Under this new terminology, the result by Lin et al. [12] states that every UCA (resp. UIG) model $\mathcal{M}$ admits a $(\bullet, k, q)$-extension $\mathcal{M}^*$ for some $k, q \geq 0$. In analogy to minimal models, we say that $\mathcal{M}^*$ is a *minimal $\bullet$-extension* of $\mathcal{M}$ when $q \leq q'$ and $k \leq k'$ for every $(\bullet, k', q')$-extension of $\mathcal{M}$. The *minimal power*

of a cycle (resp. path) problem ($\textsc{MinC}_q^k$) consists of finding $\mathcal{M}^*$ when the UCA (resp. UIG) model $\mathcal{M}$ is given as input. Note that any (∞, k, q)-extension $\mathcal{M}^*$ of $\mathcal{M}$ can be regarded as being a $(\mathbb{N}, k, q)$-extension of $\mathcal{M}$; just consider the circumference of $\mathcal{M}^*$ is the even value $t(A_n)+1$ (where A_n is the maximum arc of $\mathcal{M}^*$). Thus, $\textsc{MinC}_q^k$ is indeed a generalization of $\textsc{MinP}_q^k$. We now discuss how to solve $\textsc{MinC}_q^k$ and $\textsc{MinP}_q^k$.

The fact that $\mathcal{M}$ admits a minimal $\mathbb{N}$-extension follows by Theorems I.1 and I.5. Indeed, if ℓ^* is the minimum odd number such that $\mathcal{M}$ is equivalent to a $(c, \ell^*, 1, 1)$-CA model, and c^* is the minimum even number such that $\mathcal{M}$ is equivalent to a $(c^*, \ell, 1, 1)$-CA model, then $\mathcal{M}$ is equivalent to a $(c^*, \ell^*, 1, 1)$-CA model by Theorem I.5. Furthermore, $\text{sep}_{c^*,\ell^*,1,1}(A_i \to A_j)$ is even for every edge $A_i \to A_j$ of $\mathcal{S}$, by (sep_1)–(sep_4). Thus, all the beginning points of $\mathcal{M}^* = \mathcal{U}(\mathcal{M}, c^*, \ell^*, 1, 1)$ are even. Then, $\mathcal{M}^*$ is $\mathbb{N}$-completable by (ext_1)–(ext_3), while it is equivalent to $\mathcal{M}$ by Theorem I.1. That is, $\mathcal{M}^*$ is the minimal $\mathbb{N}$-extension of $\mathcal{M}$ and, thus, the solution to $\textsc{MinC}_q^k$. The values ℓ^* and c^* can be found $O(n^4 \log n)$ time with an algorithm similar to the one in Section I.6. (We remark that finding the $(\mathbb{N}, 1, 1)$-minimal UCA model equivalent to $\mathcal{M}$ is not enough to solve $\textsc{MinC}_q^k$, as we must ensure that c is even and ℓ is odd; however, adapting the algorithm in Section I.6 to this case is trivial.)

For the special case in which $\mathcal{M}$ is a UIG model, we observe that any $(\infty, 1, 1)$-minimal model $\mathcal{M}^*$ equivalent to $\mathcal{M}$ is a minimal ∞-extension of $\mathcal{M}$. Just recall that the length ℓ^* of the arcs in $\mathcal{M}^*$ is equal to $\text{const}(\mathcal{W}, 1, 1) - 1$ for some path $\mathcal{W}$ of $\mathcal{S}(\mathcal{M}^*)$. Since $\text{const}(\mathcal{W}, 1, 1)$ is even, it follows that ℓ^* is odd and, thus, $\text{sep}_{\infty,\ell^*,1,1}(A_i \to A_j)$ is even for every edge $A_i \to A_j$ of $\mathcal{S}(\mathcal{M}^*)$. By ($\text{ext}_1$)–($\text{ext}_3$), this implies that $\mathcal{M}^*$ is an ∞-extension of $\mathcal{M}$ which, of course, is minimal by (min-uig_1) and (min-uig_2). By Theorem 7, $\textsc{MinP}_q^k$ is solvable in $O(n^2)$ time and linear space.

7 Further remarks

Synthetic graphs proved to be an important tool for studying what the UIG representations of PIG graphs look like. The generalization to PCA models is direct; to represent the separation constraints that an equivalent UCA model must satisfy, all we had to add to Pirlot's original formulation was the variable c representing the circumference of the circle. Generalizations of simple ideas from PIG to PCA graphs are not always as easy to obtain. Unfortunately, Pirlot's ideas were not exploited in the context of PCA graphs. In this Chapter we proved that the synthetic graph is not only valuable from a theoretical perspective, but also from an algorithmic point of view. In this closing section we provide some remarks and discuss some open problems related to this chapter.

In Section 5 we fixed Mitas' algorithm to solve the minimization problem for UIG models. Unfortunately, the running time of the patched algorithm is $O(n^2)$. Two bottlenecks are responsible for this running time. On the one hand, we have to compute the minimum length value ℓ^*. On the other hand, we have to apply the Bellman-Ford algorithm to compute the actual model.

With respect to the space complexity, it is not hard to observe that ℓ^* can be computed in unambiguous logspace. Indeed, all we have to do is to find the distance between the leftmost and rightmost arcs for every height. As the canonical drawing is a plane graph with $O(1)$ vertices with 0 in-degree, this problem requires unambiguous logspace [11]. Finding logspace algorithms to compute ℓ^* and the minimal model remain as open problems.

As discussed in Chapter I, UCA graphs can admit an exponential number of *non-equivalent* UCA models, each of which is equivalent to a minimal UCA model. We say, therefore, that a model $\mathcal{M}$ is *minimum* when it satisfies (min-uca$_1$) and (min-uca$_2$) for every model $\mathcal{M}'$ such that $G(\mathcal{M})$ is isomorphic to $G(\mathcal{M}')$. As observed in Section I.7, any minimal UIG model of $G(\mathcal{M})$ is minimum. Consequently, the ∞-completion $\mathcal{P}_q^k$ of $\mathcal{M}$ is such that q and k are the minimum for which $G(\mathcal{M})$ is an induced subgraph of P_q^k. Unfortunately, we cannot assert that the $\mathbb{N}$-completion $\mathcal{C}_q^k$ of a UCA model $\mathcal{M}$ is such that q and k are the minimum for which $G(\mathcal{M})$ is an induced subgraph of C_q^k. In fact, as discussed in Section I.7, a UCA graph may admit minimal models whose circumference and arc lengths differ. Thus, the problem of computing the minimum values q and k such that a UCA graph G is an induced subgraph of C_q^k remains open.

Acknowledgements

I want to thank the anonymous reviewers for the time they spent to help me improve this long contribution. I'm particularly indebted to the reviewer who read the original manuscript before it was split, as without his/her report the paper would have never been considered for publication, and to the communicating editor, Sue Whitesides, whose help was essential for transforming the first manuscript into a publishable work.

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, Cambridge, MA, third edition, 2009.

[2] V. Costa, S. Dantas, D. Sankoff, and X. Xu. Gene clusters as intersections of powers of paths. *J. Braz. Comput. Soc.*, 18(2):129–136, 2012. doi:10.1007/s13173-012-0064-8.

[3] G. Durán, F. Fernández Slezak, L. N. Grippo, F. de Souza Oliveira, and J. L. Szwarcfiter. On unit interval graphs with integer endpoints. In *LAGOS'15—{VIII} Latin-American Algorithms, Graphs and Optimization Symposium*, volume 50 of *Electron. Notes Discrete Math.*, pages 445–450. Elsevier Sci. B. V., Amsterdam, 2015. doi:10.1016/j.endm.2015.07.074.

[4] G. Durán, A. Gravano, R. M. McConnell, J. Spinrad, and A. Tucker. Polynomial time recognition of unit circular-arc graphs. *J. Algorithms*, 58(1):67–78, 2006. doi:10.1016/j.jalgor.2004.08.003.

[5] N. J. Fine and R. Harrop. Uniformization of linear arrays. *J. Symb. Logic*, 22:130–140, 1957. doi:10.2307/2964174.

[6] M. C. Golumbic and P. L. Hammer. Stability in circular arc graphs. *J. Algorithms*, 9(3):314–320, 1988. doi:10.1016/0196-6774(88)90023-5.

[7] H. Kaplan and Y. Nussbaum. Certifying algorithms for recognizing proper circular-arc graphs and unit circular-arc graphs. *Discrete Appl. Math.*, 157(15):3216–3230, 2009. doi:10.1016/j.dam.2009.07.002.

[8] P. Klavík, J. Kratochvíl, Y. Otachi, I. Rutter, T. Saitoh, M. Saumell, and T. Vyskočil. Extending partial representations of proper and unit interval graphs. *CoRR*, abs/1207.6960v2, 2014. URL: http://arxiv.org/abs/1207.6960v2.

[9] P. Klavík, J. Kratochvíl, Y. Otachi, I. Rutter, T. Saitoh, M. Saumell, and T. Vyskočil. Extending partial representations of proper and unit interval graphs. *Algorithmica*, 77(4):1071–1104, 2017. doi:10.1007/s00453-016-0133-z.

[10] J. Köbler, S. Kuhnert, and O. Verbitsky. Solving the canonical representation and Star System Problems for proper circular-arc graphs in logspace. *J. Discrete Algorithms*, 38–41:38–49, 2016. doi:10.1016/j.jda.2016.03.001.

[11] N. Limaye, M. Mahajan, and P. Nimbhorkar. Longest paths in planar DAGs in unambiguous log-space. *Chic. J. Theoret. Comput. Sci.*, 2010(8), 2010. doi:10.4086/cjtcs.2010.008.

[12] M. C. Lin, D. Rautenbach, F. J. Soulignac, and J. L. Szwarcfiter. Powers of cycles, powers of paths, and distance graphs. *Discrete Appl. Math.*, 159(7):621–627, 2011. doi:10.1016/j.dam.2010.03.012.

[13] M. C. Lin and J. L. Szwarcfiter. Unit circular-arc graph representations and feasible circulations. *SIAM J. Discrete Math.*, 22(1):409–423, 2008. doi:10.1137/060650805.

[14] R. M. McConnell, K. Mehlhorn, S. Näher, and P. Schweitzer. Certifying algorithms. *Comput. Sci. Rev.*, 5(2):119–161, 2011. doi:10.1016/j.cosrev.2010.09.009.

[15] J. Mitas. Minimal representation of semiorders with intervals of same length. In *Orders, algorithms, and applications (Lyon, 1994)*, volume 831 of *Lecture Notes in Comput. Sci.*, pages 162–175. Springer, Berlin, 1994. doi:10.1007/BFb0019433.

[16] M. Pirlot. Minimal representation of a semiorder. *Theory and Decision*, 28(2):109–141, 1990. doi:10.1007/BF00160932.

[17] M. Pirlot. Synthetic description of a semiorder. *Discrete Appl. Math.*, 31(3):299–308, 1991. doi:10.1016/0166-218X(91)90057-4.

[18] M. Pirlot and P. Vincke. *Semiorders*, volume 36 of *Theory and Decision Library. Series B: Mathematical and Statistical Methods*. Kluwer Academic Publishers Group, Dordrecht, 1997. Properties, representations, applications. doi:10.1007/978-94-015-8883-6.

[19] F. J. Soulignac. *On proper and Helly circular-arc graphs*. PhD thesis, Universidad de Buenos Aires, Mar. 2010. Accessed 11 October 2016. URL: http://digital.bl.fcen.uba.ar/Download/Tesis/Tesis_4660_Soulignac.pdf.

[20] F. J. Soulignac. Bounded, minimal, and short representations of unit interval and unit circular-arc graphs. Chapter I: theory. *J. Graph Algorithms Appl.*, 21(4):455–489, 2017. doi:10.7155/jgaa.00425.

[21] D. Stolee and N. V. Vinodchandran. Space-efficient algorithms for reachability in surface-embedded graphs. In *2012 IEEE 27th Conference on Computational Complexity—CCC 2012*, pages 326–333. IEEE Computer Soc., Los Alamitos, CA, 2012. doi:10.1109/CCC.2012.15.

[22] A. Tucker. Structure theorems for some circular-arc graphs. *Discrete Math.*, 7:167–195, 1974. doi:10.1016/S0012-365X(74)80027-0.