LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

INSTITUT FÜR INFORMATIK
LEHR- UND FORSCHUNGSEINHEIT FÜR
PROGRAMMIER- UND MODELLIERUNGSSPRACHEN

Interval-Based Graph Representations for Efficient Web Querying

Antonius Weinzierl

Diplomarbeit

Beginn der Arbeit:	1.11.2008
Abgabe der Arbeit:	27.4.2009
Betreuer:	Prof. Dr. François Bry
	Dr. Tim Furche
	Dr. Norbert Eisinger

Erklärung

Hiermit versichere ich, dass ich diese Diplomarbeit selbständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

München, den 27.4.2009 Antonius Weinzierl

Zusammenfassung

Diese Arbeit untersucht Erweiterungen des Continuous Image Graph (CIG) Beschriftungsschemas, welches auf der Consecutive Ones Eigenschaft von Matrizen basiert. Das CIG Beschriftungsschema ist eine effizienzerhaltende Verallgemeinerung von Baum-Beschriftungsschemata auf Graphen, d.h. Adjazenztests haben konstante Laufzeit, der Speicherbedarf pro Knoten ist konstant und zu Testen, ob ein gegebener Graph gemäß dem CIG Schema beschriftet werden kann, ist in polynomieller Zeit möglich. Das CIG Beschriftungsschema basiert auf dem Finden einer Ordnung der Knoten eines gegebenen Graphen, sodass sämtliche Kindknoten eines jeden Knotens bezüglich der gefundenen Ordnung in einem fortlaufendem Intervall liegen. Die hier vorgestellten Erweiterungen behandeln Beschriftungsschemata, welche auf mehrere Graphen anwendbar und effizienzbewahrend sind. Die Erweiterungen des CIG-Beschriftunsschemas sind derart, dass

- mehrere Intervalle pro Knoten,
- mehrere Ordnungen pro Graph,
- mehrdimensionale Intervalle pro Knoten, oder
- eine beliebige Kombination davon

erlaubt werden. Es wird gezeigt, dass diese Erweiterungen in der Tat weitere Graphen abdecken und (echte) Hierarchien von Beschriftungsschemata bilden mit steigendem Speicherbedarf und Anwendbarkeit auf immer mehr Graphen. Jedoch wird gezeigt, dass jede dieser natürlichen Erweiterungen auch ein NP-vollständiges Erkennungsproblem besitzt, d.h. der Test, ob ein gegebenes derartiges Beschriftungsschema auf einen gegebenen Graphen angewandt werden kann, ist NP-vollständig. Die Ergebnisse dieser Arbeit deuten daher stark darauf hin, dass das CIG Beschriftungsschema gerade noch eingeschränkt genug ist, um ein polynomielles Erkennungsproblem zu haben, und dass für signifikant verallgemeinerte Beschriftungsschemata sehr wahrscheinlich nur approximierende Erkennungsalgorithmen effizient sind.

Abstract

We investigate natural extensions of the Continuous Image Graph (CIG) labelling scheme, which is based on the Consecutive Ones Property of matrices. The CIG labelling scheme generalizes tree labelling schemes to many graphs yet remaining as efficient, i.e., adjacency tests run in constant time, the space requirement is constant per node, and testing whether a graph is a CIG is possible in polynomial time. The CIG labelling scheme is based on finding an ordering of the nodes of a graph such that every node has all its children in one consecutive interval under the ordering. Our extensions cover labelling schemes that are applicable to more graphs yet retain the same efficiency. We propose extensions of the CIG labelling scheme such that

- multiple intervals per node,
- multiple orderings per graph,
- multi-dimensional intervals per node, or
- any combination of the above

are allowed. We show that our extensions indeed cover more graphs and give (proper) hierarchies of labelling schemes with ever-increasing space needs, covering even more graphs. Unfortunately, however, we also show that each of these natural extensions yields an NP-complete recognition problem, i.e., testing whether a given labelling scheme of ours is applicable to a given graph is NP-complete. Our results strongly suggest that the CIG labelling scheme is just restricted enough to still be recognizable in polynomial time and that for significantly more general labelling schemes most likely only approximate recognition algorithms are efficient.

Danksagung

Ich bedanke mich ganz herzlich bei Prof. Dr. François Bry für die menschliche Betreuung, die große Unterstützung und Förderung, welche sich nicht nur auf diese Arbeit beschränkt, sondern mein gesamtes Studium betrifft.
Besonders möchte ich mich auch bei Dr. Tim Furche für die Betreuung, die unterhaltsamen Besprechungen und ganz besonders die Diskussionen auch über die Informatik hinaus bedanken.
Ebenso geht mein herzlicher Dank an Dr. Norbert Eisinger, welcher nicht nur jederzeit ein offenes Ohr für studentische Belange hat, sondern mit viel gutem Rat und ebenso unterhaltsam zum Gelingen dieser Arbeit beigetragen hat.
Außerdem geht mein Dank an Simon Brodt für die hervorragenden Diskussionen und viele Beweisideen, welche sich hier auch teilweise wiederfinden.
Ich danke meiner Schwester, meiner Mutter und meinem Vater für die Unterstützung in jeglicher Hinsicht und dafür, dass sie mir das Studieren überhaupt erst ermöglicht haben.
Ganz besonders bedanke ich mich bei Franz und Franziska Fischer für ihre Unterstützung, die sich mit Worten kaum treffend beschreiben lässt.

Inhaltsverzeichnis

1 Introduction

The act of querying data today has become a routine task of everday life. Be it while using Google or using the search feature of one of the many social network services. There is, however, much more potential in the way of (1) how we express our queries and (2) what type of data we address by the queries. The first leads to the field of query languages while the second leads to the field of data representations. In the context of the Web the second field becomes of increasing importance as real-world data often is graph shaped, like your network of friends in a social community is a web of relationships and no hierarchy of command structures. Of course, queries addressed to such data should be able to query the underlying structure of the data and allow queries like "give me the names of all friends of my friends that themselves are friends to each other". This type of query cannot be answered by Google and it cannot be answered efficiently at all – unless sophisticated data structures are employed. In this work we focus on such data structures and try to increase the class of graph data for which adjacency can be tested in constant time and constant per-node space. We focus on adjacency as it is the innermost operation of structured queries. For example the above query can be evaluated alone by looking at adjacent elements of the "friend-of" relation. Thus our goal is that arbitrary, graph-shaped data can be queried as fast as tree-shaped data. Note that the change from tree data to arbitrary graph data is further accommodated by the standardization of data formats genuinely representing graphs, e.g., the ID/IDREF facility of XML, or the inherently graph-shaped data-model underlying the W3C recommendation SPARQL.

In our approach to querying graph-shaped data, we value three properties most: First, the runtime cost of the adjacency test must be as efficient for graphs as it is for trees. Second, the storage or space cost of an element of the data set must be as efficient for graphs as it is for trees. Third, the cost of applying our solution to a database must be as fast as possible.

For tree-shaped data the best data structures (e.g. [13]) achieve the following properties:

- testing whether one node is adjacent to another is possible in constant time
- the space required to store which nodes are adjacent is constant for every node
- applying such a data structure to a tree is possible in linear time

In this thesis we investigate data structures that have the same properties, but work also on graph-shaped data.

1.1 The CIG Labelling Scheme

We base our work on a recently proposed graph labelling scheme [10], called the Continous Image Graph (CIG) labelling, which is based on the well-studied Consecutive Ones Property (C1P) of adjacency matrices and generalizes interval-based tree-labelling such as the pre-/post-encoding [13]. In this work a graph labelling is the assignment of a label to every node of a directed graph where the actual shape of the label depends on the employed labelling scheme. The CIG labelling scheme is an apt starting point as it is applicable to directed graphs such that:

- Testing adjacency requires only constant time per node.

- The space requirement per node is constant.
- Labelling a graph, i.e., applying this data structure to a graph, is possible in quadratic time in the order of its nodes.

By that the CIG labelling scheme already fulfils our objectives. However, it is only applicable to certain graphs. So our primary mission is to identify extensions of the CIG labelling scheme which are applicable to more graphs while the above properties of the CIG labelling scheme are retained (or relaxed as little as possible). Note that we consider only directed graphs and therefore drop the adjective when speaking of directed graphs.

Intuitively, the CIG labelling scheme puts all nodes of a graph into a linear ordering σ (i.e., a serial numbering) such that each node has all its children nodes (i.e., nodes connected by an outgoing edge) in one consecutive interval. By that the children of each node can be represented and stored by just that interval, i.e., by two numbers. As the purpose of the interval is to serve as a complete description of all children of a node, the interval assigned to each node contains exactly all children and no other node. Testing whether node v' is a child of v then requires to look-up the position $\sigma(v')$ of v' in the linear ordering σ of the graph's nodes and checking if $\sigma(v')$ is inside the childr interval assigned to v. Clearly, this procedure is independent of the size of the graph and independent of the number of children of v. This is also true for the space requirement as for every node only its position in the ordering and the interval for its children needs to be stored. In effect each node is fully described by three numbers. Thus the label assigned to every node exactly consists of those three numbers. The last of the above properties — an algorithm which labels a given graph in linear time in the order of the number of edges of the graph — can be achieved by an algorithm from [4].

To simplify terminology we use the acronym "CIG" for two meanings:

1 A "CIG" is a Continous Image Graph, i.e., a graph for which a CIG labelling exists.

2 "CIG" denotes the class of all graphs that are Continous Image Graphs in the first sense.

Thus for a graph G, "G is a CIG" and "$G \in$ CIG" mean the same. We apply the same to labelling schemes introduced later. Thus "every CIG is a 2-CIG" and "CIG is a subclass of 2-CIG" mean the same. Observe that every correct algorithm which labels graphs according to a given labelling scheme implicitly solves the recognition problem of the labelling scheme, e.g., the labelling algorithm for CIG gives for every graph G whether $G \in$ CIG as it returns a CIG labelling for G if one exists.

Example 1. Figure 1 shows a directed graph G_{ex} on the left. Obviously G_{ex} is no tree. On the right the ordering $\sigma = (F, A, B, C, D, E, H, G, J)$ of the nodes of G_{ex} is shown. The ordering σ assigns 1 to F, i.e., $F \mapsto 1$, $A \mapsto 2$, $B \mapsto 3$, …, $J \mapsto 9$. The intervals for every node are indicated by $\vdash\!\!\!-\!\!\!-\!\!\!-\!\!\dashv$ preceded by the name of the node to which this interval belongs. Thus the children of J form the interval from 1 to 4, denoted with $[1, 4]$, those of H the interval 2 to 6, i.e., $[2, 6]$, and those of G the interval 7 to 7, i.e., $[7, 7]$, as G has only one child.

As stated above, the CIG labelling scheme allows us to extend the advantageous properties of tree labelling schemes to non-tree graph-data. The major disadvantage, however, is the

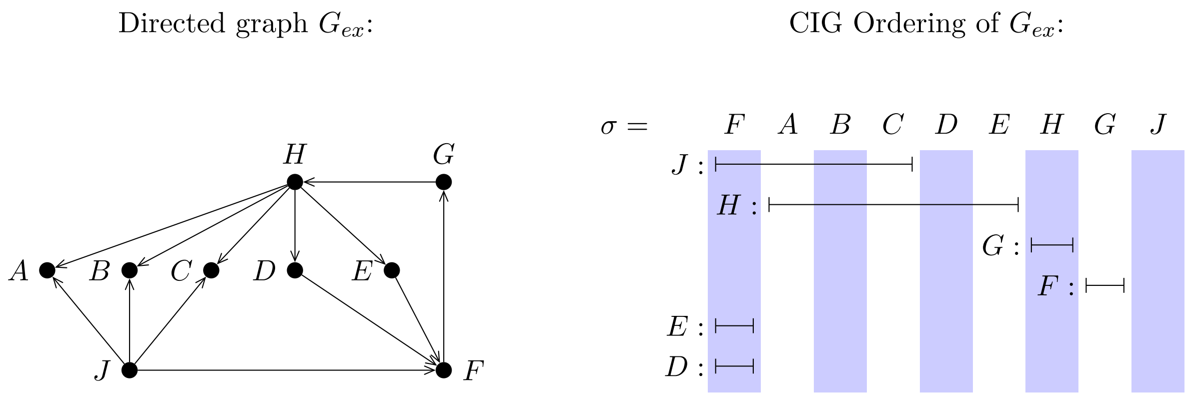


Figure 1: (cf. *Example* 1) Graph G_{ex} on the left and an ordering $\sigma = (F, A, B, C, D, E, H, G, J)$ of its nodes on the right. Every node of G_{ex} has all its children in one interval.

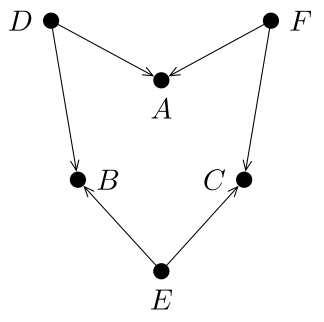


Figure 2: (cf. *Example* 2) A graph which cannot be labelled according to the CIG scheme.

fact that even small graphs may not be represented by any CIG labelling as there is no linear ordering where the children of all nodes are continuous.

Example 2. The graph in Figure 2 is no CIG, i.e., it cannot be labelled according to the CIG scheme. This is due to D, E, and F whose children A, B, and C form a cyclic arrangement, as every of the parent nodes has two from A,B,C as children but not the third one. For any linear ordering this cycle has to be broken: with $\sigma = (A, B, C, D, E, F)$ F has its children A and C not in one consecutive interval, with $\sigma' = (A, C, B, D, E, F)$ D has its children not in one interval, with $\sigma'' = (B, A, C, D, E, F)$ E has its children not in one consecutive interval. For any ordering of A,B, and C there is one node whose two children are the first and last node, but not the one in the middle. Thus this graph cannot be labelled according to the CIG scheme.

This shows that for a wider applicability the constraints of the CIG scheme have to be weakened. It is the main goal of this thesis to find CIG-based labelling schemes, which still have the properties of constant time adjacency tests per node as well as constant space requirements per node, but are applicable to more graphs than the CIG labelling.

1.2 Contribution: Extending the CIG Labelling Scheme

This thesis focuses on extensions of the CIG labelling scheme, such that those features of the CIG labelling which allow efficient query evaluation are retained while the extended scheme is applicable to more graphs than the CIG labelling. Our main contributions to this are:

- The identification of the k-CIG labelling schemes (and the class of graphs it is applicable to), which is the class of labellings with multiple intervals per node, briefly introduced in Section 1.2.1 and investigated in Section 4.
- The identification of the (d, d', k)-CIG labelling schemes, which is the most general class of labelling schemes including multi-dimensional intervals per node, briefly introduced in Section 1.2.2, and investigated in more detail in Section 5.
- The proof that testing wether a graph is a k-CIG is NP-complete in Section 4.2.
- The proof that testing wether a graph is a (d, d', k)-CIG is NP-complete in Sections 5.2 to 5.4.

The overall result of those proofs is that with just one step in any direction away from the $(1, 1, 1)$-CIG, viz. the original CIG labelling scheme, the recognition problem for the reached labelling scheme is already NP-complete. Specifically the decision problems of the labelling scheme of $(2, 1, 1)$-CIG, $(2, 2, 1)_{strong}$-CIG, $(2, 2, 1)_{weak}$-CIG, and $(1, 1, 2)$-CIG are NP-complete. Thus, unfortunately, any slight extension of the CIG labelling scheme in any natural way exhibits already NP-complete recognition problems.

1.2.1 Introducing the k-CIG Labelling Schemes

The class of k-CIG labelling schemes consists of natural extensions of the CIG labelling scheme: Where for a CIG each node may use exactly one interval for all its children, a k-CIG labelling scheme allows every node to have up to k intervals for its children. This defines an infinite number of labelling schemes as k is an arbitrary, but fixed value. So the class of k-CIG labelling schemes consists of the 2-CIG labelling scheme, the 3-CIG labelling scheme, the 4-CIG labelling scheme, and so on.

Example 3. Reconsider the non-CIG from Example 2. In Figure 3 the same graph along with the ordering $\sigma = (D, E, F, A, B, C)$ is shown. The ordering σ allows that the children of each node form at most two intervals. The intervals required here are $I_D = \{[4, 5]\}$, $I_E = \{[5, 6]\}$, and $I_F = \{[4, 4], [6, 6]\}$, so F is the only node that needs two intervals in σ for its children.

Since we allow up to k intervals each k_1-CIG labelling scheme is also a k_2-CIG labelling scheme if $k_1 \leq k_2$, i.e., the labelling schemes contained in k-CIG form a hierarchy where for each graph G with $G \in k$-CIG it holds that $G \in (k + i)$-CIG for $k, i \in \mathbb{N}$. In 4.1 we show that this hierarchy indeed is a proper one. Also note that for any fixed value of k every k-CIG labelling scheme has the properties of constant time adjacency tests and constant space requirement per node. This holds as for a fixed k there are at most k intervals to check for adjacency, and each node is described by $1 + 2k$ numbers, viz. the position in the ordering and 2 borders for every of the k intervals.

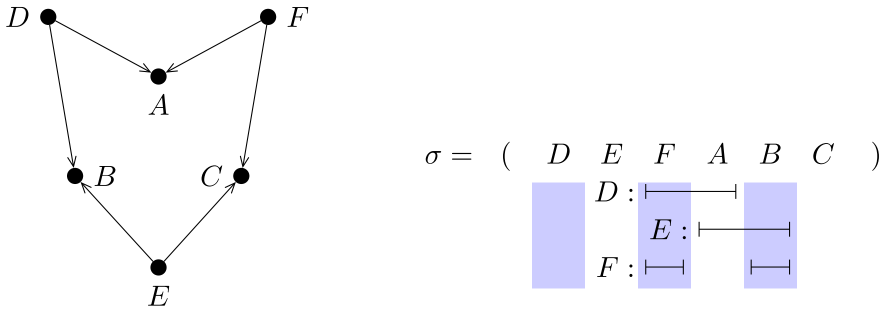


Figure 3: (cf. *Example* 3) A 2-CIG (left) with a possible ordering σ (right). Node F requires two intervals.

1.2.2 Introducing the (d, d', k)-CIG Labelling Schemes

The class of (d, d', k)-CIG labelling schemes is a further generalization of the CIG labelling. It covers several extensions of the CIG scheme: Multiple orderings per graph, multi-dimensional intervals per node, and multiple intervals per node. Due to the latter it contains the class of k-CIG labelling schemes. Note that the multi-dimensional intervals are located in the space of multiple orderings: Each dimension consists of one ordering of all nodes of the input graph. By that the first two extensions, multiple orderings and multi-dimensional intervals, directly relate to each other. The term (d, d', k)-CIG is read as: A labelling scheme which uses d dimensions/orderings, intervals which extend over at most d' dimensions per node, and at most k of such intervals per node. Note that all values for d, d', and k are arbitrary, but fixed and always $d' \leq d$, i.e., no interval can use more dimensions than available.

Example 4. Reconsider the non-CIG G from Example 2 again. In Figure 4 the same graph is shown together with a two-dimensional ordering $\Sigma = \{1{:}(F, D, E, B, A, C) = \sigma_1, 2{:}(F, D, E, A, B, C) = \sigma_2\}$ (on the right). All nodes of the graph are put on their position in the 2-dimensional space according to the orderings. At the lower right the intervals for D, E, and F are drawn and oriented according to the dimension they lie in. Given the ordering Σ with the ordering σ_1 in dimension 1 and σ_2 in dimension 2, the children of each parent node of G form one interval in one of the one-dimensional orderings. For instance the children of node D form the interval $[4, 5]$ in dimension 2 (in σ_2), children of E the interval $[5, 6]$ in σ_2, and finally children of F form the interval $[5, 6]$ in σ_1. Thus G is a $(2, 1, 1)$-CIG as two orderings suffice and every parent has all its children in one interval which extends over one dimension only.

If we only extend the CIG labelling scheme by multiple orderings, we obtain the $(d, 1, 1)$-CIG labelling schemes, if we also consider multi-dimensional intervals, we obtain the $(d, d', 1)$-CIG labelling schemes, and if we only extend the CIG labelling by multiple intervals, we obtain the class of $(1, 1, k)$-CIG labelling schemes, i.e., the class of k-CIG labelling schemes. Thus (d, d', k)-CIG indeed is the most general labelling scheme considered in this thesis subsuming all natural extensions. Again, note that for fixed values of d, d', and k a (d, d', k)-CIG labelling scheme provides constant time adjacency tests and only requires constant space per

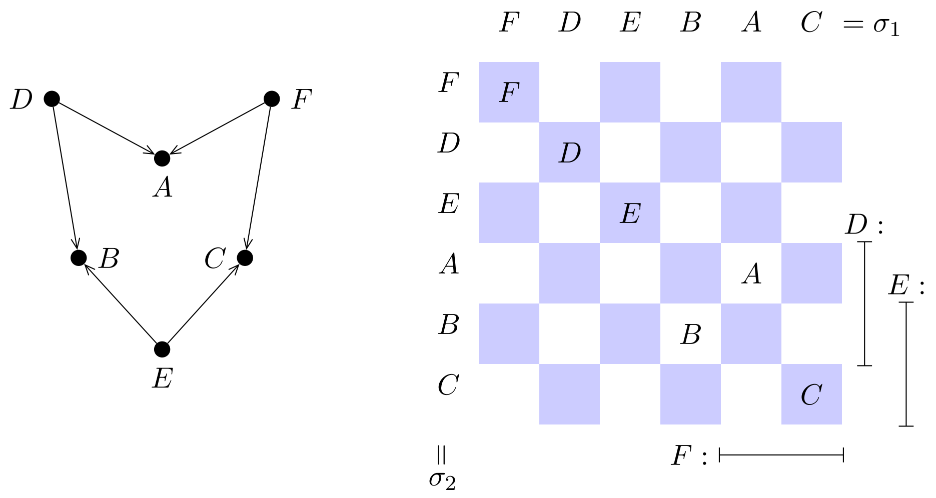


Figure 4: (cf. *Example* 4) A non-CIG and a 2-dimensional ordering of its nodes.

node. This holds as the adjacency test only has to check if the d coordinates of a presumptive child node are inside the at most $d' \cdot k$ borders of the at most k intervals. Furthermore every node is fully described by at most $d + 2d \cdot' k$ numbers, viz. for every of the d orderings one position and for every of the k intervals at most $2d'$ borders, i.e., two borders per dimension. As all these values are constant for fixed d, d, and k, each (d, d', k)-CIG labelling scheme allows constant time adjacency tests and only requires constant space per node.

In addition to that we propose a further discrimination of (d, d', k)-CIG: Assume that the children of node v are contained in the two dimensional interval I_v which ranges from 2 to 4 in dimension 1 and from 5 to 7 in dimension 2, denoted $I_v = (1{:}[2, 4], 2{:}[5, 7])$. So a 2-dimensional interval always consists of 2 one-dimensional interval boundaries. Testing if a node v' is adjacent to v is done by testing if v' is contained in the above 2-dimensional interval. However, there are two possible ways how to test: The first testing method succeeds if all positions of v' are in the respective one-dimensional intervals of v (we call this the strong version) while the second succeeds if one of the positions of v' is in one of the respective intervals of v (called the weak version). So we further identify the classes of $(d, d', k)_{strong}$-CIG and $(d, d', k)_{weak}$-CIG, where strong means that intersection is applied among the dimensions while weak means that union is applied among the dimensions.

Example 5. Figure 5 illustrates the difference between the weak and strong labelling schemes. It shows two different graphs on top, which both can be labelled according to a (d, d', k)-CIG labelling scheme. For both graphs the labelling consists of a 2-dimensional ordering $\Sigma = \{\sigma_1 = (A, B, C, D), \sigma_2 = (D, A, B, C)\}$ and a 2-dimensional interval $I_D = \{1{:}[2, 3], 2{:}[2, 3]\}$ for the children of node D. For the left graph this labelling is a correct one if its interpretation is that of weak which results in the dimensions of the interval of D being united, thus D having A, B, and C as children. For the right graph the very same labelling is a correct one if its interpretation is that of strong which results in the dimensions of the interval of D being intersected, thus D having B as its only child. This shows that the labelling actually represents two different graphs depending on whether the labelling is read as a

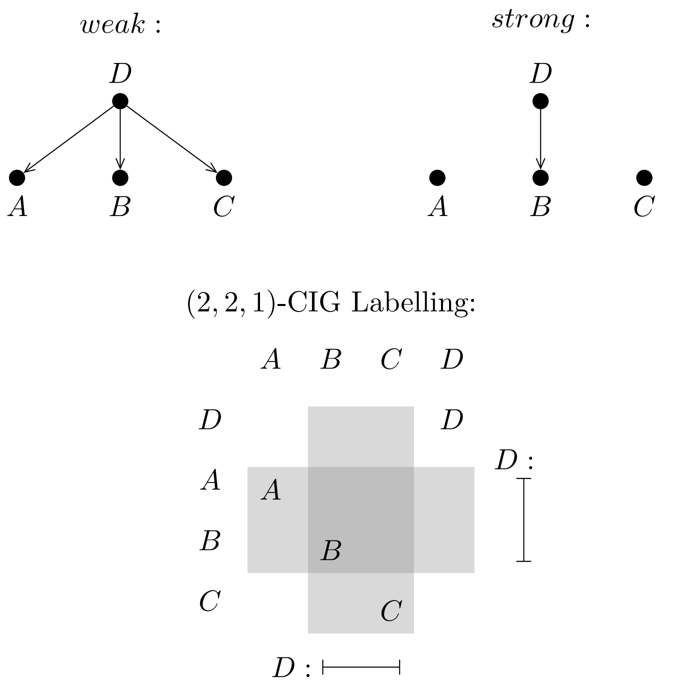


Figure 5: (cf. *Example* 5) Two different graphs having the same labelling if the $(2, 2, 1)_{weak}$-CIG scheme is applied to the left and the $(2, 2, 1)_{strong}$-CIG scheme is applied to the right. The dark gray rectangle is the 2-dimensional interval interpreted as strong while the gray cross is the same interval interpreted as weak.

$(2, 2, 1)_{strong}$-CIG or a $(2, 2, 1)_{weak}$-CIG. Note that this is unusual in so far as all previous labelling schemes result in a labelling that uniquely identifies a graph, i.e., each labelling scheme induces an injective function. Here the labelling scheme is injective only if the information is given whether the labelling scheme is weak or strong.

1.2.3 Relationships between CIG-based Labelling Schemes

As already mentioned, the k-CIG labelling schemes form a proper hierarchy, i.e., k_1-CIG is a subclass of k_2-CIG, for $k_1 \leq k_2$. Note that the k-CIG labelling scheme with $k = 1$ is applicable to the same class of graphs as the original CIG, i.e., 1-CIG = CIG.

The (d, d', k)-CIG classes also form hierarchies, but their relationships are more involved, cf. Section 5.5. Note that the $(1, 1, k)$-CIG = k-CIG and $(1, 1, 1)$-CIG = CIG.

Additionally there are some interrelations between the available number of dimensions and the available number of intervals. Consider, for example, a graph where one node needs $k+1$ intervals of d dimensions while all other nodes need only up to k d-dimensional intervals. So this graph is a $(d, d, k + 1)$-CIG. By addition of another dimension, i.e., another ordering, the one node requiring $k + 1$ intervals can have all its adjacent nodes in one interval in the new ordering. So this graph then would be a $(d+1, d, k)$-CIG. In Section 5.5 all investigated relationships among our extensions are discussed and summarized.

1.2.4 Each k-CIG Recognition Problem is NP-complete

The recognition problem of k-CIG, i.e., the test wether a given graph is a k-CIG, is NP-complete for $k \geq 2$. The recognition problem of k-CIG is similar to a problem of DNA mapping, which arose in the mid 1990s during the Human Genome Project. Goldberg et al.[12] showed that their problem of DNA mapping in the presence of so called chimericism is NP-complete. We show in Section 4 that this problem is equivalent to the recognition problem of 2-CIG. Furthermore a generalization of this problem which translates to k-CIG for any fixed $k \geq 2$ is given in [12]. The result for the generalized problem is also proven in [12] by a reduction from 3SAT. Our reduction to k-CIG then amounts to a matrix transposition.

1.2.5 Each (d, d', k)-CIG Recognition Problem is NP-complete

The major part of this thesis is dedicated to several proofs showing that the most interesting subclasses of (d, d', k)-CIG yield NP-complete recognition problems. In the first step the recognition problem of the $(2, 1, 1)$-CIG labelling scheme is proven to be as hard as the NP-complete problem of Consecutive Ones Matrix Partition [11]. The second proof shows that the recognition problem for each $(d, 1, 1)$-CIG with fixed $d \geq 3$ is NP-complete. This is shown by a reduction from d-Colorability. The third step reduces the recognition problem of $(d, 1, 1)$-CIG to the decision problem of $(d, d', k)_{weak}$-CIG for $d, d' \geq 2$ and any value of k. Finally the labelling schemes of $(d, d', 1)_{weak}$-CIG and $(d, d', 1)_{strong}$-CIG are proven to be of equal hardness, as a graph $G \in (d, d', 1)_{weak}$-CIG if and only if $\overline{G} \in (d, d', 1)_{strong}$-CIG where $\overline{G}$ denotes the complement graph of G.

The rest of this thesis is organized as follows: In Section 2 the common definitions used in this thesis are given. Section 3 summarizes related work together with the origins of the CIG labelling scheme. In Section 4 the k-CIGlabelling scheme is investigated and shown to induce proper hierarchies. Section 5 covers the (d, d', k)-CIG labelling schemes. It contains the reductions to $(d, 1, 1)$-CIG, $(d, d', 1)$-CIG, and $(d, d', k)_{weak}$-CIG proving the NP-completeness of their recognition problems and the investigated hierarchies among the labelling schemes. Finally in Section 6 all results are briefly summarized and several directions for future work are sketched.

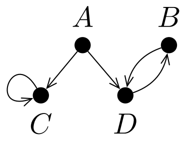


Figure 6: (cf. *Example* 10) Small graph G_{ex} (nodes denoted by black dots, edges by arrows).

2 Preliminaries

This section contains common definitions that are used throughout this work. The first part recalls some standard notions on graphs and adjacency matrices. The second part introduces orderings, intervals, and the original CIG labelling scheme.

2.1 On Graphs and Adjacency Matrices

Graphs We use the standard notion of finite directed graphs, and define parent, child, and complement graphs in the usual way.

Definition 6 (Graph)**.** A graph $G = (V, E)$ is a pair consisting of a finite set of nodes V and a set of edges $E \subseteq V \times V$. In the following we consider directed graphs only, i.e., $(v, v') \in E$ is different from $(v', v) \in E$. We further omit the adjective directed when speaking of directed graphs.

Definition 7 (Complement Graph)**.** Let $G = (V, E)$ be a graph. The complement of G, denoted $\overline{G}$, is defined as $\overline{G} = (V, \overline{E})$ where $\overline{E} = \{(v, v') \in V \times V \mid (v, v') \notin E\}$.

Definition 8 (Parent)**.** Let $G = (V, E)$ be a graph and $(v, v') \in E$, then v is the parent of v' and we call v a parent in G. The set of all parent nodes of a graph $G = (V, E)$ is denoted by $parents(V, E) = \{v \in V \mid (v, v') \in E\}$.

Definition 9 (Child)**.** Let $G = (V, E)$ be a graph and $(v, v') \in E$, then v' is called a child of v. The set of all child nodes of a node is denoted by $children(v) = \{v' | (v, v') \in E\}$.

Example 10. Figure 6 shows the graph $G_{ex'} = (V, E)$ with nodes $V = \{A, B, C, D\}$ and edges $E = \{(A, C), (A, D), (C, C), (B, D), (D, B)\}$. Thus C and D are children of A, A and B are both parents of D, and C is both its own parent as and its own child.

The central focus of our work is on complexity results for different graph labelling algorithms. As these algorithms are not applicable to every graph, every such labelling algorithm implicitly answers the recognition problem of its labelling scheme, i.e., the algorithm answers if a given graph can be labelled according to the scheme. We call this the recognition problem.

Definition 11 (Recognition Problem)**.** Let P be a non-trivial property of graphs, i.e., $\{G \mid P(G)\} \neq \emptyset \wedge \{G \mid \neg P(G)\} \neq \emptyset$. The problem of deciding whether P holds for a graph G is called the recognition problem of P. We say that P is NP-complete for short if we mean that the recognition problem of P is.

Matrices We use the standard definitions of matrices and adjacency matrices, but further define the operations of union, intersection and complement on adjacency matrices.

Definition 12 (Matrix)**.** Let K be a set and $m, n \in \mathbb{N}$, then $A \in K^{m \times n}$ denotes a matrix with m rows and n columns. $(a_{ij}) \in K$ denotes the entry of the i-th row and the j-th column of A.

Definition 13 (Adjacency matrix)**.** Let $G = (V, E)$ be a graph. Then $A \in \{0,1\}^{|V| \times |V|}$ is an adjacency matrix of G w.r.t. an ordering σ if $(a_{\sigma(v)\sigma(v')}) = 1 \iff (v, v') \in E$. We write $A = \mathit{adj}(G)$ if A is the adjacency matrix of G under the standard ordering σ_G of the nodes of G (cf. Definition 16). Thus $A = \mathit{adj}(G)$ is well-defined and in the following we identify G with its (unique) adjacency matrix A. We say that an adjacency matrix A has a certain property P meaning that P holds for G where $A = \mathit{adj}(G)$.

Definition 14 (Column Permutation)**.** Let $G = (V, E)$ be a graph, $A = \mathit{adj}(G)$ its adjacency matrix, and $\sigma : V \to \mathbb{N}_{|V|}$ an ordering of its nodes. By $A' = \sigma(A)$ we denote the adjacency matrix which has its columns permuted according to σ, i.e., $(a'_{ij}) = (a_{i\sigma(j)})$. In the following we write permutation of a matrix meaning the column permutation of a matrix.

Definition 15 (Union, Intersection, Complement)**.** Let A and $A^1, \dots, A^d$ be adjacency matrices of the same size, i.e., $A, A^1, \dots, A^d \in \{0,1\}^{n \times n}$ for $n \in \mathbb{N}$.

- The complement $\overline{A} \in \{0,1\}^{n \times n}$ of A is the matrix where 0's and 1's are switched, i.e., $(\overline{a_{ij}}) = 1 - (a_{ij})$. It follows that two nodes in A are adjacent if and only if the same nodes are not adjacent in $\overline{A}$.
- The union $\bigcup_{\ell=1}^{d} A^\ell$ is defined as $(a'_{ij}) = \begin{cases} 1 & \text{if } \exists A^\ell : (a^\ell_{ij}) = 1 \\ 0 & \text{otherwise} \end{cases}$.
- The intersection $\bigcap_{\ell=1}^{d} A^\ell$ is defined as $(a'_{ij}) = \begin{cases} 1 & \text{if } \forall A^\ell : (a^\ell_{ij}) = 1 \\ 0 & \text{otherwise} \end{cases}$.

2.2 On Orderings and Intervals

Orderings A common property of all labelling schemes investigated in this thesis is that all of them seek for orderings of nodes of a given graph.

Definition 16 (Ordering)**.** Let $G = (V, E)$ be a graph. An ordering of the nodes V of G is a bijective function mapping every node $v \in V$ to its position $\sigma(v)$, i.e., $\sigma : V \to \mathbb{N}_{|V|}$ with $v \mapsto \sigma(v)$. Every ordering induces a total, linear order relation on V. We write orderings in tuple notation, i.e., the ordering σ mapping A to 2, B to 3, and C to 1 is written $\sigma = (C, A, B)$. The inverse ordering σ^{-1} of σ is defined as usual, i.e., $\sigma^{-1}(i) = v \iff \sigma(v) = i$.

In the following we assume that every graph $G = (V, E)$ is associated with an arbitrary but fixed ordering σ_G, called the standard ordering of G. We further assume that this ordering is used implicitly whenever a numerical value for a node is required, e.g., in the case of adjacency matrices we address rows and columns directly by $v \in V$ instead of writing $\sigma_G(v)$.

An essential property of orderings, which is used throughout the later proofs, is the concept of neighbourhood of nodes in a given ordering.

Definition 17 (Neighbour)**.** Let $G = (V, E)$ be a graph and σ an ordering of V. Two nodes $v, v' \in V$ are neighbours in σ if there is no other node between them in the ordering, i.e., $|\sigma(v) - \sigma(v')| = 1$.

For the different classes of CIG-based labelling schemes the definition of neighbourhood can be weakened with respect to the child relation of a graph:

Definition 18 (Consecutive Children)**.** Let $G = (V, E)$ be a graph, σ an ordering of V, and $p \in V$ some parent with $children(p) = \{c_1, \ldots, c_n\}$. We say that c_i and c_j are consecutive in σ if and only if there are only children of p between the positions of c_i and c_j, i.e., $\forall \sigma(c_i) \leq \sigma(x) \leq \sigma(c_j) : (p, x) \in E$.

Example 19. Reconsider the graph $G_{ex'}$ from Example 10. $\sigma = (A, D, C, B)$ is an ordering of the nodes V of $G_{ex'}$. Here node D has A and C as neighbours while A has only one neighbour, D. Furthermore the children of A, namely C and D, are consecutive in σ. This is also true for B which only has one child, thus all its children are consecutive in σ (as would be in any other ordering).

Intervals and the CIG labelling scheme All CIG-based labelling schemes use the notion of intervals which allows all children of a node to be fully described by one interval.

Definition 20 (Interval)**.** A (one-dimensional) interval $int \in \mathbb{N} \times \mathbb{N}$ is an ordered pair of the beginning and ending of the interval, i.e., $int = [begin, end]$. The empty interval is denoted by $[-]$ and an interval containing only one element x is written $[x]$. In the presence of an ordering we further allow that intervals are circular, which is the case if $begin > end$ for an interval $[begin, end]$. For example the interval $[15, 3]$ is circular and contains every node except those between 3 and 15. In the following we write $x \in [begin, end]$ meaning that x is inside the interval w.r.t. some ordering $\sigma : V \to \mathbb{N}_{|V|}$, formally:

$$x \in [b, e] \iff \begin{cases} b \leq x \leq e & \text{if } b \leq e \\ x \leq b \vee e \geq x & \text{if } b > e \wedge x \leq |V| \end{cases}$$

All of our intervals are closed, i.e., the borders b and e of an interval $[b, e]$ are inside the interval.

We say that an interval $[b, e]$ covers the children of a node p under an ordering σ if all children of p are inside the interval, formally: If $(x \in children(p) \iff \sigma(x) \in [b, e])$ then $[b, e]$ covers all children of p under σ.

Example 21. Recalling Example 19 with the ordering $\sigma = (A, D, C, B)$ of the nodes of the graph $G_{ex'}$. The interval $[2, 3]$ covers all children of node A.

Definition 22 (Continuous Image Graph (CIG))**.** A graph $G = (V, E)$ is a CIG, i.e., $G \in CIG$ if and only if there is an ordering σ of the nodes V, such that the children of every node $v \in V$ are covered by one interval, formally:

$$\exists \sigma : \forall v \in V : \exists [b, e] : (v, v') \in E \iff \sigma(v') \in [b, e]$$

Note that this definition deviates from [10] where circular intervals are not allowed. Yet the same properties hold:

- Testing if v' is a child of v is possible in constant time (our definition only requires the additional check whether the interval of v is circular).
- The space required per node is constant (the position in σ and the interval for the children are stored per node, i.e., only three numbers).
- The CIG labelling can be applied to a graph in time linear to the number of its edges (see [23]). Thus the recognition problem of CIG is polynomial.

A definition which characterizes the same property as the CIG labelling is given in [20]. It is interesting as this property is defined purely on set theoretic terms and also because this definition gives some insights to the CIG.

Definition 23. A family $\mathfrak{M}$ of subsets of a finite set X is linear if there is a sequence of elements of X such that

I every $x \in X$ occurs in the sequence exactly once,

II every $M \in \mathfrak{M}$ appears as a segment, i.e., a set of $|M|$ consecutive terms of the sequence.

The translation to our vocabulary is: The set X corresponds to the set of nodes V and $\mathfrak{M}$ is a collection of all child sets, i.e., every $M \in \mathfrak{M}$ represents the set *children*(v) of a node $v \in V$. The sequence of elements of X by condition I is a total ordering of V, i.e., our ordering σ. Condition II then states that all children of each parent must occur in one consecutive interval.

From this definition we draw an interesting observation that will simplify much in this thesis. In definition 23 all that matters to the sequence is the family of subsets. In our vocabulary: All that matters to the ordering σ is the sets of child nodes. In this definition and in the definition of CIG it does not matter to whose parent a set of children actually belongs. Thus each CIG ordering for a graph G is applicable to any graph G' which has the same sets of children. So the intervals for G and G' are the same, but they are assigned to different parents in G than in G'. By that the CIG labelling defines some kind of equivalence relation under child sets. Furthermore G and G' not even need to be isomorphic as long as their sets of children are the same.

Effects of that are visible throughout the graphs we use in our proofs: First, all constructed graphs are “flat” in the sense that no parent node is child of another node. Second, while searching for an ordering σ of a graph it is very simple to position nodes which are no child of any other node, i.e., the positioning of all of our constructed parent nodes: They are all put on the lower (or upper) end of σ in an arbitrary order. As they are no child of any other node, their position does not matter to any other node and when placed at the lower (or upper) end, they do not interact with positions of the other (child) nodes. This is the reason why we mostly ignore the positions of parent nodes throughout the following examples and proofs.

Observe that for any graph G which is no CIG it holds that every other graph containing G is no CIG either. This property also holds for any extension introduced later.

Proposition 24. Let the graph $G = (V, E)$ be a strongly connected component of the graph $G' = (V', E')$. If G is no CIG (or k-CIG or (d, d', k)-CIG, resp.) then G' is no CIG (or k-CIG or (d, d', k)-CIG, resp.) either.

Proof. As G is a strongly connected component in G' it holds for every node $v \in V$ that its set of children is the same in G' as in G. Assume that $G \notin$ CIG (or k-CIG or(d, d', k)-CIG, resp.), i.e., it holds for any ordering of V that there is one node which can not cover its children with one interval (or multiple intervals or multi-dimensional intervals, resp.). We show by contradiction that G' cannot be a CIG (or k-CIG or (d, d', k)-CIG, resp.). Assume G' is a CIG (or k-CIG or (d, d', k)-CIG, resp.), then there exists an ordering σ (or multidimensional ordering Σ, resp.) such that all nodes $v \in V'$ can cover their children with one interval (or multiple intervals or multi-dimensional intervals, resp.). As $V \subseteq V'$ and the set of children for every node $v \in V$ are the same in G' as in G, it follows that σ (or Σ, resp.) also is an ordering for G. This, however, contradicts the assumption that G is no CIG (or k-CIG or (d, d', k)-CIG, resp.). Thus G' cannot be a CIG (or k-CIG or (d, d', k)-CIG, resp.). □

Corollary 25. Every $G \notin$ CIG (or k-CIG or (d, d', k)-CIG,resp.) induces an infinite number of graphs which are no CIG (or k-CIG or (d, d', k)-CIG, resp.).

3 Related Work

This section summarizes research results which are directly related to the CIG labelling scheme. This section specifically covers

- the origins of the CIG labelling, especially the Consecutive Ones Property (C1P) on adjacency matrices.
- properties on adjacency matrices, which generalize the C1P and are similar to our extensions.
- interval graphs, which use intervals on the real line, but employ ideas similar to our extensions of the CIG labelling.
- applications in network routing, which use labelling schemes close to the CIG labelling.

Origins of the CIG labelling The CIG labelling scheme has its foundation in research on interval graphs and algorithms testing whether a given graph is an interval graph. For that a property on $\{0, 1\}$ matrices, i.e., adjacency matrices has been defined, namely the Consecutive Ones Property (C1P), which is directly related to interval graphs. In [4] a linear algorithm testing whether a given $\{0, 1\}$ matrix has the C1P is given. This effectively yields an algorithm testing for interval graphs. Although the field of interval graphs and linear testing algorithms for the C1P have drawn a lot of attention, it was not before [10] that the C1P and its linear algorithm were first employed for a constant space and time labelling for graphs. By that the C1P is an alternate definition of the properties required by the CIG labelling scheme, but the C1P is defined on $\{0, 1\}$ matrices instead of graphs:

Definition 26 (Consecutive Ones Property (C1P))**.** A matrix A of 0's and 1's is said to have the Consecutive Ones Property if and only if there is a permutation of the columns such that in every row of the permuted matrix all 1's are consecutive.

Observe that the column permutation is an ordering of the nodes of G with A being the adjacency matrix of G. The constraint that all 1's of the permuted matrix are consecutive in every row effectively is the same as the constraint that every node covers all its children by one interval, i.e., the positions of the first and last 1 of the consecutive 1's of each row of the permuted matrix define the beginning and end of the interval of each parent node in G. In [4] the first algorithm is presented which runs in time linear to the number of edges whose result is a permuted adjacency matrix, i.e., a matrix where in each row all 1's are consecutive. By that the CIG labelling of a graph is computable in linear time, if no interval is circular.

Several modifications to the original algorithm have been proposed, among those [23] and [16], which give alternative algorithms to test whether an adjacency matrix has the C1P. Moreover the algorithm in [23] tests for a slightly generalized form of the C1P, namely the Circular Ones Property. The Circular Ones Property allows intervals on the left and right end of a matrix to "wrap around", i.e., it allows circular intervals. In other words in every row either the 1's or the 0's are consecutive. Computing a permutation of a matrix so that the result has the Circular Ones Property is possible in linear time. By that the CIG labelling can be applied in linear time also.

There are many other applications of the C1P, most prominently in detecting interval graphs. Interval graphs are used in traffic light control, interval temporal logics, physical mapping of DNA, scheduling problems, radiation therapy, and more [22].

Properties on adjacency matrices Similar to one of our extensions of the CIG labelling, namely k-CIG, is the NP-complete problem of Consecutive Block Minimization (CBM) from [17]. Note that k-CIG is a class of labelling schemes while CBM is a property of matrices. Nevertheless the underlying structural property of k-CIG which is also visible on adjacency matrices is similar to the CBM property. The definition of CBM according to [11] is:

Definition 27 (Consecutive Block Minimization (CBM))**.** Given an $n \times n$ matrix A of 1's and 0's and a positive integer k. Is there a permutation of the columns of A that results in a matrix B having at most k consecutive 1's, i.e., having at most k entries b_{ij} such that $b_{ij} = 1$ and either $b_{ij} = 0$ or $j = n$?

The difference between k-CIG and CBM is the way of counting k. While the k-CIG counts the number of consecutive blocks per row, i.e., per node, the CBM counts the overall number of consecutive blocks. The problem remains NP-complete even if the number of 1's per row is restricted to 2 [14]. Recently it has been shown in [15] that the CBM problem is 1.5 approximable. This means that there is a polynomial algorithm determining a value for k which is not larger than 1.5 times the optimum k.

Although the CBM problem has promising results it is important to see that a labelling scheme based on CBM lacks the property of constant time adjacency testing as well as the constant space requirement per node. This is due to the CBM minimizing the *overall* number of consecutive blocks. So it is possible that the minumum number of blocks is achieved if all but one node have a very low number of blocks while the last node has a very high number of blocks. There is no other limit to the number of blocks of this last node other than the overall number of blocks.

Other algorithms have been proposed for cases where a matrix does not have the C1P, but is close to having it. These algorithms either try to find the largest submatrix which has the C1P, try to find a partitioning of the matrix such that both partitions independently of each other have the C1P, or try to find the minimum number of edges which have to be added such that the resulting matrix has the C1P. These problems are known as Consecutive Ones Submatrix, Consecutive Ones Matrix Partition, and Consecutive Ones Matrix Augmentation. Although all three problems are NP-complete [11], polynomial approximation algorithms for Consecutive Ones Submatrix are known for several restricted classes of matrices, e.g., matrics having only two 1's in each row but any number of 1's in columns, see [6] for many more. Those results are interesting as each of the above problems leads to a slight generalization of the CIG labelling scheme.

Another NP-completeness result comes from a very different area of research: In the Human Genome Project the so called problem of "DNA mapping in the presence of chimericism" was formalized as the k-Consecutive Ones problem [12]. Although the k-Consecutive Ones Property (k-C1P) is defined on matrices and uses row permutation, it is effectively equivalent to our k-CIG problem, cf. Section 4.2. In [1] it is shown that the k-C1P remains NP-complete even if restricted to sparse matrices with at most 7 ones in any row and 8 ones in any column.

From the same article we conclude that any approximation within a constant distance of 1.5 to the optimum solution for k-CIG is NP-complete.

Interval graphs The area of interval graphs is closely connected to the C1P. Indeed the PQ-tree algorithm for solving the C1P problem is designed to solve the interval graph isomorphism problem [4].

Definition 28 (Interval Graph)**.** *A graph $G = (V, E)$ is an interval graph if every node $v \in V$ can be assigned an interval $I(v)$ on the real line such that the intervals of two nodes v, v' have a non-empty overlap if and only if there is an edge between v and v' in G, i.e., $I(v) \cap I(v') \neq \emptyset \iff (v, v') \in E$.*

The connection of interval graphs and CIGs is via the clique-vertex matrix of the interval graph: G is an interval graph if and only if there exists a linear ordering of its cliques such that for each vertex v, the elements of $C(v)$ appear consecutively within the ordering, where $C(v)$ is the set of cliques which contain v [9].

The concept of interval graphs also has been generalized to higher-dimensions. The concept is known as boxicity then and differs from the definiton of an interval graph only in that d-dimensional rectangles overlap in a d-dimensional space. Boxicity then is defined as the minumum number of dimensions necessary to represent a given graph. In [18] it is shown that even testing whether a graph has boxicity 2 is NP-complete. The idea of multi-dimensional intervals is picked up in Section 5 in the extension of the CIG labelling to multi-dimensional intervals. In constrast to the one-dimensional case, there is, however, no known correspondence between testing whether a graph has boxicity d and testing a graph whether a d-dimensional CIG-based labelling can be appliedh.

The boxicity concept is of further interest, because it is shown in [5] to have a strong connection to the treewidth of a graph: For a fixed number of dimensions, the NP-complete problem of boxicity, becomes polynomial if graphs with bounded treewidth are considered. By that a box representation of many graph classes can be computed polynomially, e.g., for chordal graphs, cographs, circular arc graphs, chordal bipartite graphs, permutation graphs, circle graphs, etc. It would be interesting to know if similar results hold for our (d, d', k)-CIG labelling schemes.

Another highly interesting result comes from [19] which finds that interval digraphs are a proper superset of CIG. For that the notion of a zero-partitionable adjacency matrix is introduced. It is notable that a zero-partitionable adjacency matrix gives rise to a graph labelling which also has the constant time per adjacency test and constant space per node properties. However, the best known labelling algorithm for interval digraphs has time complexity of about $O(n^8)$ [21]. Further note that our extensions of the CIG labelling scheme are applicable to more grahps than interval digraphs.

An extensive survey on interval graphs and their connection to the C1P, together with several applications, is presented in [22].

Applications in network routing In network routing the idea of interval-based graph labelling is also considered under the name Interval Routing Scheme (IRS). An IRS differs from a CIG in that labels are assigned to nodes and edges and the fact that an IRS labelling must allow for shortest-path routing. The routing procedure is as follows: Assume a message with destination *dest* arrives at node v. Node v then looks up the number $\sigma(dest)$ of the destination node. Then v searches all its outgoing edges for one that is labelled by an interval *int* with $\sigma(dest) \in int$, i.e., an edge whose assigned interval contains the number of the destination node. The message is then sent along this edge to the next node where this procedure starts again until it finally reaches *dest*. The advantage of such routing is that every node of the network only has to store the ordering of all nodes and the intervals assigned to its own edges instead of the whole routing table. The most interesting results in comparison to our CIG-based labelling schemes are that IRS have seen the extensions to multiple intervals per edge (k-IRS) and further to multiple multi-dimensional intervals per edge ((k, d)-IRS). Both yield NP-complete recognition problems, see [7] and [8]. Although those problem definitions seem to be similar to our k-CIG and (d, d', k)-CIG, in fact no translation from one to the other is known.

4 Multi-Interval CIG Labelling Schemes

The class of k-CIG labelling schemes is the first of our extensions of the CIG labelling scheme, where the restriction of "one interval" covers all children is weakened to "k intervals" cover all children of a node. For every fixed k then there is labelling scheme, i.e., the 2-CIG labelling scheme, the 3-CIG labelling scheme, and so on. By that each of those labelling schemes retains the features of the CIG labelling scheme, i.e., constant space per node and constant time per adjacency test. This, however, comes at the cost of an NP-complete recognition problem for each k-CIG labelling scheme for fixed $k \geq 2$, e.g., testing whether a given graph is a 2-CIG is NP-complete. This also implies that finding the minimal k for a k-CIG labelling of a given graph is NP-complete.

4.1 Introduction to the class of k-CIG Labelling Schemes

Definition 29 (k-CIG)**.** A graph $G = (V, E)$ is a k-CIG if and only if there exists an ordering σ of the nodes V such that for every node $v \in V$ there exists a set of intervals $I_v = \{[b_1, e_1], \ldots, [b_i, e_i]\}$ with $i \leq k$ such that $\mathit{children}(v) = \bigcup_{[b,e] \in I_v} \{v' \in V \mid \sigma(v') \in [b, e]\}$.

Note that this definition ensures that the class of 1-CIG equals the class original CIG (1-CIG = CIG) as both require that all nodes have all their children covered by one interval with their nodes ordered by some σ.

Example 30. Figure 7 shows a 3-CIG along with an ordering $\sigma = (c_{21}, c_{22}, c_{12}, c_1, c, c_2, c_{20}, c_{11}, c_{10}, c_{01}, c_0, c_{00},\ c_{02}, p_{00}, \ldots, p_{22})$ in the lower half. Below σ all intervals of all parent nodes of the graph are drawn. The graph contains of 9 parent nodes which each have a unique child and each have the central node as child. The parents are grouped such that three parents share one additional child. The rows below σ show the intervals needed per parent where one row contains exactly all intervals of one parent. The column of the central node c thus has an interval in every of the 9 rows, as it is child of 9 parent nodes. Note that p_{02} requires three intervals under σ, so the graph is a 3-CIG with σ. Indeed the graph is a proper 3-CIG, i.e., there is no ordering such that every node needs only 2 intervals. This is due to the graph being the result of *construct*(k) with $k = 3$, a function introduced below that generates proper k-CIG graphs.

Properties of k-CIG labelling schemes The following properties hold for all k-CIG labelling schemes:

1. Each k-CIG labelling scheme requires $O(k)$ space per node and allows for an $O(k)$ adjacency test.
2. For every graph there is a k such that the graph can be labelled with a k-CIG labelling scheme.
3. The k-CIG labelling schemes form a proper hierarchy, i.e., for every $k \in \mathbb{N}$ there is a graph G such that $G \notin k$-CIG.

Property 1: Observe that for a fixed k the k-CIG labelling scheme yields a representation where testing adjacency requires constant time per node and constant space for storage: To

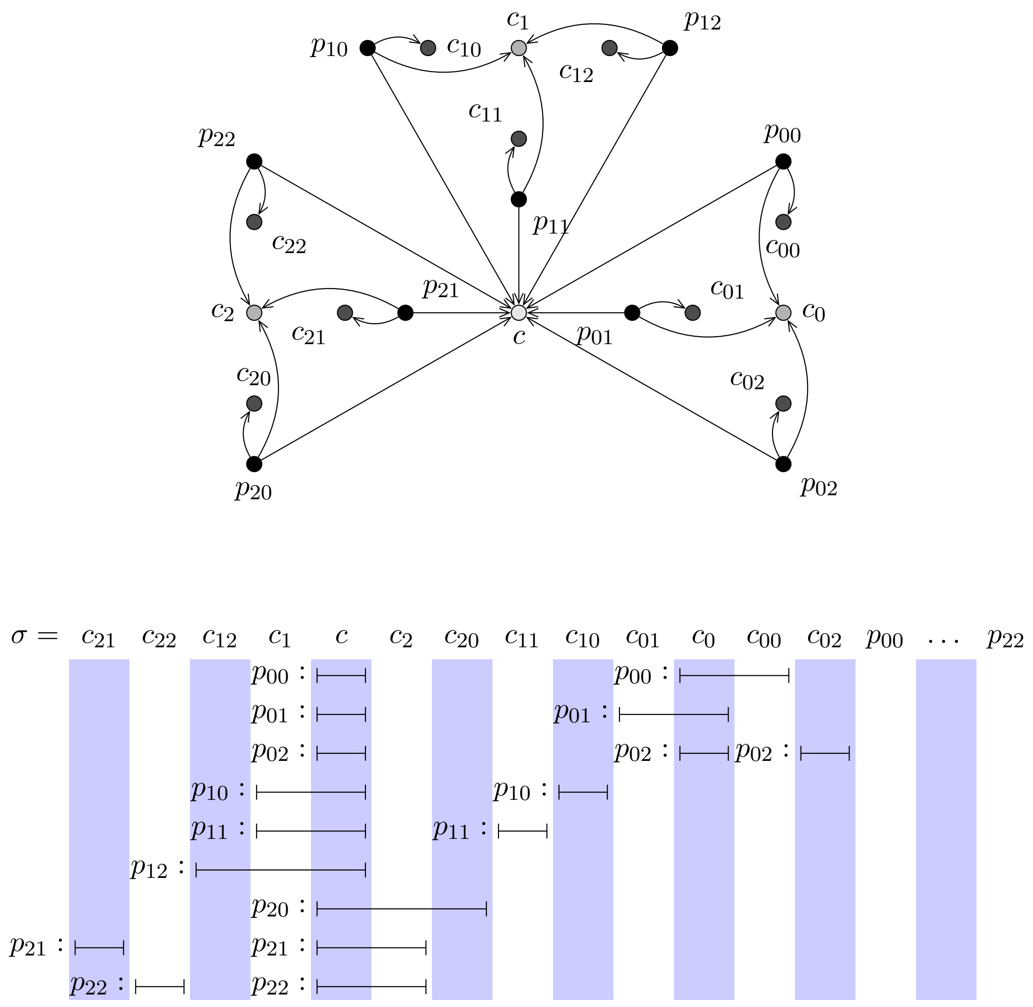


Figure 7: (cf. *Example* 30) A 3-CIG and an ordering σ of its nodes. Every row below σ contains the intervals of one parent node.

test if node v' is a child of v in a k-CIG, it suffices to the lookup the position of v' under the associated ordering σ and to test whether $\sigma(v')$ is inside of one of the at most k intervals of v. For each graph $G \in k$-CIG there are at most k intervals per node v, so this test runs in constant time ($O(k)$) per node for every such G. Additionally the only information needed for this test is σ and up to k intervals per each node. Thus the space requirement for every node is constant, regardless of the rest of the graph.

Property 2: It is notable that the hierarchy of k-CIG labelling schemes indeed is able to cover any finite graph. This means that for every graph G there exists a k such that $G \in k$-CIG. Let $G = (V, E)$ be a graph and k be the maximum out-degree of G, i.e. $k = max\{|children(v)| \mid v \in V\}$. Then $G \in k$-CIG as for any ordering σ each node $v \in V$ has all its children in at most k consecutive intervals. This is due to the fact that every node can use one interval to cover exactly one of its children. As the maximum number of children per node is k, there are enough intervals per node to cover all children of the node. By that there is a constant space per node representation for every graph. However, since here k depends on G and is in $O(|V|)$ there is no gain over simpler representations, such as adjacency lists. The challange remains finding an order of the nodes of G that minimizes the number of such intervals.

Property 3: To show that the k-CIG labelling schemes form a proper hierarchy, we give a construction resulting in a graph G which is a k-CIG, but no $k-1$-CIG. Observe that a node s has exactly one predecessor and one successor in any ordering σ except in the cases where s is the first or last node of the ordering where it has even fewer neighbours, viz. only one. Creating a parent node p_1 of s that has a unique child node c_1 results in either c_1 being a direct neighbour of s under σ or p_1 needing two intervals for its children s and c_1. If three nodes like p_1 are added, i.e., p_1, p_2, and p_3 with unique children c_1, or c_2 or c_3, respectively, then in any ordering one parent needs two intervals. This is due to the fact that only two of the c_i can be neighbours of s and thus only two parents may use only one interval. The third parent, say p_j has its two children s and c_j not consecutive and so needs two intervals. Which parent requires two intervals differs from ordering to ordering, but for any ordering there is at least one such parent.

The recursive application of this leads to the function *construct* which takes a number $k > 1$ as input and returns a graph $G = (V, E)$ in which at least one node $v \in V$ requires k intervals for its children under any possible ordering σ of V.

To generate one node with k intervals the function *construct*(k) produces a graph with 3^{k-1} parent nodes and $\sum_{i=0}^{k-1} 3^i$ child nodes. Each parent has k children and the maximum number of parents per child node is 3^{k-1}. The number of edges in the result is $k \cdot 3^{k-1}$.

Example 31. The result of *construct*(1) to *construct*(3) is given in Figure 8. The top graph shows the result of *construct*(1) which is just one parent node and one child node. In the middle the graph resulting from *construct*(2) is shown. This graph consists in three copies of the output from *construct*(1) and a new central node. Eeach of the three parents from the invocations of *construct*(1) is a parent of the new central node. The bottom graph finally is the result of *construct*(3) where three copies of the ouput graph of *construct*(2) occur. Again a new central node is added as child of every parent from the invocations of

```
$construct(1) = (\{p, s\}, \{(p, s)\})$
        where $p$ and $s$ are new nodes
$construct(k) = \left( \bigcup_{i \leq 3} V_i \cup \{s\}, \bigcup_{i \leq 3} (E_i \cup P_i) \right)$
        where $s$ is a new node and
            $(V_i, E_i) = construct(k-1)$
            $P_i = \{(p, s) \mid p \in parents\,(V_i, E_i)\}$
```

Algorithm 1: *construct*(k)

construct(2).

Theorem 32. *Let $G = (V, E) = construct(k)$ then $G \in k$-CIG and $G \notin (k-1)$-CIG.*

Proof. Observe that each parent node in G has exactly k children, as in each recursive step of *construct*(k) exactly one new child is added to every parent of *construct*$(k-1)$. Thus the maximum out-degree of G is k and every parent in G can use one interval to cover one child, thus $G \in k$-CIG.

To see that $G \notin (k-1)$-CIG, we show that at least one parent node in G requires k intervals. We prove this by induction on *construct* with an induction hypothesis of: *construct*$(n) = (V, E)$ generates a graph that for every ordering σ of V contains at least one parent node that requires at least n intervals to cover all its children. This is obviously true for the base case of *construct*$(1) = (V^1, E^1)$ as $p \in V^1$ requires one interval to cover its child $s \in V^1$. In the inductive case we assume that the property holds for *construct*(n). For the case of *construct*$(n+1)$ there are at least three parents $p_1, p_2, p_3 \in V_1^n \cup V_2^n \cup V_3^n$ which each require n intervals. This is due to $(V_i^n, E_i^n) = construct(n)$ which by the induction hypothesis contains at least one parent node that needs at least n intervals. Each p_i then gets s as a new child. As s can have only two neighbours, at least one of p_1, p_2, p_3 can not have any of its other children consecutive to s. Thus for any ordering at least one of them, say p_j, requires another interval to cover s and so p_j needs $n+1$ intervals to cover all its children. $\square$

Corollary 33. k-CIG $\subsetneq (k+1)$-CIG for any $k \in \mathbb{N}$.

With this corollary we arrive at a proper hierarchy for k-CIG. Additionally this is also establishes that the 2-CIG is a proper superclass of the original CIG, i.e., CIG = 1-CIG $\subsetneq$ 2-CIG. So far the k-CIG labelling schemes meet all of our goals. It only remains to consider the recognition algorithm.

4.2 NP-Completeness of k-CIG

So far the k-CIG satisfies all required properties, lacking only an efficient algorithm to find a k-CIG labelling for a given graph. Indeed, this problem turn out to be NP-complete for

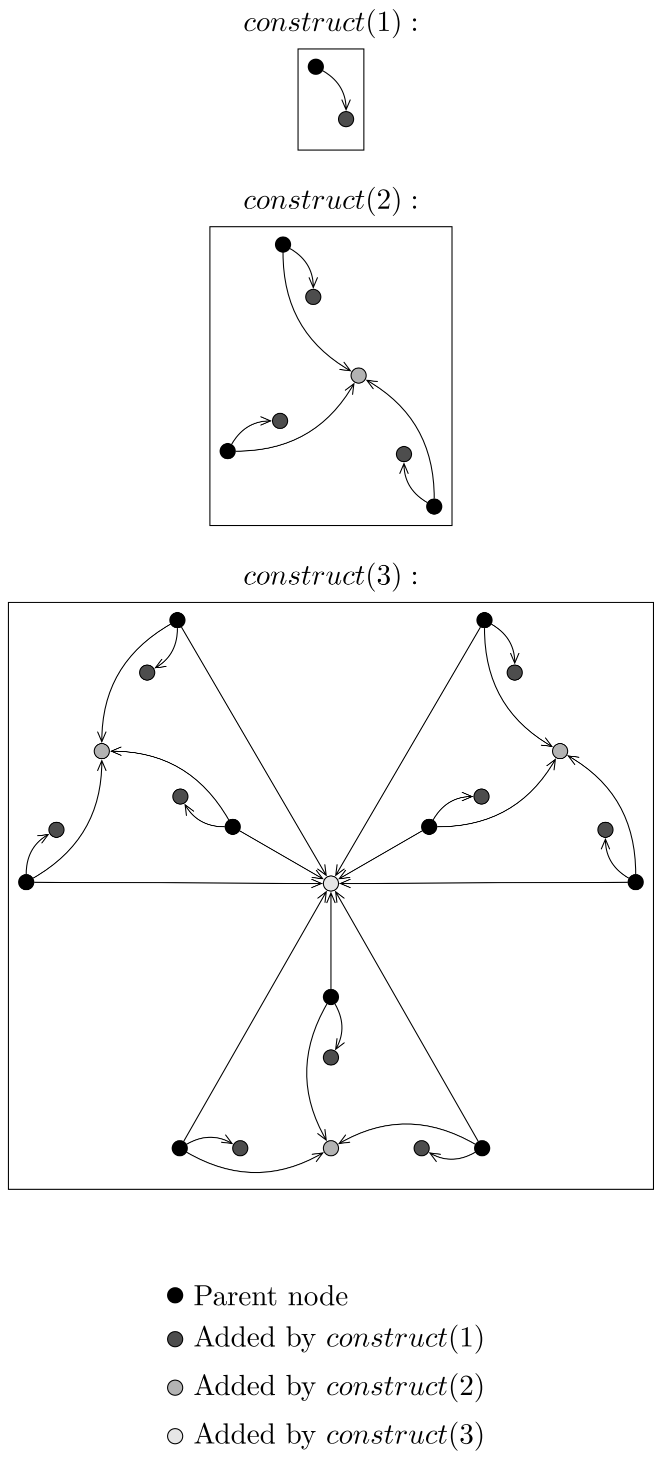

construct(1) :
construct(2) :
construct(3) :
● Parent node
● Added by *construct*(1)
● Added by *construct*(2)
○ Added by *construct*(3)

Figure 8: (cf. *Example* 31) Results of *construct*(1), *construct*(2), and *construct*(3).

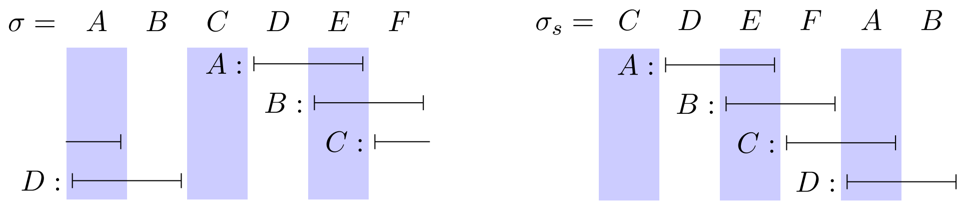


Figure 9: (cf. *Example* 35) Two orderings and intervals representing the same graph. The left ordering σ needs circular intervals while the right ordering σ_s resolves circular intervals by shifting.

every $k \geq 2$ as a very similar problem has already been proven to be NP-complete. In the context of the Human Genome Project the problem of "DNA mapping in the presence of chimerism" is formalized as the k-Consecutive Ones Property (k-C1P) and proven to be NP-complete in [12]. The k-C1P in turn can easily be reduced to the recognition problem of k-CIG. In fact, it is nearly equivalent though defined on adjacency matrices and on row permutations instead of column permutations.

Definition 34 (k-Consecutive Ones Property (k-C1P))**.** The k-Consecutive Ones Property of a $\{0, 1\}$ matrix A is the property that the rows of A can be permuted such that within each column, there are at most k sequences of consecutive ones.

To see that the two problems are equally hard observe that the k-CIG problem can be defined on adjacency matrices, too. By that a graph G is a k-CIG if and only if the columns of its adjacency matrix can be permuted such that in every row there are at most k sequences of consecutive ones. Here the permutation of the columns of the matrix defines the permutation σ from the above k-CIG definition and vice versa. The set of at most k intervals of the k-CIG definition corresponds to the at most k sequences of ones in the adjacency matrix.

Note that there is still a fine difference as k-CIG is based on intervals that can be circular while the sequences of consecutive ones of k-C1P are not circular. The circularity of intervals, however, only makes a difference if multiple intervals (of different nodes) overlap so that they form a cycle through the whole ordering (or through all columns of an adjacency matrix). If they do not form such a cycle then every circular interval can be transformed into a casual one by shifting the whole ordering. Example 35 shows this procedure.

Example 35. In Figure 9 on the left an ordering $\sigma = (A, B, C, D, E, F)$ together with a set of intervals is given. The interval of node C is circular under σ. On the right is the ordering $\sigma_s = (C, D, E, F, A, B)$ which is the same as σ but shifted two positions to the left. Under σ_s no interval is circular. Especially note node C which has no parents and is positioned on the lower end of σ_s. If such a node that has no parents is present, every ordering that needs circular intervals can be shifted so that no interval is circular. This always is the case if the ordering is shifted such that this node without any parents is at one of the ends of the ordering. As the node without any parents cannot be inside any interval, thus no interval has to "wrap around" the ends of the ordering. Further note that shifting an ordering does not change the neighbours of any node, thus does not break up any intervals.

So it holds that every labelling based on circular intervals is equivalent to a labelling based on non-circular intervals, if one node without any parents is present in the labelled graph. Thus if a graph containing one node that has no parents can be labelled using circular intervals, then the same graph can be labelled without circular intervals. Formally $G = (V, E) \in k$-CIG using non-circular intervals exactly if the $G' = (V \cup \{u\}, E)$ with the new node u, which has no parents, is a k-CIG using circular intervals. So the addition of a new node without parents makes the difference of circular intervals go away.

The only difference remaining is that the k-CIG recognition problem permutes columns seeking intervals in rows while k-C1P permutes rows seeking intervals in columns. This, however, is a difference which can be dissolved by the application of matrix transposition, i.e., let A be an adjacency matrix, then A is a k-CIG if and only if the transposed matrix A^t has the k-C1P. Clearly, computing the transposition of A is possible in polynomial time.

In [12] the k-C1P is proven to be NP-complete for every fixed $k \geq 2$ by a reduction from 3SAT. Thus it follows that

Corollary 36. The recognition problem of k-CIG is NP-complete for every fixed $k \geq 2$.

The absence of tractable labelling algorithms for the k-CIG schemes (unless P=NP) implies severe limitations to its use. It may be argued that if the k-CIG scheme is employed in a database the algorithm is run only once at indexing time, but the benefits of the structure are used in every access. On the other hand such an argument never applies to dynamic structures where updates of the database occur often.

5 Multi-Dimensional CIG Labelling Schemes

The previous section establishes that the k-CIG extensions of the CIG labelling exceed the limit of tractability already at its smallest extensions of two intervals per node (unless P=NP). So the aim of this section is to identify other labelling schemes based on the CIG scheme that have the same advantages as CIG, cover a wider class of graphs than CIG, but can be parameterised with a finer granularity than k-CIG. The motivation for the latter is the idea to "zoom" into the "vicinity" between CIG = 1-CIG and 2-CIG in order to explore the borderline between tractable and untractable at a finer scale. For that we investigate labelling schemes where multiple orderings are used as well as schemes which use multi-dimensional intervals. Those extended schemes still retain the constant space per node and constant time per adjacency test properties of the original CIG labelling scheme. Their recognition problem, however, also turns out to be NP-complete.

5.1 The Family of (d, d', k)-CIG Labelling Schemes

In this section the most general class of our extensions to the CIG labelling scheme, namely the class of (d, d', k)-CIG labelling schemes where $d, d', k \in \mathbb{N}$ and $d' \leq d$, is established. This class of labelling schemes contains the extension of the CIG labelling using multiple orderings, namely $(d, 1, 1)$-CIG. It contains the class of labelling schemes using multi-dimensional intervals, namely $(d, d', 1)$-CIG, and it contains the class of schemes allowing more than one interval, namely $(1, 1, k)$-CIG. Note that the $(1, 1, k)$-CIG labelling scheme is basically the same as a k-CIG labelling scheme, thus it follows from the previous section that the $(d, 1, 1)$-CIG and $(d, d', 1)$-CIG labelling schemes are the most interesting, as the $(1, 1, k)$-CIG labelling schemes are already shown in the previous section to have NP-complete recognition problems.

5.1.1 On Multi-dimensional Orderings and Intervals

The above classification also implies that the cases with multiple orderings, i.e., $(d, 1, 1)$-CIG, are special cases of the extension using multi-dimensional intervals, i.e., special cases of $(d, d', 1)$-CIG. Indeed multiple dimensions are the same as multiple orderings in our definitions. The main reason for this is that the original CIG is considered be the one-dimensional instance. Thus one dimension means one ordering which naturally leads to consider multiple dimensions as multiple orderings. Thus the space in which our multi-dimensional intervals are located consists of multiple orderings, i.e., a multi-dimensional ordering:

Definition 37 (Multi-dimensional Ordering)**.** Let $G = (V, E)$ be a graph and $\sigma_1, \ldots, \sigma_d$ be d orderings of the nodes V, then the set $\Sigma = \{1{:}\sigma_1, \ldots, d{:}\sigma_d\}$ is a d-dimensional ordering of V where $1{:}\sigma_1$ states that σ_1 is the ordering of dimension 1. We write $\sigma \in \Sigma$ if and only if there exists $i \in \mathbb{N}$ such that $\Sigma = \{1{:}\sigma_1, \ldots, i{:}\sigma, \ldots, d{:}\sigma_d\}$.

Example 38. Reconsider Figure 4 which used an ordering $\sigma_1 = (F, D, E, B, A, C)$ in dimension 1 and $\sigma_2 = (F, D, E, A, B, C)\}$ in dimension 2. So the 2-dimensional ordering of the graph depicted in Figure 4 is $\Sigma = \{1{:}(F, D, E, B, A, C), 2{:}(F, D, E, A, B, C)\}$.

The definition of interval also has to be adjusted to the multi-dimensional context. We introduce the notion of a boundary segment first which gives us a clear distinction between a constituent part of a multi-dimensional interval, i.e., the boundary segment, and a one-dimensional interval.

Definition 39 (Boundary Segment)**.** A boundary segment in a d-dimensional ordering consists of the borders b and e of an interval which are preceeded by the number of the dimension this interval is located in, i.e., $dim{:}[b, e]$ denotes a boundary segment in dimension number dim which begins at position b and ends at e. Formally a boundary segment is a triple $dim{:}[b, e] \in (\mathbb{N}_d \times \mathbb{N} \times \mathbb{N})$.

Intuitively a multi-dimensional interval then is just a tuple of boundary segments in different dimensions. Note that we allow intervals to have fewer dimensions then there are available. This is the reason for every boundary segment having associated the number of the dimensions it is located in. For example this means that there may be 4 dimensions available, but every node can use only a 2-dimensional interval to cover all its children. This is mainly a restriction on the space usage per node.

Definition 40 (Multi-dimensional Interval)**.** A d'-dimensional interval in a d-dimensional ordering is a $3d'$-tuple $(dim_1{:}[b_1, e_1], \ldots, dim_{d'}{:}[b_{d'}, e_{d'}]) \in (\mathbb{N}_d \times \mathbb{N} \times \mathbb{N})^{d'}$ of d' boundary segments. In the following we write for a d'-dimensional interval int that $dim[b, e] \in int$ if $int = (dim_1{:}[b_1, e_1], \ldots, dim{:}[b, e], \ldots, dim_{d'}{:}[b_{d'}, e_{d'}])$.

In the following we assume that a multidimensional interval is well-defined, i.e., each of its boundary segments uses a unique dimension number, thus $|\{dim | dim{:}[b, e] \in int\}| = d'$.

Example 41. Let X be a node in a graph with a 4-dimensional ordering and (1:$[3, 4]$, 3:$[5, 8]$, 4:$[1, 3]$) the children interval of X. Then its children form a 3-dimensional interval in the first, third, and fourth dimension. In dimension 1 it reaches from 3 to 4, in dimension 3 from 5 to 8, and in dimension 4 from 1 to 3.

As we aim for a most general extension of the CIG labelling scheme which covers all of our extensions, the case where multiple intervals per node are allowed can occur in the context of multiple dimensions, again.

Example 42. Let Y be a node with interval set $I_Y = \{(3{:}[2, 5], 2{:}[1, 4]), (1{:}[2, 3])\}$. Then Y has its children in two intervals where the first interval extends over two dimensions and the second interval extends over one dimension only.

In the following we use d to denote the available number of dimensions and d' to denote the maximum number of dimensions allowed per multi-dimensional interval.

5.1.2 Interpreting Multi-dimensional Intervals

In the case of multi-dimensions with multi-dimensional intervals the question arises how the dimensions interact with each other. Given a parent node p and a d'-dimensional interval for its children, the question is which nodes in the d-dimensional ordering are considered inside the d'-dimensional interval. Note that a multi-dimensional interval is just a collection of boundary segments in some dimensions, thus it does not give an answer to this question

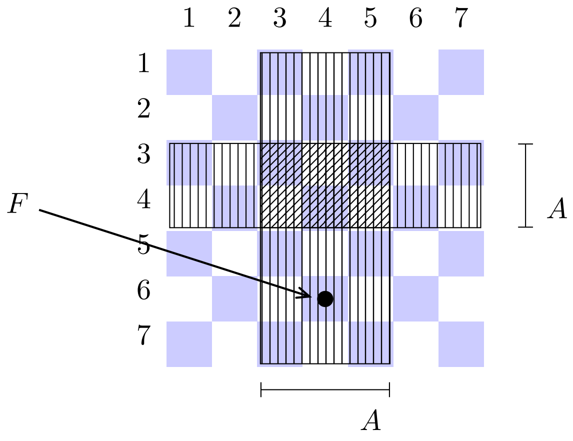


Figure 10: (cf. *Example* 43) Two interpretations of a 2-dimensional interval: Union (weak) and Intersection (strong). Weak results in a cross, strong in a rectangle.

by itself. The first interpretation for reading a collection of boundary segments is that multi-dimensional intervals look like hyper-rectangles, i.e., the children of p are exactly those nodes of the d-dimensional ordering which are inside the boundary segment of every of the d' dimensions. Thus, this interpretation uses the operation of intersection on the d' boundary segments and we call this the strong interpretation. The second interpretation, called weak, uses the operation of union on the d' dimensions. The shape of a weak interval thus is not a hyper-rectangle, but a "hyper-cross".

Example 43. Figure 10 visualizes the difference of weak and strong where a node A has two boundary segments assigned, i.e., a 2-dimensional interval. This interval is $I_A = \{(1{:}[3,5], 2{:}[3,4])\}$. The first (strong) variant using intersection results in hyper-rectangles and the children of A are all nodes inside the rectangle drawn with diagonal lines. The resulting (d, d', k)-CIG labelling is then called strong. The alternative is the use of union which results in the cross drawn with vertical lines. This case is called the weak variant. Assume that there is a node F with position 4 in dimension 1 and position 6 in dimension 2. In the weak case the above multi-dimensional ordering represents a graph $G = (V, E)$ where F is a child of A, i.e., $(A, F) \in E$. In the strong case the same odering represents a graph $G' = (V, E')$ where F is no child of A.

5.1.3 Defining (d, d', k)-CIG Labelling Schemes

The notation (d, d', k)-CIG is read as: d is the number of orderings/dimensions, d' is the maximum dimensionality of an interval, k is the maximum number of intervals per node and weak or strong, respectively, tells how multi-dimensional intervals are interpreted.

Definition 44 ((d, d', k)-CIG)**.** Let $G = (V, E)$ be a graph. $G \in (d, d', k)$-CIG if and only if there exists a d-dimensional ordering $\Sigma = \{1{:}\sigma_1, \ldots, d{:}\sigma_d\}$ of V such that each node $v \in V$ has all its children in at most k intervals each having at most d' dimensions. Formally, for every node $v \in V$ there exists a set I_v which contains at most k intervals each extending over at most d' dimension, i.e.,

$$I_v = \left\{ \left(d_1^1\colon \left[b_1^1, e_1^1\right], \ldots, d_1^{d'_1}\colon \left[b_1^{d'_1}, e_1^{d'_1}\right] \right), \ldots, \left(d_i^1\colon \left[b_i^1, e_i^1\right], \ldots, d_i^{d'_i}\colon \left[b_i^{d'_i}, e_i^{d'_i}\right] \right) \right\}$$

such that:

- each node uses at most k intervals, i.e., $1 \leq i \leq k$.
- each interval uses at most d' dimensions, i.e., $1 \leq d'_1, \ldots, d'_i \leq d'$.
- the edge relation is reconstructible from the intervals, i.e.,

$$(v, v') \in E \iff v' \in \bigcup_{J \in I_v} \begin{cases} \bigcup_{dim:[b,e] \in J} \{v' \in V \mid \sigma_{dim}(v') \in [b, e]\} & \text{for weak} \\ \bigcap_{dim:[b,e] \in J} \{v' \in V \mid \sigma_{dim}(v') \in [b, e]\} & \text{for strong} \end{cases}$$

.

In the following $(d, d', k)_{weak}$-CIG is the class of graphs to which a (d, d', k)-CIG labelling is applicable using the weak interpretation. $(d, d', k)_{strong}$-CIG is the class of graphs to which a (d, d', k)-CIG labelling is applicable using the strong interpretation. (d, d', k)-CIG is the class of graphs where either a $(d, d', k)_{weak}$-CIG or a $(d, d', k)_{strong}$-CIG is applicable, i.e. (d, d', k)-CIG $= (d, d', k)_{weak}$-CIG $\cup$ $(d, d', k)_{strong}$-CIG. Note that there is no difference between weak and strong in those cases where each interval is one-dimensional and thus is not further indicated for $(d, 1, 1)$-CIG.

Note that $(1, 1, 1)$-CIG $=$ CIG, i.e., $G \in (1, 1, 1)$-CIG if and only if $G \in$ CIG. In the case that only one dimension is allowed the definition of (d, d', k)-CIG is equivalent to the previously defined k-CIG, i.e., $(1, 1, k)$-CIG $= k$-CIG. Thus $G \in (1, 1, k)$-CIG if and only if $G \in k$-CIG.

Example 45. Figure 11 shows a proper $(2, 2, 1)_{weak}$-CIG, i.e., a graph which is no CIG and no $(2, 1, 1)$-CIG. On the left the graph is shown. It consists of a central node x, five parent nodes $p_1, \ldots, p_5$ and a unique child node $c_1, \ldots, c_5$ for each parent. On the right side a 2-dimensional ordering, namely $\Sigma = \{1{:}(p_1, p_2, p_3, p_4, p_5, c_1, c_2, c_3, c_4, x, c_5)$, $2{:}(p_1, p_2, p_3, p_4, p_5, c_1, c_2, x, c_3, c_4, c_5)\}$ is given. On the bottom and right sides of the 2-dimensional ordering the intervals of every parent are indicated. Note that a one-dimensional interval is sufficient for every parent except p_1. The 2-dimensional interval of p_1 is drawn as the gray cross inside the space defined by the 2-dimensional ordering.

5.1.4 Properties of (d, d', k)-CIG Labelling Schemes

The most important properties of all (d, d', k)-CIG labelling schemes are:

1. Only constant space per node and constant time for testing adjacency is required for fixed d, d', and k.
2. For every graph there are d, d', and k such that it can be labelled with the (d, d', k)-CIG labelling scheme.
3. The $(d, d', k)_{weak}$-CIG classes form proper hierarchies.

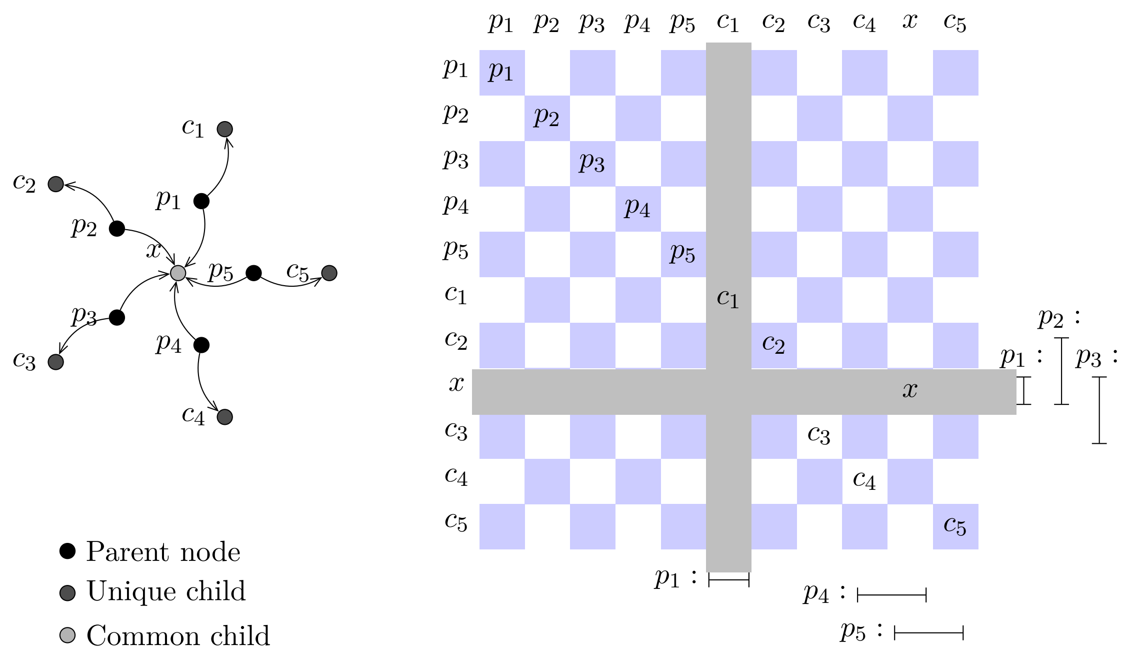


Figure 11: (cf. *Example* 45) A graph and a $(2, 2, 1)_{weak}$-CIG ordering.

Property 1: Observe that in every (d, d', k)-CIG labelling scheme for fixed values of d, d' and k there exists an upper bound of $O(kd')$ for the space requirement of every node. This is due to the restricted number of intervals, namely up to k, where each interval has a restricted number of dimensions, namely up to d'. The same argument applies for the adjacency test as testing wether v' is a child of v just requires testing whether v' is inside one boundary segment of one interval in the weak case. In the strong case the testing is whether v' is inside all boundary segments of one interval of v. Both cases have an upper boundary of $O(kd')$. Thus the adjacency test runs in constant time.

Property 2 It follows directly from Property 2 of the k-CIG labelling schemes and k-CIG corresponding to $(1, 1, k)$-CIG that every graph can be labelled by some (d, d', k)-CIG labelling scheme. This property holds for an arbitrary, but fixed k. Furthermore the same is true if $k = 1$ and d is arbitrary. In the case that $d' = 1$ it holds that for any graph $G = (V, E)$ this graph is a $(|V|, 1, 1)$-CIG: Every parent node has its own ordering where all its children are consecutive. A similar argument as for k-CIG shows that $G \in (d, d', 1)_{weak}$-CIG if d' equals the maximum out-degree of all nodes in G, i.e., every node then can use one boundary segment to cover one of its children.

Property 3 The proof that the (d, d', k)-CIG labelling schemes form proper hierarchies is postponed to Section 5.5 as the proof of each proper hierarchy uses results from sections 5.2 to 5.4 about the NP-completeness of (d, d', k)-CIG. In effect every NP-completeness result for (d, d', k)-CIG yields a reduction function which can be used to generate a graph that is not in (d, d', k)-CIG for some values of d, d', and k. From property 2, however, it follows that

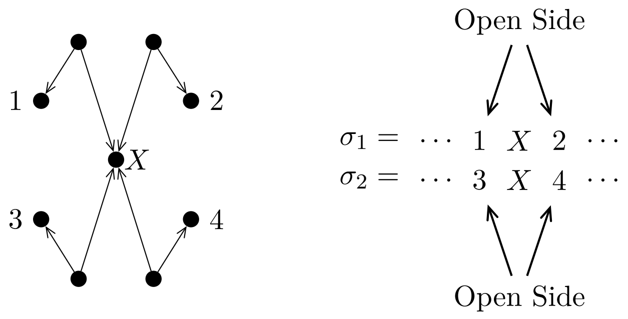

1
2
X
3
4
Open Side
$\sigma_1 = \cdots\ 1\ X\ 2\ \cdots$
$\sigma_2 = \cdots\ 3\ X\ 4\ \cdots$
Open Side

Figure 12: (cf. *Example* 47) Visualization of open sides of node X.

this result of the reduction indeed is member of some other (d, d', k)-CIG labelling scheme, now with greater values for d, d', k. For this class, again, there is a reduction function generating a graph not being a member of that class. Inductively it follows that there is a proper hierarchy among an infinite number of different (d, d', k)-CIG labelling schemes.

Properties of Multi-dimensional Orderings Observe that a d-dimensional ordering over a set of nodes V consists of d one-dimensional, total, and linear orderings of V. This means that two nodes $v, v' \in V$ with $v \neq v'$ have different positions in any of the one-dimensional oderings, i.e., it is not possible that v and v' have the same position in one dimension. For example, it is not possible that v has the "coordinates" $(1, 3)$ and v' has $(2, 3)$ in a 2-dimensional ordering, because v and v' have the same position in the second dimension, here 3.

In later proofs a certain property of CIG labelling schemes, called open sides, is used often. Note that every node $v \in V$ has a well-defined set of neighbours under some multi-dimensional ordering Σ. This set consists of exactly those nodes $v' \in V$ which are neighbours of v in any one-dimensional ordering $\sigma \in \Sigma$. As the number of dimensions is a fixed value, the maximum number of neighbours is fixed also, i.e., two neighbours per dimension. In the following we call the positions of neighbours of v the open sides of v.

Definition 46 (Open Side)**.** Let $G = (V, E)$ be a graph and Σ a mulditimensional ordering of V. In each ordering σ in Σ every node v has exactly two neighbours, viz. a predecessor and a successor. Each such neighbouring position is called an open side of v in Σ. If there are $d = |\Sigma|$ linear orders, then v has $2d$ open sides.

Example 47. The graph in Figure 12 visualizes the open sides of the node X. X has four parents which each have X and another unique node as child. There are two dimensions, i.e., two orderings σ_1 and σ_2, thus X has four open sides. Those open sides are consumed by the nodes 1 to 4 as pictured on the right of the figure. Observe that a fifth parent could not have one of its children consecutive to X as there are no open sides left. This however only applies in the weak case, for $(d, d', k)_{strong}$-CIG labelling schemes this does not hold immediately.

5.1.5 Recognition Problems of (d,d',k)-CIG Labelling Schemes are in NP

The recognition problem of each (d,d',k)-CIG labelling scheme indeed is in NP, i.e., can be solved by a non-deterministic Turing machine in polynomial time. We give a guess-and-check algorithm as proof: Let $G=(V,E)$ be the graph to decide whether it can be labelled with a (d,d',k)-CIG scheme for fixed values of d,d', and k. The recongition algorithm first assigns non-deterministically to each node $v \in V$ a position in a d-dimensional ordering. Then the algorithm assigns non-deterministically to every node up to k intervals with each interval consisting of up to d' boundary segments where each boundary segment is non-deterministically assigned a beginning and an end. Thus the algorithm non-deterministically creates a (d,d',k)-CIG labelling for a graph $G'=(V,E' \subset (V \times V))$ possibly differing from G by edges. This labelling is of size $O(|V| \cdot d' \cdot k + |V| \cdot d)$ and it is created in polynomial time.

The second part of the recognition algorithm then only has to check whether the created labelling represents G. This can be checked by enumerating the set of all possible edges and testing for every possible edge $(v,v') \in (V \times V)$ whether the edge is in the original graph, i.e., $(v,v') \in E$, whenever it also occurs in the created labelling, i.e., $(v,v') \in E'$. This runs the adjacency test for every pair of nodes on the created labelling and checks that an edge is in the graph if and only if it is also in the graph defined by the labelling. As testing adjacency is possible in constant time w.r.t. d,d',k, testing whether the created labelling correctly represents the original graph G is possible in time $O(|V|^2 \cdot d \cdot d' \cdot k)$. If the labelling is correct, then $G \in (d,d',k)$-CIG and if it is not correct, then $G \notin (d,d',k)$-CIG, i.e., the algorithm solves the recognition problem of (d,d',k)-CIG.

Note that the definition of a non-deterministic Turing machine states that correct values are used if there exist such values, i.e., a correct labelling is found whenever there exists one. As the runtime of the above algorithm is polynomial and non-deterministic, we conclude that (d,d',k)-CIG is in NP.

5.2 NP-Completeness of $(d,1,1)$-CIG

In this section we prove the recognition problem of $(d,1,1)$-CIG to be NP-complete for $d \geq 2$ by reducing the NP-complete problem of graph d-Colorability to $(d,1,1)$-CIG. As d-Colorability is NP-complete only for $d \geq 3$, another proof is required for the case of $(2,1,1)$-CIG. A similar problem from literature is known as the Consecutive Ones Matrix Partition (C1MP) problem, [11]. In the next section we show that C1MP and $(2,1,1)$-CIG indeed are equivalent. Only in Section 5.2.2 the d-colorability problem is reduced to $(d,1,1)$-CIG.

5.2.1 Equivalence of C1MP and $(2,1,1)$-CIG

The formal definition of the C1MP is introduced first and followed by the proof that it is equivalent to $(2,1,1)$-CIG.

Definition 48 (Consecutive Ones Matrix Partition (C1MP) [11])**.** Given an $m \times n$ matrix A of 0's and 1's. Can the rows of A be partitioned into two groups such that the resulting

$$\left(\begin{array}{ccccccc} & A & B & C & D & E & F \\ A & & & & & & \\ & & & & & & \\ B & & & & & \iddots & \\ C & 0 & 0 & 0 & 0 & & \\ D & 1 & 1 & 0 & 0 & & \\ E & 0 & 1 & 1 & 0 & & \\ F & 1 & 0 & 1 & 0 & & \end{array}\right) \Rightarrow \begin{array}{c} \left(\begin{array}{ccccccc} & B & C & A & D & E & F \\ A & & & & & \iddots & \\ B & 0 & 0 & 0 & 0 & & \\ E & 1 & 1 & 0 & 0 & & \\ F & 0 & 1 & 1 & 0 & & \end{array}\right) \\ \\ \left(\begin{array}{ccccccc} & A & B & C & D & F & E \\ C & 0 & 0 & 0 & 0 & 0 & 0 \\ D & 1 & 1 & 0 & 0 & 0 & 0 \end{array}\right) \end{array}$$

Figure 13: (cf. *Example* 49) Partitioning of an adjacency matrix according to C1MP.

$m_1 \times n$ and $m_2 \times n$ matrices $(m_1 + m_2 = m)$ each separately have the Consecutive Ones Property?

Example 49. Figure 13 shows how an adjacency matrix is partitioned into two matrices according to the C1MP. The adjacency matrix on the left represents the $(2, 1, 1)$-CIG of the introductory example, i.e., Figure 4. Recall that this graph can not be labelled according to the CIG labelling scheme. The matrix, however, can be partitioned into two matrices each having the Consecutive Ones Property according to the C1MP. Each matrix then has its own column ordering. Note that both matrices on the right can be extended by additional rows containing 0's such that they each include a row for every node of the graph. The extended matrices then each represent a dimension of the $(2, 1, 1)$-CIG with the ordering of the columns of a partition being the ordering of the dimension. In this case the 2-dimensional ordering induced by the two partitions is $\Sigma = \{1{:}(B, C, A, D, E, F), 2{:}(A, B, C, D, F, E)\}$.

Theorem 50. *Let A be an $n \times n$ matrix over $\{0, 1\}$ and $A = adj(G)$. A can be partitioned according to C1MP if and only if G is a $(2, 1, 1)$-CIG.*

Proof. Assume A can be partitioned into A_1 and A_2 with both parts having the Consecutive Ones Property. This means there is a permutation σ_1 of the columns of A_1 and a permutation σ_2 of the columns of A_2, such that all 1's are consecutive in $\sigma_1(A_1)$ and in $\sigma_2(A_2)$. As A has n columns, $A = adj(G)$, and $G = (V, E)$, we have that V contains n elements. But A_1 and A_2 each have n columns by definition, thus σ_1 and σ_2 each define an ordering of the nodes of V. We use σ_1 and σ_2 to construct the 2-dimensional ordering $\Sigma = \{\sigma_1, \sigma_2\}$. It remains to show that G is a $(2, 1, 1)$-CIG under Σ, i.e., for each node $v \in V$ there exists a one-dimensional interval such that all its children are consecutive in this interval. By definition of C1MP the row containing the adjacent children of v is part of A_i where i is either 1 or 2 and all 1's in this row of A_i are consecutive. Thus v has all its children consecutive in σ_i, i.e., i is the dimension where v has all its children consecutive. As this applies to every node $v \in V$, G is a $(2, 1, 1)$-CIG.

Assume G is a $(2, 1, 1)$-CIG, i.e., there exists a 2-dimensional ordering $\Sigma = \{\sigma_1, \sigma_2\}$ such that each node $v \in V$ has all its children consecutive in σ_i for i either 1 or 2. Let $A'_1 = \sigma_1(A)$

and $A_2' = \sigma_2(A)$. Now every node $v \in V$ is represented in a row in A_1' as well as one in A_2'. By the definition of $(2, 1, 1)$-CIG either the row in A_1 or in A_2 consists of a consecutive interval of ones. We then define A_1 to be the matrix A_1' where all rows which contain no consecutive block of ones are deleted. A_2 is defined in the same way by deletion of rows from A_2'. If a node v has its children consecutive in σ_1 and σ_2 the corresponding row in A_2 is deleted to prevent the row of v occuring in both matrices. Thus A_1 and A_2 are a partitioning of A and both have the Consecutive Ones Property. $\square$

As the C1MP is NP-complete, see [11] problem SR15, it follows that:

Corollary 51. The recognition problem of $(2, 1, 1)$-CIG is NP-complete.

5.2.2 Reduction of d-Colorability to $(d, 1, 1)$-CIG

In this section we show the recognition problem of $(d, 1, 1)$-CIG to be NP-complete by a reduction from d-Colorability. The underlying intuition is that each dimension of a $(d, 1, 1)$-CIG serves as a color which can be assigned to nodes.

Definition 52 (d-Colorability)**.** Given an undirected graph $G = (V, E)$ and a positive integer $d \leq |V|$. Is G d-colorable, i.e., does there exist a function $col : V \to \{1, 2, \ldots, d\}$ such that $col(u) \neq col(v)$ whenever $\{u, v\} \in E$?

Proposition 53 ([11] problem GT4)**.** The decision problem whether an undirected graph G is colorable with d colors is NP-complete for $d \geq 3$.

In the following we use the alternative characterization of d-Colorability:

1. Each node has exactly one color.
2. Every two adjacent nodes have different colors.

Principles of the Reduction The goal of this section is a polynomially computable function, called $reduction_d$, so that for every graph G_c it holds that G_c is d-colorable if and only if $reduction_d(G_c) = G_r$ is a $(d, 1, 1)$-CIG. In the following we address the (undirected) graph that is to color by $G_c = (V_c, E_c)$ while the output of the reduction is addressed by the (directed) graph $G_r = (V_r, E_r)$. As the number of colors on the one side equals the number of dimensions on the other side, it is obvious that dimensions serve as colors.

A node $v \in V_c$ is assumed to have color number d_v if a certain set of nodes of V_r appears as a special sequence in dimension number d_v. This special sequence of nodes is called the color section of v. Function $color_d$ gives the construction of a color section for v such that the color section of v appears for any possible ordering of G_r in only one dimension. This ensures Property 1 of the characterization of d-Colorability.

Assume a node $e_{A,B} \in G_r$ being forced to appear in any possible ordering of G_r between A and c^A as well as between B and c^B, i.e., for any $(d, 1, 1)$-CIG ordering Σ of G_r there are dimensions σ_i, σ_j in Σ such that $\sigma_i = (\ldots, A, e_{A,B}, c^A, \ldots)$ and $\sigma_j = (\ldots, B, e_{A,B}, c^B, \ldots)$. The conclusion from the above is that $i \neq j$ as in every ordering $e_{A,B}$ has exactly one position. If $A, e_{A,B}, c^A$ is considered to be the color section of A while $B, e_{A,B}, c^B$ is considered to be the color section of B, then the node $e_{A,B}$ effectively ensures that the color sections of A and B are in different dimensions, i.e., A and B have different colors. The reduction function is based on this observation and adds for every $(v, v') \in E_c$ a node $e_{v,v'} \in E_r$ in the color sections of v and v'. Thus two adjacent nodes in G_c are ensured to have their color sections in different dimensions, thus they are considered to have different colors. This ensures Property 2 of the characterization of d-Colorability.

Encoding of Colors The first step towards the reduction is function color_d which constructs the color section of a node $v_i \in V_c$ w.r.t. a set S of nodes that are already known to be part of the color section of v_i. Let $S = \{s_1, \ldots, s_n\}$, $v \in V_c$, and $d \in \mathbb{N}$ then color_d is defined such that for $G' = \mathit{color}_d(v, S)$ in every $(d, 1, 1)$-CIG ordering Σ of G' there is one dimension $\sigma \in \Sigma$ where v_i, c^{v_i}, and all nodes $s \in S$ appear in a consecutive sequence such that v_i and c^{v_i} terminate the sequence.

Example 54. Let $A \in V_c$, $d = 3$, $S = \{e_B, e_C, e_D\}$, and $G' = \mathit{color}_3(A, S)$. A $(d, 1, 1)$-CIG ordering of G' then may contain $\sigma \in \Sigma$ with $\sigma = (\ldots, A, e_D, e_C, e_B, c^A, \ldots)$. The sequence A, e_D, e_C, e_B, c^A is the color section of A. Another possible ordering would be $\sigma = (\ldots, c^A, e_B, e_D, e_C, A, \ldots)$, i.e., the ordering of the nodes e_B, e_C, e_D is irrelevant but they are enclosed between A and c^A.

For the definition of color_d observe that a node p which has two children, b_1 and b_2, requires those to be consecutive in one of the available dimensions of every $(d, 1, 1)$-CIG ordering. If b_1 and b_2 were not consecutive in any dimension, then p would require two intervals to cover its children, thus the graph would be no $(d, 1, 1)$-CIG. Furthermore every node v in a graph that is labelled according to a $(d, 1, 1)$-CIG labelling scheme has at most $2d$ open sides, i.e., two neighbours in each of the d dimensions. This means that there can be at most $2d$ parent nodes $p_1, \ldots, p_{2d}$ which each have a pair of children, namely v and b_i where all $b_1, \ldots, b_{2d}$ are unique nodes. Thus by the addition of $2d - 1$ parent nodes which have v and each a new node b_i as children, all but one open side of v are used up. We then use this last open side of v to signal the coloring of v, i.e., the color section of v_i is added at this last open side.

The input of color_d is a single node $v_i \in V_c$ and a sequence S of nodes which are included in the color section of v_i. The output of color_d is a directed graph (V_i, E_i) whose $(d, 1, 1)$-CIG ordering contains one color section in exactly one of the d dimensions.

$$
\begin{aligned}
\mathit{color}_d(v, S) = &(\{v, c^v, b_1^v, \ldots, b_{2d-1}^v, p_1^v, \ldots, p_{2d-1}^v, p_{Sc}^v, p_S^v\}, E)\\
&\quad \textbf{where } \text{all } p_i^v, b_i^v, p_{Sc}^v, \text{and, } p_S^v \text{ are new nodes}\\
&\qquad E = \textstyle\bigcup_{i=1}^{2d-1} \{(p_i^v, v), (p_i^v, b_i^v)\}\\
&\qquad\quad \cup \{(p_{Sc}^v, v), (p_{Sc}^v, c^v)\} \cup \{(p_{Sc}^v, s) \mid s \in S\}\\
&\qquad\quad \cup \{(p_S^v, v)\}) \cup \{(p_S^v, s) \mid s \in S\}
\end{aligned}
$$

Algorithm 2: $\mathit{color}_d(v, S)$

Example 55. Figure 14 shows the output of $\mathit{color}_3(A, \{s_1, \ldots, s_n\})$. It consists of a graph

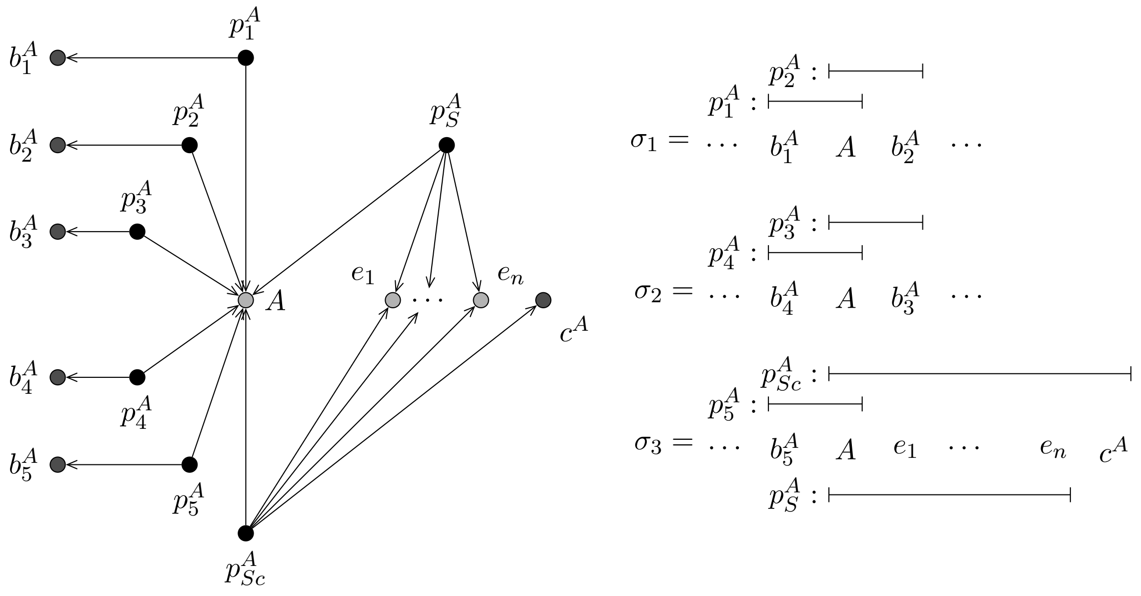


Figure 14: (cf. *Example* 55) Result of $color_3(A, \{s_1, \dots, s_n\})$ and a possible 3-dimensional ordering. The color section $A, e_1, \dots, e_n, c^A$ of A is in dimension 3.

which enforces that exactly one dimension, here 3, contains a color section. The intervals where a parent p_i has its children consecutive in an order are indicated. As there are five parents $p_1, \dots, p_5$ which each have A and one unique child, there is exactly one open side left to put the color section of A. The color section is formed by p_S and p_{Sc} where p_S ensures that all $s_1, \dots, s_n$ have to be enclosed by A and c^A in order to satisfy the consecutiveness conditions of p_S^A and p_{Sc}^A. The output of $color_3(A, \{s_1, \dots, s_n\})$ is shown on the left. On the right side a possible 3-dimensional ordering with all intervals is drawn. The ordering here is $\Sigma = \{1{:}(\dots, b_1^A, A, b_2^A, \dots) = \sigma_1, 2{:}(\dots, b_4^A, A, b_3^A, \dots) = \sigma_3, 3{:}(\dots, b_5^A, A, e_1, \dots, e_n, c^A, \dots) = \sigma_3\}$. Thus the color section of A is in dimension 3.

Note that c^{v_i} and all $s \in S$ indeed occur in every dimension, but they are consecutive as $v_i, s_1, \dots, s_n, c^{v_i}$ exactly in one dimension.

Theorem 56. *Let $V_c = \{v_1, \dots, v_n\}$ and $S_1, \dots, S_n$ be sets of nodes, $(V_i, E_i) = color_d(v, S_i)$ the color construction applied to every node in V_c w.r.t. S_i, and $G' = (\bigcup_{i=1}^n V_i, \bigcup_{i=1}^n E_n)$ the union of all those color constructions. For every $(d, 1, 1)$-CIG labelling of G' the following applies to the d-dimensional ordering Σ .*

1. *Every node $v_i \in V_c$ has exactly one color section in Σ.*
2. *If the color section of v_i is in dimension d_i in Σ then all nodes $s \in S_i$ are between the positions of v_i and c^{v_i} in dimension d_i.*
3. *No color sections of two nodes $v_i, v_j \in V_c$ overlap in any ordering of Σ.*
4. *If a node $s \in S_i \cap S_j$ is part of the color sections of v_i and v_j then the color sections of v_i and v_j appear in different dimensions.*

Property 1. All but one open side of v_i in Σ is used by $b_1^{v_i}, \ldots, b_{2d-1}^{v_i}$ which must be neighbours to v_i due to $p_1^{v_i}, \ldots, p_{2d-1}^{v_i}$. As a color section contains v_i the only possibility for the color section to appear is at the remaining open side of v_i. Thus there can be only one dimension in Σ where the full color section of v_i appears, i.e., there is only one dimension where $p_S^{v_i}$ and $p_{Sc}^{v_i}$ have their children consecutive. □

Property 2. Let d_i be the dimension where the last open side that is not consumed by $b_1^{v_i}, \ldots, b_{2d-1}^{v_i}$ of v_i appears. As there is only one open side of v_i left where the color section can appear, it follows that v_i itself must be at one end of the color section. If v_i is not at the end of the color section, then the color section would consume two open sides, which can not be the case. This last open side is the only possibility for $p_S^{v_i}$ to have its children $v_i, s_1, \ldots, s_n$ consecutive where $s_1, \ldots, s_n \in S_i$. Thus $s_1, \ldots, s_n$ indeed must follow immediately to v_i. The same is true for $p_{Sc}^{v_i}$ which has one additional child, namely c^{v_i}. As $p_{Sc}^{v_i}$ needs $v_i, s_1, \ldots, s_n, c^{v_i}$ consecutive, the last child c^{v_i} thus must be placed on the remaining end of the color section. By that the color section ends on the one side with v_i and on the other with c^{v_i}. As all $s \in S_i$ are part of the color section, they thus must be between v_i and c^{v_i} in dimension d_i. □

Property 3. Assume that neither of $v_i, c^{v_i}, v_j, c^{v_j} \in S_i \cup S_j$, i.e. the ends terminating color sections are distinct from the contents of each color section. By that no color section is fully contained in another color section while every color section is terminated by unique nodes. If the color sections of v_i and v_j would overlap then one of the terminating nodes of v_i would be inside the color section v_j. As neither v_i nor c^{v_i} are part of the color section of v_j this would break up the color section of v_j and make Σ no ordering for a $(d, 1, 1)$-CIG labelling of G'. Thus the color sections can not overlap. □

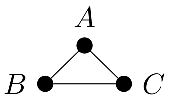
Property 4. Assume the two color sections of v_i and v_j to be in the same dimension *dim* in Σ. By Property 2 node s is inside both color sections. By Property 3 the color sections of v_i and v_j do not overlap. Thus by assumption s appears twice in dimension *dim*. As every dimensions is an ordering of nodes, s cannot appear twice in any dimensions. Thus the assumption must be wrong and the color sections of v_i and v_j do not appear in the same dimension. □

The Reduction Function The final step of the reduction is a function instantiating $color_d$ such that every edge $\{v, v'\} \in E_c$ is transformed into a node named $e_{v,v'} \in V_r$. Then $color_d$ is applied on every node $v \in V_c$ where the set of nodes being inside the color section of v is exactly the set of transformed edges which are incident to v in G_c. For any edge $\{v, v'\} \in E_c$, $color_d(v, S_v)$, and $color_d(v' S_{v'})$ it holds that $e_{v,v'} \in S_v$ and $e_{v,v'} \in S_{v'}$. The reduction function is called $reduction_d$, its input is an undirected graph $G_c = (V_c, E_c)$ with $V_c = \{v_1, \ldots, v_n\}$, and it returns a directed graph $G_r = (V_r, E_r)$.

Example 57. Figure 15 shows a 3-colorable graph, say G, with three nodes A, B, and C. Below this graph, a possible 3-dimensional ordering is shown. This $(3, 1, 1)$-CIG ordering belongs to the graph $G' = reduction_3(G)$ which is the result of the reduction function $reduction_d$ applied to G. The coloring of G can be reconstructed from the ordering as the dimension where the color section of a node is placed represents the color of the node.

$$\mathit{reduction}_d(V, E) = \Big(\bigcup_{i=1}^{n} V_i, \bigcup_{i=1}^{n} E_i\Big)$$
$$\textbf{where } (V_i, E_i) = \mathit{color}_d(v_i, \{e_{v_i,v'} \mid \{v_i, v'\} \in E\}) \text{ for each } i \le n$$

Algorithm 3: *reduction*$_d$



$$\begin{array}{lllll}
\sigma_1 = & \dots\ b^A\ \overline{A\ e_{A,B}\ e_{A,C}\ c^A} & \dots\ b^B\ B\ b^B & \dots\ b^C\ C\ b^C & \dots \\
\sigma_2 = & \dots\ b^A\ A\ b^A & \dots\ b^B\ \overline{B\ e_{A,B}\ e_{B,C}\ c^B} & \dots\ b^C\ C\ b^C & \dots \\
\sigma_3 = & \dots\ b^A\ A\ b^A & \dots\ b^B\ B\ b^B & \dots\ b^C\ \overline{C\ e_{A,C}\ e_{B,C}\ c^C} & \dots
\end{array}$$

Figure 15: (cf. *Example* 57) A 3-colorable graph and an ordering of its nodes after reduction to $(3, 1, 1)$-CIG. Color sections are overlined.

Every edge $\{A, B\} \in E$ from G is represented by a node $e_{A,B}$ which must be inside the color sections of its two end points A and B. As every node has exactly one position in any dimension, the two color sections of A and B thus must be in different dimension as an overlap of color sections is not possible, because the nodes terminating the color section of B, namely B and c^B do not occur in the color section of A. The pictured 3-dimensional ordering assigns color 1 to A, color 2 to B, and color 3 to C.

Theorem 58. *$G_c = (V_c, E_c)$ is d-colorable if and only if $G_r = (V_r, E_r)$ is a $(d, 1, 1)$-CIG with $G_r = \mathit{reduction}_d(G_c)$.*

Proof. Assume G_c is d-colorable, i.e., every two adjacent nodes have different colors. We show that G_r is a $(d, 1, 1)$-CIG by sketching an d-dimensional ordering Σ for G_r: Let $\mathit{col}(v)$ be the color of node v according to some coloring of G_c with d colors. We organize Σ such that the color section of every node $v \in V_c$ is in dimension $\mathit{col}(v)$, i.e. the color sections represent the same coloring as the one of G_c. By definition of $\mathit{reduction}_d$ and d-Colorability every pair of adjacent nodes in V_c then has its color sections appearing in different dimensions in Σ. Thus for every pair of nodes $v_i, v_j \in V_c$ with $s \in S_i$ and $s \in S_j$, i.e., $\{v_i, v_j\} \in E_c$, it follows that the color sections created by $\mathit{color}_d(v_i, S_i)$ and $\mathit{color}_d(v_j, S_j)$ are in different dimensions. With the color sections placed all that remains are the children of nodes $p_1^v, \dots, p_{2d-1}^v$ for $v \in V_c$. Their placement, however, is already induced by the color section, i.e., simply place all $b_1^v, \dots, b_{2d-1}^v$ as direct neighbours to v.

Assume G_c is not d-colorable, i.e., there exists no coloring of G_c using only d colors. By Property 4 of $color_d$ the color sections of every pair of adjacent nodes in E_c have to appear on different dimensions in Σ. By Property 1 every node has exactly one color section while every node $v \in V_c$ is ensured to have a color section by definition of $reduction_d$. For a contradiction assume that G_r can be labelled according to the $(d, 1, 1)$-CIG labelling scheme. Then there exists an ordering Σ where each node $v \in V_c$ has a color section in some dimensions and for every adjacent pairs of nodes in $\{v, v'\} \in E_c$ the two have their color sections in different dimensions. Thus a coloring of G_c using only d colors is induced by Σ. This is a contradiction to the assumption that no such coloring exists for G_c.

□

Theorem 59. *The recognition problem of* $(d, 1, 1)$*-CIG is NP-complete for every fixed* $d \geq 2$*.*

Proof. Theorem 50 gives that the $(2, 1, 1)$-CIG recognition problem is NP-complete. Theorem 58 defines a reduction from d-Colorability to $(d, 1, 1)$-CIG. It remains to be shown that this reduction is polynomial. In fact the function $color_d$ is linear in the size of its input d and a color section. Function $reduction_d$ calls $color_d$ once for each node v with a color section of the size of the edges incident to v. It then only combines the results of these calls. Thus the resulting graph is linear in the size of the original graph. □

5.3 NP-Completeness of $(d, d', k)_{weak}$-CIG

In this section the recognition problem of each $(d, d', k)_{weak}$-CIG labelling scheme is proven to be NP-complete for $d \geq 2$. The proof itself is a polynomial reduction from $(d, 1, 1)$-CIG to $(d, d', 1)_{weak}$-CIG which then is expanded to a reduction to $(d, d', k)_{weak}$-CIG. The first step for this reduction is the function seq_d returning a graph, which forces a set of nodes S to be consecutive in one dimension in any multi-dimensional ordering (and thus in any $(d, d', k)_{weak}$-CIG labelling). This means that any $(d, d', 1)_{weak}$-CIG labelling of such a graph contains an ordering among its multi-dimensional ordering where all nodes of S are positioned continuously. This, however, is the same effect as if the set of nodes were children of one parent in a $(d, 1, 1)$-CIG representation. Therefore seq_d yields a construction that allows to represent the child set of a node of a $(d, 1, 1)$-CIG in a $(d, d', k)_{weak}$-CIG, but places no further restrictions on the multi-dimensional ordering. Thus the full reduction just has to apply seq_d to the set of children of every parent and the resulting graph is a $(d, d', 1)_{weak}$-CIG exactly if the original graph is a $(d, 1, 1)$-CIG. This stems from the Definition 23 and the observation that the sets of children is all that matters to any CIG-based ordering.

Recall that a multi-dimensional interval is a collection of multiple boundary segments where a boundary segment is of the form $dim{:}[b, e]$, i.e., basically a one-dimensional interval. The function seq_d is based on the observation that in the weak case (in any $(d, d', k)_{weak}$-CIG) two children of a parent must be neighbours in one of the dimensions to only require one boundary segment of the multi-dimensional interval of the parent. If both children are not neighbours in any dimension, then the parent node needs two boundary segments to cover its children. This fact follows directly from the definition of $(d, d', k)_{weak}$-CIG which defines a multi-dimensional interval to be the union of one-dimensional intervals, i.e., a union of boundary segments. Note that in the strong version, this property does not hold as some nodes in the boundary segment for dimension i might be dropped if they do not occur in the boundary segment of some other dimension j. Therefore the children of the parent need not be neighbours in dimension i.

Nodes with maximal use of boundary segments From the definition of open sides 46 it follows that for a graph $G = (V, E)$ each node $v \in V$ has at most $2d$ open sides in a $(d, d', 1)_{weak}$-CIG with d dimensions/orderings. This gives an upper bound to the maximum neighbours of a node. With these observations we can construct a graph containing a node that requires 2 boundary segments in any possible ordering. Such a graph contains a central node s which is the child of $2d+1$ parent nodes $p_1, \ldots, p_{2d+1}$ with each parent p_i having an unique child c_i. Each parent p_i needs only one boundary segment if its child c_i is a neighbour of s. The central node s, however, can only have $2d$ neighbours while $2d+1$ parent nodes need to have their second child to be a neighbour of s. Thus for any possible multi-dimensional ordering there is at least one parent node which can not have its two children consecutive in any of the available dimensions.

$construct_d(n)$ applies this procedure of adding more neighbours than the available number of open sides recursively and creates a graph such that for each n-dimensional ordering there is at least one parent node that requires n boundary segments. It can be seen as the multi-dimensional version of *construct* from section 4.1. Let $construct_d$ be a function having a natural number as input and returning a graph and let $parents(V, E) = \{v \in V \mid \exists v' \in V : (v, v') \in E\}$ be the set of all parent nodes in a graph (V, E).

$$construct_d(1) = (\{p, s\}, \{(p, s)\})$$

$$\textbf{where } p \text{ and } s \text{ are new nodes}$$

$$construct_d(n) = \left(\bigcup_{i \leq 2d+1} V_i \cup \{s\}, \bigcup_{i \leq 2d+1} (E_i \cup P_i) \right)$$

$$\begin{aligned} &\textbf{where } s \text{ is a new node and} \\ &\qquad (V_i, E_i) = construct(n-1) \\ &\qquad P_i = \{(p, s) \mid p \in parents\,(V_i, E_i)\} \end{aligned}$$

Algorithm 4: $construct_d$

Example 60. Figure 16 visualizes the behaviour of $construct_d$. Note the similarities to *construct* from section 4.1 which is depicted in Figure 8. In principle both functions generate more than the maximum number of open sides, which are 2 for *construct* and $2d$ for $construct_d$. Again different shades are used to indicate at what recursive depth a node was generated. In the first (top) step only one node with one child is constructed. The second (middle) step now generates 5 parent nodes each having a unique child. As the central node of this step has only 4 open sides, one of the parents can not have its child consecutive to the central node. Thus this parent needs two boundary segments. The third (bottom) step uses five copies of the result from the second step, thus there are 5 parent nodes which already need two boundary segments. As a new central node is added, one of those 5 parents can not have any of its other children consecutive to the central node. Thus this parent needs an additional boundary segment, i.e., the third one.

Note that in every step of $construct_d$ there are $2d+1$ recursive calls which each produce a new graph with new nodes. Also $construct_d(n)$ is exponential in n and polynomial in d. To generate one node with n boundary segments the function $construct_d(n)$ produces a graph with $(2d+1)^{n-1}$ parent nodes and $\sum_{i=0}^{n-1}(2d+1)^i$ child nodes. Each parent has n children and the maximum number of parents per child node is $(2d+1)^{n-1}$. The number of edges in the result is $n \cdot (2d+1)^{n-1}$. Note that $construct_d$ will only be called with fixed values for d and n which makes its output of constant size then.

Consecutive Set of Nodes In the following we give a construction using $construct_d$ which ensures that $S = \{s_1, \ldots, s_n\}$ are consecutive in at least one dimension in any $(d, d', 1)_{weak}$-CIG ordering. Let *seq* be this function having d, d' and S as input and a graph as output. The idea of *seq* is that $n \cdot 2d + 1$ parents which require $d' - 1$ boundary segments on their own are added to S such that they have all of $s_1, \ldots, s_n$ as additional children. Thus there is at least one parent of S which can not have one of its children consecutive to any of the s_i as the nodes of S only have $n \cdot 2d$ open sides. As this parent requires at least $d' - 1$ boundary segments for its other children, there is only one boundary segment left to cover $s_1, \ldots, s_n$. This forces $s_1, \ldots, s_n$ to be consecutive in some dimension.

$$
\begin{aligned}
seq(d, d', S) = &\left(\bigcup_{1 \le i \le 2d+1} V_i \cup S, \bigcup_{1 \le i \le 2d+1} (E_i \cup P_i) \right) \\
&\textbf{where } (V_i, E_i) = construct_d(d' - 1) \\
&\qquad\quad P_i = \{(p, s) \mid p \in parents(V_i, E_i) \wedge s \in S\}
\end{aligned}
$$

Algorithm 5: *seq*

Example 61. Figure 17 contains the result of $seq(2, 2, \{s_1, s_2\}$ together with a 2-dimensional ordering on the left that is no $(2, 2, 1)_{weak}$-CIG ordering. A $(2, 2, 1)_{weak}$-CIG ordering is on the right. Note that both orderings are depicted in a simplified way (to make the result easier to grasp) as some nodes share the same coordinate which is not possible for our 2-dimensional orderings. This, however, does not affect the important point, viz. the depicted relation of neighbourhood:s_1 has four neighbours, two in each dimension, in the left ordering. The same holds for s_2. As *seq* generated nine neighbours, there is one neighbour, say c_i, with parent p_i, such that p_i requires not two but three boundary segments to cover all its nodes. Only this ninth parent is shown, other parents are omitted. On the right side s_1 and s_2 are neighbours in σ_1, so all parents, specifically p_i has two of its three children consecutive. Thus p_i can use the remaining dimension to cover its third child c_i. By that the position of that child as well as all other children $c_1, \ldots, c_9$ that are generated by *seq*, does not matter any more and they can be placed on arbitrary positions in the 2-dimensional ordering.

Lemma 62. Let $S = \{s_1, \ldots, s_n\}$ then $G = seq(d, d', S)$ is a $(d, d', 1)_{weak}$-CIG if and only if S is consecutive in one dimension.

Proof. Assume the opposite: $G \in (d, d', 1)_{weak}$-CIG and S is not consecutive. This implies

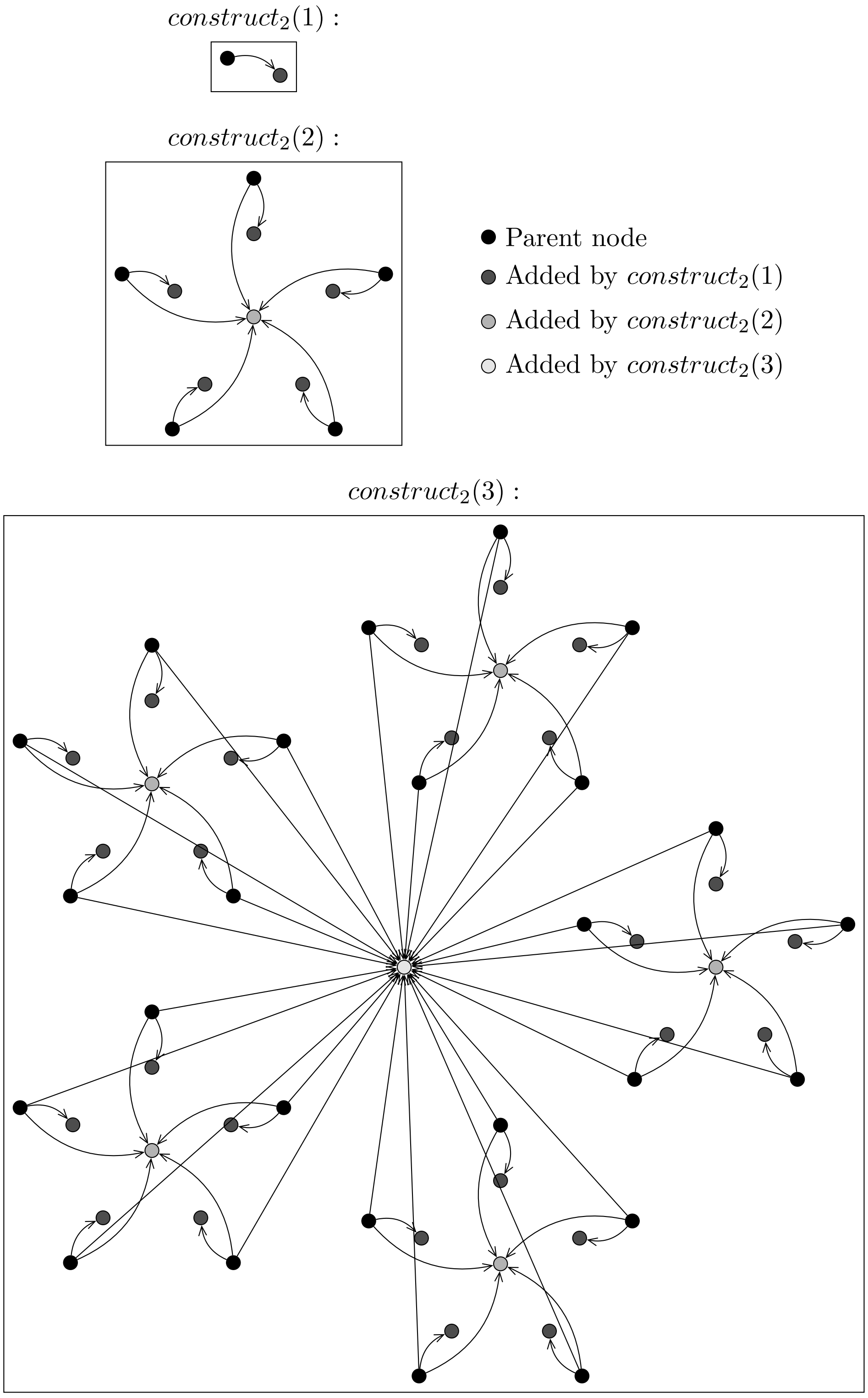

$construct_2(1)$:
$construct_2(2)$:
● Parent node
● Added by $construct_2(1)$
● Added by $construct_2(2)$
○ Added by $construct_2(3)$
$construct_2(3)$:

Figure 16: (cf. *Example* 60) The resulting graphs from $construct_2(1)$, $construct_2(2)$, and $construct_2(3)$.

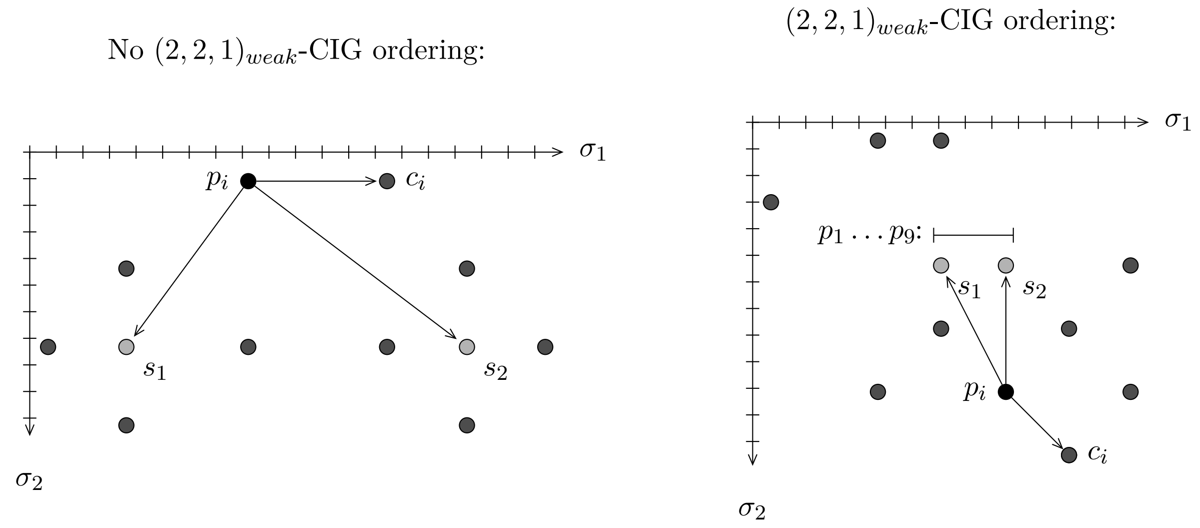


Figure 17: (cf. *Example* 61) The result of $seq(2, 2, \{s_1, s_2\})$. Only one of nine parents, p_i, is shown, but all their children $c_1, \dots, c_n$ are drawn as ●

that in any dimension at least two nodes s_i and s_{i+1} are not consecutive. By the definition of *seq* there are $n \cdot 2d+1$ parents, $p_1, \dots, p_{n \cdot 2d+1}$ of s_i and s_j which already need $d'-1$ boundary segments for their other children. Thus each has only one boundary segment left to cover the set S. As in every dimension S is not consecutive each parent can only cover one part of S, i.e., either s_i or s_j with the remaining boundary segment. We assume the parent covers s_i with the remaining boundary segment and positions one of its other children consecutive to s_j Thus this parent can cover all its children by d' boundary segments. However, the whole set S only has $n \cdot 2d$ open sides, thus at most $n \cdot 2d$ parents can position one of their children consecutive to S. As *seq* generates $n \cdot 2d + 1$ parents of this kind, there is a parent, say p_ℓ, that cannot position any of its children consecutive to a node of S. Thus p_ℓ needs two additional boundary segments to cover s_i and s_j, i.e., p_ℓ needs $d' + 1$ boundary segments. This contradicts the assumption. □

Note that on the other hand if all $s_1, \dots, s_n$ are consecutive, then the ordering of all other nodes does not matter. This is due to the fact that each parent has $d' - 1 + n$ children, where n children are consecutive, thus only need one boundary segment. Therefore $d' - 1$ boundary segments remain for $d' - 1$ children, i.e., one boundary segment per child and thus the ordering of all other children does not matter. Additionally the ordering of nodes which have no parent does not matter. By this *seq* enforces S to be consecutive, but it does not add any other constraint on the ordering of the nodes. As a parent node in a $(d, 1, 1)$-CIG forces its children to be consecutive in one dimension, *seq* has the same effect in a $(d, d', 1)_{weak}$-CIG, i.e., all s_i are consecutive in one dimension.

Reduction by Simulating Children Sets Recall that a parent node in a $(d, 1, 1)$-CIG simply ensures that all its children are consecutive in one of the d dimensions. This is the same effect

$seq(d, d', S)$ has on a set of nodes S, i.e., *seq* ensures that for every $(d, d', 1)_{weak}$-CIG ordering of the graph $G' = seq(d, d', S)$ all nodes $s \in S$ are consecutive in one dimension. Thus *seq* can be used to simulate the parent relationship of a $(d, 1, 1)$-CIG inside a $(d, d', 1)_{weak}$-CIG. The reduction $red_{d,d'}$ subsitutes every parent node with the *seq* construction on its children:

$$red_{d,d'}(V, E) = \left(\bigcup_{1 \leq i \leq n} V_n, \bigcup_{1 \leq i \leq n} E_i \right)$$

$$\textbf{where } parents(V, E) = \{p_1, \ldots, p_n\} \text{ and}$$
$$\text{for } 1 \leq i \leq n : (V_i, E_i) = seq(d, d', children(p_i))$$

Algorithm 6: $red_{d,d'}$

Example 63. In Figure 18 a small graph G is shown on top. Below the application of the reduction function is depicted where a rectangle represents the output graph from one call to *seq*. Thus a rectangle in Figure 18 is a construction enforcing that the nodes inside, e.g., A, B, C, are consecutive in one dimension. Note that nodes which have no parent in G are not elements of the reduction graph. This is due to the fact that nodes which have no parent add no constraints of placements on the possible orderings, i.e., nodes without any parent can be placed at the end of every ordering. Of course every parent adds constraints by its set of children, but those are kept by the reduction.

Theorem 64. *$(d, 1, 1)$-CIG is polynomially reducible to $(d, d', 1)_{weak}$-CIG, i.e., $(d, 1, 1)$-CIG $\leq_p$ $(d, d', 1)_{weak}$-CIG.*

Proof. Let $G = (V, E) \in (d, 1, 1)$-CIG and $G' = red_{d,d'}(V, E)$. By the definition of $red_{d,d'}$ and *seq* all children of every parent in G are forced to be consecutive in G' in one dimension. If there is a $(d, 1, 1)$-CIG ordering of the children in d dimensions for G then the same ordering of nodes $v \in V$ is a $(d, d', 1)_{weak}$-CIG ordering for G'. The ordering of the additional nodes of G' does not matter (see remarks on *seq*). Those additional nodes are placed arbitrarily on the lower end of the ordering for G, effectively yielding a d-dimensional ordering of nodes $v \in V'$. This ordering is a $(d, d', 1)_{weak}$-CIG ordering of G' as every constraint on the placement of nodes is fulfilled. On the other hand, if there is an ordering for a $(d, d', 1)_{weak}$-CIG labelling of the nodes of G' then the same ordering is sufficient for a $(d, 1, 1)$-CIG labelling of G (modulo the additional nodes of G', i.e., $V' \setminus V$).

As d and d' are fixed and $construct_d$ thus generates a constant number of nodes per instantiation, the reduction is polynomial. Likewise, for fixed d, d' $seq(d, d', S)$ is linear in S and thus $red_{d,d'}$ is linear in the number of edges of the input graph. □

NP-Completeness of $(d, d', k)_{weak}$-CIG Recognition Problems Recall the definition of $(d, d', k)_{weak}$-CIG labellings where the set of children I_v of node v must satisify:

$$\bigcup_{J \in I_v} \bigcup_{dim:[b,e] \in J} \{v' \in V \mid \sigma_{dim}(v') \in [b, e]\}$$

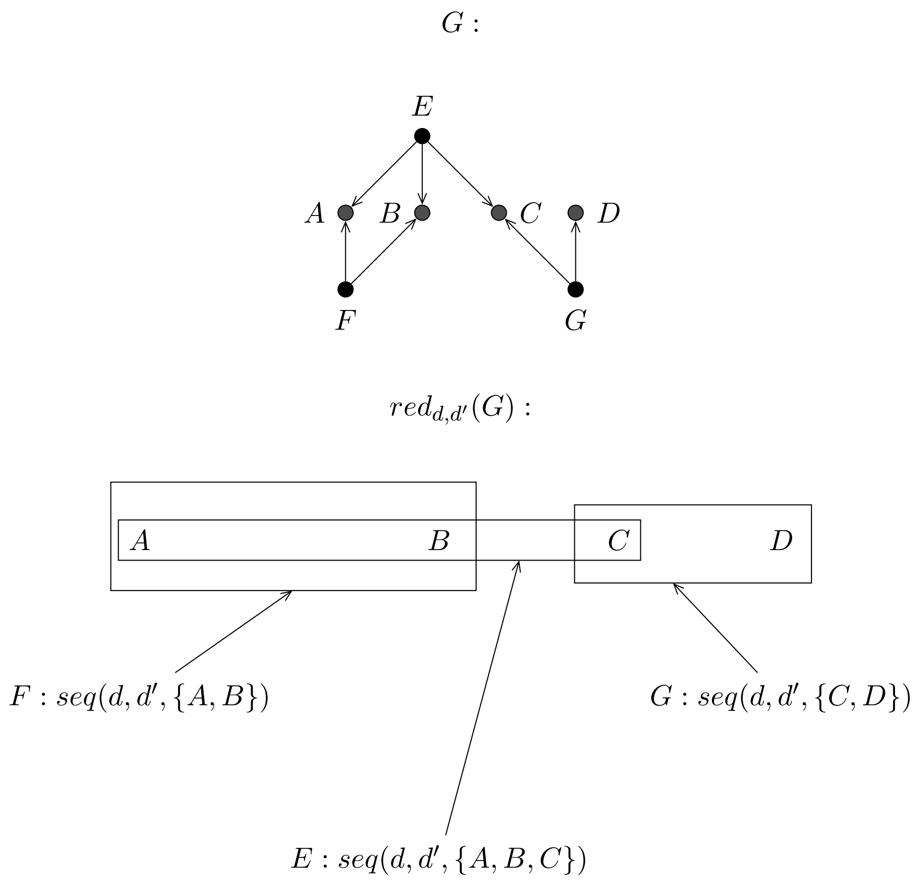

$G:$
E
A
B
C
D
F
G
$red_{d,d'}(G):$
A
B
C
D
$F: seq(d, d', \{A, B\})$
$G: seq(d, d', \{C, D\})$
$E: seq(d, d', \{A, B, C\})$

Figure 18: (cf. *Example* 63) A graph and the sets of children enforced to form sequences by $red_{d,d'}$.

In other words the set of children of node v is the union over all intervals where each interval is the union over its boundary segments. The intuition behind the reduction to the recogntion problems of $(d, d', k)_{weak}$-CIG labelling schemes is that those two unions can be written as one union:

$$\bigcup_{J \in I_v} \bigcup_{dim:[b,e] \in J} \{v' \in V \mid \sigma_{dim}(v') \in [b, e]\} = \bigcup_{\text{dim}:[b,e,] \in bigcupI_v} \{v' \in V \mid \sigma_{dim}(v') \in [b, e]\}$$

By that multiple intervals per node just add more boundary segments to the union, but the basic operations do not change. Note that this is not true for the strong cases as there is a union over intersections and not a union over unions. In the weak case, however, the two unions can be merged into one union. By that every node v in a $(d, d', k)_{weak}$-CIG has at most $k \cdot d'$ boundary segments whose union is the set of children of v. This observation yields a sequence enforcing construction $seq(d, d', k, S)$ using the same technique as $seq(d, d', S)$, but consuming $k \cdot d'$ boundary segments per node instead of only d'. Thus the sequence S is forced to be consecutive in one dimension if one node needs at least $(k \cdot d') - 1$ boundary segments for its other children, such that this node has only one boundary segment left to cover the whole sequence $S = \{s_1, \ldots, s_n\}$. For k intervals of d' boundary segments the function $construct_d$ has to be applied with a recursive depth of $d' \cdot k$. As the second value of seq is the depth of the recursion of $construct_d$ used inside seq, it is sufficient to increase that value. Thus the function $seq(d, d', k, S)$ which ensures that S is consecutive in one dimension under a $(d, d', 1)_{weak}$-CIG labelling is:

$$seq(d, d', k, S) = seq(d, d' \cdot k, S)$$

The full reduction for $(d, d', k)_{weak}$-CIG in principle is the same as for $(d, d', 1)_{weak}$-CIG with the only difference being a recursion depth of $d'k$ instead of only d' for sequence enforcing. Again, we use the reduction function for $(d, d', 1)_{weak}$-CIG to define the reduction for $(d, d', k)_{weak}$-CIG:

$$red_{d,d',k}(V, E) = red_{d,d' \cdot k}(V, E)$$

Clearly, this reduction function $red_{d,d',k}$ is still polynomial if d, d', k are fixed as it inherits the complexity of $red_{d,d'}$ then.

Theorem 65. *$(d, 1, 1)$-CIG is polynomially reducible to $(d, d', k)_{weak}$-CIG, i.e., $(d, 1, 1)$-CIG $\leq_p$ $(d, d', k)_{weak}$-CIG*

Proof. Assume some graph $G = (V, E)$ with $G \in (d, 1, 1)$-CIG and $G' = red_{d,d',k}(G)$. We have to show that $G' \in (d, d', k)_{weak}$-CIG, i.e., we need a d-dimensional ordering such that every parent in G' can cover all its children with at most k d'-dimensional intervals. By assumption there is a d-dimensional ordering Σ of the nodes of G such that every parent of G has its children consecutive in one dimension. We take this ordering Σ and add the additional nodes of the reduction function to this ordering such that the added nodes are right of all original nodes in every dimension. We call this ordering Σ' and show that G under Σ' is a $(d, d', k)_{weak}$-CIG. Note that the ordering among the added nodes in Σ' is arbitrary. In G there are only parents built by the *seq*. In Σ' every sequence is consecutive in one dimension, as it is so in Σ. It remains to show that each parent p in G' can cover its remaining $d'k - 1$ children with its $d'k - 1$ remaining boundary segments. This, however, is trivial.

Assume for some graph $G = (V, E)$ with $G \notin (d, 1, 1)$-CIG and $G' = red_{d,d',k}(G)$. We have to show that $G' \notin (d, d', k)_{weak}$-CIG. By assumption there is no d-dimensional ordering for G such that all parents have all their children consecutive. By definition of $red_{d,d',k}$ there is therefore a sequence S in any ordering of G' such that S is not consecutive in any dimension. By definition of $red_{d,d',k}$, however, there is at least one parent p of S which requires $d'k$ boundary segments for all its other children, i.e., p needs more than $d'k$ boundary segments to cover all its other children. Thus $G' \notin (d, d', k)_{weak}$-CIG. $\square$

5.4 Equivalence of the Recognition Problems of Weak and Strong $(d, d', 1)$-CIG Labelling Schemes

To show that weak and strong are equivalent if only one interval per node is allowed, observe that the one is a union over CIG labellings while the other is the intersection over CIG labellings. As the class of CIG is closed under complement, the equivalence of weak and strong in principle is an application of DeMorgan's Laws.

Thus proof for the equivalence of the recognition problem on weak and strong CIG labellings is based on the observation that a graph G is a weak CIG if and only if its complement $\overline{G}$ is a strong CIG with the same parameters on d and d'. The proof for this observation consists of two parts: First, the decomposition of a d-dimensional CIG into d one-dimensional CIGs in Section 5.4.1. Those one-dimensional CIGs can be complemented with each complement being a one-dimensional CIG again, as the class of CIG is closed under complement. Second, the composition of the complemented CIGs into a d-dimensional CIG in Section 5.4.2: If decomposition and composition are defined such that the mode of operation is union for decomposition and intersection for composition, (or vice versa), then a $(d, d', 1)_{weak}$-CIG can be transformed into a $(d, d', 1)_{strong}$-CIG, (or vice versa). It remains to show that the resulting graph after composition indeed is the complement of the decomposed one. The overall idea of this approach can be summarized as: We show that $(d, d', 1)$-CIG obey the DeMorgan's Laws.

The following definition specifies how the information of a (d, d', k)-CIG can be stored in a relational context. This explicit specification of all data to represent a (d, d', k)-CIG is also used in this proof.

Definition 66 (CIG Representation)**.** Let $G = (V, E)$ be a (d, d', k)-CIG. As the definition of (d, d', k)-CIG only requires the existence of a d-dimensional ordering of V, there may be multiple such orderings. Let $\Sigma_G = \{1{:}\sigma_1, \ldots d{:}\sigma_d\}$ be one such ordering. A CIG representation $R_G = (\Sigma_G, M_G)$ of G is a pair consisting of the d-dimensional ordering Σ_G, and a set of 5-tuples M_G describing boundary segments. The information per 5-tuple is: To which node this boundary segment belongs, the number of the multi-dimensional interval which this boundary segment is part of, in which dimension the boundary segment is located, and finally the begin and end of the boundary segment itself. So $(v_3, 4, 3, 1, 7)$ is read as node v_3 contains in its fourth interval a boundary segment from $1 - 7$ in the third dimension, i.e., $3{:}[1, 7] \in int_4$ while $int_4 \in I_{v_3}$. Formally $M_G \subseteq V \times \mathbb{N}_k \times \mathbb{N}_d \times \mathbb{N}_{|V|} \times \mathbb{N}_{|V|}$. For a given (d, d', k)-CIG the following properties hold for M_G:

- Each interval uses at most d' dimensions, i.e.,

$$\forall v \in V \forall 1 \leq i \leq k : |\{dim|(v, i, dim, s, e) \in M_G\}| \leq d'$$

- The edge relation is reconstructible from the intervals, i.e., $(v, v') \in E \iff$

$$v' \in \bigcup_{i=0}^{k} \begin{cases} \bigcup_{j=1}^{d} \begin{cases} \{x \in V|\, (v, i, j, s, e) \in M_G \wedge \sigma_i(x) \in [s, e]\} \\ \emptyset \qquad\qquad \text{if } \nexists(v, i, j, s, e) \in M_G \end{cases} & \text{for } weak \\ \bigcap_{j=1}^{d} \begin{cases} \{x \in V|\, (v, i, j, s, e) \in M_G \wedge \sigma_i(x) \in [s, e]\} \\ V \qquad\qquad \text{if } \nexists(v, i, j, s, e) \in M_G \end{cases} & \text{for } strong \end{cases}$$

An empty interval, i.e., no children in one dimension, is represented by the special interval from 0 to 0, i.e., an entry $(v, k, i, 0, 0)$. In the following we omit the indices in R_G, Σ_G, and M_G if clear from the context.

5.4.1 Decomposition of $(d, d', 1)$-CIG

Let $G = (V, E) \in (d, d', 1)$-CIG with a CIG representation $R_G = (\sigma, M)$. The decomposition first partitions the representation along the dimensions into $M^1, \ldots, M^d$ partitions. The later proof is simplified if all multidimensional intervals consume all available dimensions, thus our decomposition fills up all multidimensional intervals so they consume all d dimensions. So for $1 \leq \ell \leq d$ each partition M^ℓ is defined as $M^\ell = \{(n, 1, \ell, s, e) \mid (n, 1, \ell, s, e) \in M\} \cup \begin{cases} \{(n, 1, \ell, 1, |V|)\} & \exists(n, 1, \ell', s, e) \in M \wedge \nexists(n, 1, \ell, s, e) \in M \text{ for weak} \\ \{(n, 1, \ell, 0, 0)\} & \exists(n, 1, \ell', s, e) \in M \wedge \nexists(n, 1, \ell, s, e) \in M \text{ for strong} \\ \emptyset & \text{otherwise} \end{cases}$

Note that the partitions represent the original G, i.e., the union over all M^ℓ yields a $(d, d, 1)$-CIG representation of G. This is due to the “filling up” being neutral w.r.t. weak or strong, i.e., the whole set V is added if intersection is used while an empty set is added if union is used.

In the next step each M^ℓ is used to fill the entries of an adjacency matrix $B^\ell \in \{0, 1\}^{|V| \times |V|}$ such that $(b^\ell_{ij}) = \begin{cases} 1 & \text{if } (i, 1, \ell, s, e) \in M \wedge j \in [s, e] \\ 0 & \text{otherwise} \end{cases}$

Observe that each node from G only has one multi-dimensional interval per definition of $(d, d', 1)$-CIG. Such an interval is defined to have at most one boundary segment per dimension. Thus a partitioning into dimensions yields in every dimension one boundary segment per node. Therefore each row in M^ℓ contains only this one boundary segment, i.e., there is only one consecutive block of 1’s in each row of M^ℓ. Note that due to the previous “filling up” there is indeed a tuple per every row of B^ℓ. This makes B^ℓ a CIG. Note that B^ℓ is formed from the intervals of dimension ℓ and those intervals are defined under the ordering $\sigma_\ell \in \Sigma$. Thus B^ℓ has the permutation σ_ℓ applied on its columns. For convenience the adjacency matrix $A^\ell = \sigma_\ell^{-1}(B^\ell)$ is defined as the matrix where all columns are in the standard order, i.e., column j of A^ℓ represents $j = v$ and not $j = \sigma_\ell(v)$ as in B^ℓ. Those matrices A^ℓ clearly are CIGs as they are just column-permutated CIGs and the definition of the CIG labelling only requires that a column permutation (ordering), exists such that the resulting matrix has all 1's consecutive in every row.

Corollary 67. All decomposed matrices A^ℓ are CIGs.

5.4.2 Composition of $(d, d', 1)$-CIG

Let $R_G = (\sigma, M)$ be a $(d, d', 1)$-CIG representation and the d adjacency matrices $A^1, \ldots, A^d$ be the result of a decomposition as described above. Let $\overline{A^1}, \ldots, \overline{A^d}$ be the complements of $A^1, \ldots, A^d$. The composition $A' = \begin{cases} \bigcup_{\ell=1}^d \overline{A^\ell} & \text{in weak case} \\ \bigcap_{\ell=1}^d \overline{A^\ell} & \text{in strong case} \end{cases}$ is a $(d, d', 1)$-CIG again. First observe that every $\overline{A^\ell}$ is a CIG as this class of graphs is closed under complement. This is actually due to the fact that we allow circular intervals, i.e., either the 1's are in one consecutive interval or the 0's are. Then it follows from the definition of the (d, d', k)-CIG labelling schemes that $A' \in (d, d, 1)$-CIG, as every node requires either one interval made up of d one-dimensional boundary segments from $\overline{A^\ell}$, or has no children at all.

Lemma 68. Let $G \in (d, d', 1)$-CIG with decomposition $A^1, \ldots, A^d$ and complements $\overline{A^1}$, $\ldots$, $\overline{A^d}$. Then the composition A' also represents a $(d, d', 1)$-CIG.

Proof. From the above it is clear that A' is a $(d, d, 1)$-CIG. It remains to show that every node requires only a d'-dimensional interval, i.e., that $d - d'$ dimensions contain superfluous boundary segments which can be dropped from each interval without actually changing A'. To see that this indeed is possible, consider the case where a node $v \in V$ only requires a d'-dimensional interval in the representation of R_G. This d'-dimensional interval is filled up to a d-dimensional interval by inserting either an empty boundary segment in the weak case, or a boundary segment over all nodes in the strong case. Thus there exist $d - d'$ dimensions such that the boundary segments of v for that dimensions are trivial. Complementing the graph only replaces an empty boundary segment for a boundary segment over all nodes and vice versa. Thus these boundary segments are not needed for v and can be removed. There exists a representation $R_{A'}$ only needing one d'-dimensional interval per node, i.e., the graph G with $A' = (adj(G)$ can be labelled according to the $(d, d', 1)$-CIG labelling scheme. □

5.4.3 Reduction between Weak and Strong

Theorem 69.

$$G \in (d, d', 1)_{weak}\text{-}CIG \Leftrightarrow \overline{G} \in (d, d', 1)_{strong}\text{-}CIG$$

Proof. First we show: $G \in (d, d', 1)_{weak}$-CIG $\Rightarrow \overline{G} \in (d, d', 1)_{strong}$-CIG

Let A be the adjacency matrix of G. By Lemma 67 there exists a decomposition $A^1, \ldots, A^d$ of G. All A^ℓ are then complemented and the composition for the strong case is applied to the complements. By Lemma 68 this results in an adjacency matrix $\overline{A}$. It remains to show that $\overline{A}$ is the complement of A.

$$\begin{aligned}
\text{If } (a_{ij}) = 0 \text{ for some entry} &\Rightarrow (a^{\ell}_{ij}) = 0 \;\text{ for all }\; 1 \leq \ell \leq d && \text{(decomposition of weak)}\\
&\Rightarrow (\overline{a^{\ell}_{ij}}) = 1 \text{ for all } 1 \leq \ell \leq d && \text{(complement)}\\
&\Rightarrow \bigcap_{\ell=1}^{d} (\overline{a^{\ell}_{ij}}) = 1 && \text{(composition of strong)}\\
&\Rightarrow (\overline{a_{i,j}}) = 1
\end{aligned}$$

$$\begin{aligned}
\text{If } (a_{i,j}) = 1 \text{ for some entry} &\Rightarrow \exists \ell : (a^{\ell}_{ij}) = 1 && \text{(decomposition of weak)}\\
&\Rightarrow \exists \ell : (\overline{a^{\ell}_{ij}}) = 0 && \text{(complement)}\\
&\Rightarrow \bigcap_{\ell=1}^{d} (\overline{a^{\ell}_{ij}}) = 0 && \text{(composition of strong)}\\
&\Rightarrow (\overline{a_{ij}}) = 0
\end{aligned}$$

Second we show: $G \in (d, d', 1)_{strong}$-CIG $\Rightarrow \overline{G} \in (d, d', 1)_{weak}$-CIG

Let $\overline{A}$ be the adjacency matrix of $\overline{G}$. By Lemma 67 there exists a decomposition $\overline{A^1}, \ldots, \overline{A^d}$ of $\overline{G}$. All A^{ℓ} are then complemented and the composition for the weak case is applied to the complemented matrices. By Lemma 68 this results in an adjacency matrix A. It remains to be shown that A is the complement of $\overline{A}$.

$$\begin{aligned}
\text{If } (\overline{a_{ij}}) = 1 \text{ for some entry} &\Rightarrow (\overline{a^{\ell}_{ij}}) = 1 \text{ for all } 1 \leq \ell \leq d && \text{(decomposition of strong)}\\
&\Rightarrow (a^{\ell}_{ij}) = 0 \text{ for all } 1 \leq \ell \leq d && \text{(complement)}\\
&\Rightarrow \bigcup_{\ell=1}^{d} (a^{\ell}_{ij}) = 0 && \text{(composition of weak)}\\
&\Rightarrow (a_{ij}) = 0
\end{aligned}$$

$$\begin{aligned}
\text{If } (\overline{a_{ij}}) = 0 \text{ for some entry} &\Rightarrow \exists \ell : (\overline{a^{\ell}_{ij}}) = 0 && \text{(decomposition of strong)}\\
&\Rightarrow \exists \ell : (a^{\ell}_{ij}) = 1 && \text{(complement)}\\
&\Rightarrow \bigcup_{\ell=1}^{d} (a^{\ell}_{ij}) = 1 && \text{(composition of weak)}\\
&\Rightarrow (a_{ij}) = 1
\end{aligned}$$

□

Thus the reductions from $(d, d', 1)_{weak}$-CIG to $(d, d', 1)_{strong}$-CIG and vice versa are as follows: Take the adjacency matrix A of the $(d,d',1)_{weak}$-CIG or $(d,d',1)_{strong}$-CIG, respectively, graph and complement it. The complemented matrix $\overline{A}$ is a $(d,d',1)_{weak}$-CIG or $(d,d',1)_{strong}$-CIG, respectively, exactly if the original matrix is a $(d,d',1)_{strong}$-CIG or $(d,d',1)_{weak}$-CIG, respectively. Obviously, this reduction is linear in time.

5.5 The Hierarchy of (d,d',k)-CIG

In this section it is shown that the hierarchy of (d,d',k)-CIG is proper, i.e., does not collapse on some layer. It follows from the definition of the (d,d',k)-CIG labelling schemes that (d_1,d'_1,k_1)-CIG $\subseteq$ (d_2,d'_2,k_2)-CIG for $d_1 \leq d_2$, $d'_1 \leq d'_2$, and $k_1 \leq k_2$, i.e., one class of graphs includes all classes with smaller numerical values. Thus it only remains to show that those hierarchies indeed are proper and do not collapse. Alas we do not show that the (d,d',k)-CIG labelling schemes form proper hierarchies between all possible combinations of labelling schemes of (d,d',k)-CIG, but we give numerous instances where proper hierarchies exist between several subclasses of (d,d',k)-CIG labelling schemes.

Theorem 70. $(1,1,k)$-*CIG* $\subsetneq$ $(1,1,k+1)$-*CIG*

Proof. This is a direct consequence from Theorem 32 and the equivalence of $(1,1,k)$-CIG and k-CIG. □

Theorem 71. $(d,1,1)$-*CIG* $\subsetneq$ $(d+1,1,1)$-*CIG*

Proof. Consider a $d+1$-colorable graph G which is not d-colorable and let $G' = red_d(G)$ be the result of the reduction from d-Colorability to $(d,1,1)$-CIG. It follows from theorem 58 that $G' \in (d+1,1,1)$-CIG and $G' \notin (d,1,1)$-CIG, thus $(d,1,1)$-CIG $\subsetneq$ $(d+1,1,1)$-CIG. □

The function $construct_d$ (see Algorithm 4) from the NP-completeness proof of the recognition problem of $(d,d',1)$-CIG is useful to show that increasing the number of dimensions over which intervals extend leads to labelling schemes which cover more graphs.

Theorem 72. *Let* $G = (V,E)$ *and* $G = construct_d(d')$ *then* $G \in (d,d',1)_{weak}$-*CIG and* $G \notin (d,d'-1,1)_{weak}$-*CIG.*

Proof. By the definition of $construct_d$ each parent node gets one additional child per step. So each parent in G has exactly d' children and thus $G \in (d,d',1)_{weak}$-CIG as each parent has d' boundary segments to cover its d' children.

It remains to be shown that at least one node requires d' boundary segments. We show this by induction on $construct_d$ with an induction hypothesis of: $construct_d(n)$ produces a graph such that for any d-dimensional ordering there is one node which requires n boundary segments in a $(d,n,1)_{weak}$-CIG. For $construct_d(1)$ this is obviously true as each parent has one child, thus each parent needs one boundary segment. In the inductive case of $construct_d(n+1)$ there are $2d+1$ recursive calls of $construct_d(n)$, thus we assume by hypothesis that for any n-dimensional ordering there are at least $2d+1$ parents $p_1,\ldots,p_{2d+1} \in V$ which each needs n boundary segments for its children. Each of these parents has s as an additional child in E. As s has only $2d$ open sides, at least one parent p_i can not have any of its children consecutive to s. Thus p_i needs $n + 1$ boundary segments to cover its children. □

It follows directly that the $(d, d', 1)_{weak}$-CIG labelling schemes induce a proper hierarchy.

Corollary 73. $(d, d', 1)_{weak}$-CIG $\subsetneq (d, d' + 1, 1)_{weak}$-CIG

By the equivalence of $(d, d', 1)_{weak}$-CIG and $(d, d', 1)_{strong}$-CIG the above results apply to strong graphs.

Theorem 74. *$(d, d', 1)_{strong}$-CIG $\subsetneq (d, d' + 1, 1)_{strong}$-CIG*

Proof. By Theorem 72 there exists a graph G with $G \notin (d, d', 1)_{weak}$-CIG and $G \in (d, d' + 1, 1)_{weak}$-CIG. By Theorem 69 the complement $\overline{G}$ of G then is no $(d, d', 1)_{strong}$-CIG, but a $(d, d' + 1, 1)_{strong}$-CIG. Thus $\overline{G} \notin (d, d', 1)_{weak}$-CIG and $\overline{G} \in (d, d' + 1, 1)_{strong}$-CIG. □

The reasoning for the proper hierarchy of $(d, d', 1)_{weak}$-CIG labelling can be extended to show that $(d, d', k)_{weak}$-CIG $\subsetneq (d, d', k + 1)_{weak}$-CIG and $(d, d', k)_{weak}$-CIG $\subsetneq (d, d' + 1, k)_{weak}$-CIG. In the first case a graph $G = construct_d(d' \cdot k)$ is employed to show that $G \in (d, d', k)_{weak}$-CIG but $G \notin (d, d', k - 1)_{weak}$-CIG. In the second case the same graph G is employed to show that $G \in (d, d', k)_{weak}$-CIG but $G' \notin (d, d' - 1, k)_{weak}$-CIG. Both proofs inherit the reasoning from Theorem 72.

Theorem 75. *Let $G = (V, E)$ and $G = construct_d(d' \cdot k)$ then $G \in (d, d', k)_{weak}$-CIG, $G \notin (d, d', k - 1)_{weak}$-CIG, and $G \notin (d, d' - 1, k)_{weak}$-CIG.*

Proof. By the definition of $construct_d$ each parent node gets one additional child per step. So each parent in G has exactly $d' \cdot k$ children and thus $G \in (d, d', k)_{weak}$-CIG as each parent has $d' \cdot k$ boundary segments to cover its $d' \cdot k$ children.

It remains to be shown that at least one node requires $d' \cdot k$ boundary segments. $construct_d(d' \cdot k)$ has already been proven for Theorem 72 to construct a graph which contains for any ordering a parent node that requires $d' \cdot k$ boundary segments. As no parent node of a $(d, d', k - 1)_{weak}$-CIG and no parent node of a $(d, d' - 1, k)_{weak}$-CIG has $d' \cdot k$ boundary segments, it follows that G is no element of either of the two. □

Corollary 76. $(d, d', k)_{weak}$-CIG $\subsetneq (d, d', k + 1)_{weak}$-CIG and $(d, d', k)_{weak}$-CIG $\subsetneq (d, d' + 1, k)_{weak}$-CIG

6 Conclusion

This section summarizes the investigations on hierarchies of the proposed labelling schemes, recapitulates NP-completeness of the recognition problems associated to those labelling schemes, and identifies directions for further research based on our results.

6.1 Summary of Formal Results

The formal results of this thesis are listed in Table 1. Note that in our investigation all natural extensions of the CIG labelling scheme lead to NP-complete recognition problems even at the first step for every extension: The recognition problem of the extension to two intervals per node, i.e., 2-CIG, is NP-complete, that to two orderings per graph, i.e., $(2,1,1)$-CIG, is NP-complete, and that to two-dimensional intervals per node, i.e., $(2,2,1)$-CIG, also is NP-complete. This leaves only the original CIG, i.e., $(1,1,1)$-CIG, with a polynomial recognition algorithm and the class of interval digraphs is the only proper superclass of CIG, whose recognition problem is not NP-complete. As our investigation shows every extension to be NP-complete on their own, we conjecture that the unproven cases where those extensions are combined, i.e., $(d,d',k)_{strong}$-CIG labelling schemes, are NP-complete as well.

Extension for	CIG scheme	Proper Hierarchy	NP-complete for
multiple intervals	k-CIG $= (1,1,k)$-CIG	yes	$k \geq 2$
multiple orderings	$(d,1,1)$-CIG	yes	$d \geq 2$
multi-dimensional intervals	$(d,d',1)$-CIG	yes	any $d' \leq d$ and $d \geq 2$
any combination of the above	(d,d',k)-CIG	weak: yes strong: yes[1]	weak: if one of $d,d',k \geq 2$ strong: if one of $d,d',k \geq 2$[1]

Table 1: Summary of extended CIG labelling schemes

6.2 Future Work

As all of our natural extensions of the CIG labelling scheme fail to allow polynomial labelling algorithms, we think that further work should focus on that issue. Although there are white spots in our investigation of the class of $(d,d',k)_{strong}$-CIG like do the $(d,d',k)_{strong}$-CIG labelling schemes all have NP-complete recognition problems, or do they all form proper hierarchies, we strongly conjecture that the answer is yes to both. Instead, we consider the following problems most relevant for further research:

1. Can any of the labelling schemes be applied by approximations?
2. Are there heuristics for any of the labelling schemes?
3. If only certain classes of graphs are considered, does any of the labelling schemes become polynomial?

[1]Conjectured by the author.

4. Which classes of graphs occur in real-world data?
5. Are there extensions of interval digraphs with tractable labelling algorithms?

The following paragraphs describe every question in more detail.

Question 1 A general approach to solving NP-complete optimization problems is by approximation. An approximation is an algorithm which runs in polynomial time and solves the problem, but may not return the optimum solution. The difference to heuristics is that approximations have guaranteed performance, i.e., the returned solution is guaranteed to be close to the optimum solution. For an introduction to approximations see [2]. It is notable that for the k-Consecutive Ones Property, i.e. the same property underlying the k-CIG labelling scheme, an algorithm for approximation is reconstructible from [3]. Also, for the similar CBM problem a polynomial approximation is known whose result is not worse than 1.5 times the optimum result.

Question 2 Heuristics are similar to approximations though they lack the guarantee of finding a solution close to the optimum. Heuristics can be applied not only to find a good value for k such that a graph can be labelled by the k-CIG labelling scheme, but they can also be used to find a maximum part of a graph to which a CIG labelling can be applied. Two such heuristics are mentioned in [11], namely the Consecutive Ones Submatrix and the Consecutive Ones Matrix Augmentation. The first can be used to find the largest subgraph which can be labelled according to the original CIG scheme, while the second can be employed to finding the minimum number of edges that have to be added to a graph such that the result is a CIG. However, both problems are NP-complete [11] as they aim at finding optimum solutions again. The first two approaches enable heuristics or data structures which use the CIG labelling scheme, but allow parts of a graph being not labelled. The part not being labelled must be stored additionally then, for example in an adjacency list. Matrix augmentation gives rise to a data structure which uses the CIG labelling scheme together with some way to mark edges as augmented only, i.e., those edges which have been added by the matrix augmentation. If a query addresses such a data structure the marked edges then can be subtracted from each result set. Note that both heuristics still have constant time and space characteristics (on average per node) if the number of edges which are treated in a special way is linear in the size of the input graph.

Question 3 Reconsider the NP-complete boxicity problem which is similar to our multidimensional extensions of the CIG labelling. As already noted this problem becomes tractable when restricted to certain graph classes [5] like chordal graphs, cographs, circular arc graphs, chordal bipartite graphs, permutation graphs, or circle graphs. It is an open question whether the same classes of graphs yield polynomial labelling algorithms for any of our extensions of the CIG labelling scheme.

Question 4 In this thesis we formally show that no algorithm exists to compute one of the extended labelling schemes efficiently on any data (unless P=NP). However, there is a high possibility that real data is restricted to some special graph classes. By real data we understand graphs occuring in the Web and relations which are not made artificially, but represent relations between objects from the real world. If such real data is restricted, even if only in a certain area of application, this restriction might give rise to either polynomial labelling algorithms in that area or prove heuristics to be of use in that application area.

Question 5 Interval digraphs are a proper superclass of those graphs which can be labelled according to the original CIG scheme [19]. In [21] an algorithm testing whether a given directed graph is an interval digraph is given. This algorithm runs in $O(nm^6(n+m)\log n)$ where n is the number of nodes and m the number of edges in a given graph. The interval digraphs, or zero-partitionable adjacency matrices, are the only known superclass of graphs of CIG which are polynomially recognizable and yield a labelling with the desired constant time and space properties. A number of interesting objectives arise in the context of interval digraphs:

- Design a query evaluation algorithm using graph labelling based on the zero-partition property.
- Investigate if the testing algorithm for the zero-partition property is acceptable on real data.
- Investigate if there is a better recognition algorithm than the known one.
- Investigate generalizations of the zero-partition pattern such that even more graphs can be covered.

References

[1] J. E. Atkins and M. Middendorf. On physical mapping and the consecutive ones property for sparse matrices. *Discrete Applied Mathematics*, 71:23–40, 1996.

[2] G. Ausiello, P. Crescenzi, V. Kann, A. Marchetti-Spaccamela, G. Gambosi, and A. M. Spaccamela. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties.* Springer, January 2000.

[3] V. Bilò, V. Goyal, R. Ravi, and M. Singh. On the crossing spanning tree problem. In *Proceedings of Workshop on Aproximation Algorithms for Combinatorial Optimization Problems*, pages 51–60, 2004.

[4] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976.

[5] L. S. Chandran and N. Sivadasan. Boxicity and treewidth. *Journal of Combinatorial Theory, Series B*, 97(5):733–744, 2007.

[6] M. Dom and R. Niedermeier. The search for consecutive ones submatrices: Faster and more general. In H. Broersma, S. Dantchev, M. Johnson, and S. Szeider, editors, *Proceedings of the Algorithms and Complexity Workshop*, volume 9 of *Texts in Algorithmics*, pages 43–54. College Publications London, 2007.

[7] M. Flammini. On the hardness of devising interval routing schemes. *Parallel Processing Letters*, 7(1):39–47, 1997.

[8] M. Flammini, G. Gambosi, U. Nanni, and R. B. Tan. Multi-dimensional interval routing schemes. In *Proceedings of International Workshop on Distributed Algorithms*, pages 131–144, 1995.

[9] D. Fulkerson and O. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15:835–855, 1965.

[10] T. Furche. *Implementation of Web Query Language Reconsidered: Beyond Tree and Single-Language Algebras at (Almost) No Cost.* Dissertation/doctoral thesis, Ludwig-Maxmilians University Munich, 2008.

[11] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[12] P. W. Goldberg, M. C. Golumbic, H. Kaplan, and R. Shamir. Four strikes against physical mapping of DNA. *Journal of Computational Biology*, 2(1):139–152, 1995.

[13] T. Grust. Accelerating xpath location steps. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 109–120, New York, NY, USA, 2002. ACM.

[14] S. Haddadi. A note on the NP-hardness of the consecutive block minimization problem. *International Transactions in Operational Research*, 9(6):775–777, 2002.

[15] S. Haddadi and Z. Layouni. Consecutive block minimization is 1.5-approximable. *Information Processing Letters*, 108(3):132–135, 2008.

[16] W.-L. Hsu. PC-trees vs. PQ-trees. In *Proceedings of the International Conference on Computing and Combinatorics*, pages 207–217, 2001.

[17] L. T. Kou. Polynomial complete consecutive information retrieval problems. *SIAM Journal of Computing*, 6(1):67–75, 1977.

[18] J. Kratochvíl. A special planar satisfiability problem and a consequence of its NP-completeness. *Discrete Applied Mathematics*, 52(3):233–252, 1994.

[19] I. Lin, M. K. Sen, and D. B. West. Classes of interval digraphs and 0,1-matrices. In *Proceedings of Conference on Comb. Graph. Th.*, pages 201–209, 1997.

[20] W. J. Lipski. Two NP-complete problems related to information retrieval. In *Proceedings of the International Fundamentals of Computation Theory-Conference*, pages 452–458, 1977.

[21] H. Müller. Recognizing interval digraphs and interval bigraphs in polynomial time. *Discrete Applied Mathematics*, 78(1-3):189–205, 1997.

[22] D. Osmani. Matrices with the consecutive ones property, interval graphs and their applications. Master's thesis, University of Kaiserslautern, Germany, June 2001. `http://kluedo.ub.uni-kl.de/volltexte/2001/1307/`.

[23] W. K. Shih and W.-L. Hsu. A new planarity test. *Theoretical Computer Science*, 223(1-2):179–191, 1999.