Unit Interval Graphs & Maximum c-Independent Sets Maximizing the Number of Isolated Vertices

Linda Kleist[1] **and Kai Kobbe**[1]

1 **Technische Universität Braunschweig**
`{kleist,kobbe}@ibr.cs.tu-bs.de`

Abstract

We study maximum c-independent sets that maximize the number of isolated vertices and present an algorithm that computes such subgraphs for unit interval graphs in linear time. The algorithm is based on a simple test that gives a certificate whether a specific vertex can be isolated. While the crucial property seems straight-forward, its proof requires a careful analysis of the structure of c-independent sets in unit interval graphs. Surprisingly, the techniques do not generalize to interval graphs and the algorithm does not even yield an approximation on general interval graphs.

1 Introduction

Computing maximum independent sets is a fundamental problem and appears on the list of Karp's 21 NP-complete problems. Maximum independent sets and also its generalizations of maximum c-independent sets find a variety of use cases across various fields in modelling and solving real-world problems, e.g., wireless sensor networks [3], DNA sequencing in bioinformatics [13, 19], VLSI design [13, 23], job scheduling [6, 11] and resource allocation [20], as well as identifying independent strategies in game theory [28]. For $c \in \mathbb{N}$, a *c-independent set* (c-IS) of a graph is the union of c independent sets.

While the special case of $c = 1$ is the *Maximum Independent Set Problem*, the special case of $c = 2$ is also known as *Maximum Bipartite Subgraph*, *Graph Bipartization*, or *Odd Cycle Transversal*. Not only that computing a maximum c-IS is NP-complete for general graphs, it has also been shown that there is no approximation algorithm with a factor in $O(n^{1-\varepsilon})$ for any $\varepsilon > 0$ and (possibly fixed) $c \in \mathbb{N}_{\geq 1}$, unless P = NP [18, 22]. In contrast, a maximum c-IS can be computed in polynomial time for special graph classes; including interval graphs, even if c is part of the input [29].

We consider the problem of computing a maximum c-IS for unit interval graphs with the additional property of maximizing the number of isolated vertices (among all maximum c-ISs). To this end, we say that a maximum c-IS is *max-iso*, if no other maximum c-IS has more isolated vertices. For an example, consider the unit interval graph depicted in Figure 1, where the thick intervals correspond to a maximum 2-IS with one isolated vertex. Is there a maximum 2-IS with more isolated vertices?

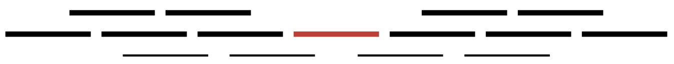

Figure 1 A unit interval graph where the subset of thick intervals is a maximum 2-IS with one isolated vertex. Note that exchanging the top four intervals with the bottom four intervals yields a maximum 2-IS with two isolated vertices, namely the first and last vertex.

Our main result is as follows.

▶ **Theorem 1.1.** *There exists an algorithm that computes a max-iso c-IS for every unit interval graph on n vertices with a running time in $O(n)$, even if c is part of the input.*

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Our key motivation is a scheduling problem where conflicts between machines appear due to shared resources or spatial proximity [7]. In this variant, each job has phases of pre- and post-processing, and no two jobs on conflicting machines may overlap in such phases. For the case that machine conflicts can be expressed by a unit interval graph, we expect that maximum c-ISs with many isolated vertices are crucial to obtain good schedules.

The rest of our paper is organized as follows. We review related work in Section 1.1, introduce fundamental concepts in Section 1.2 and discuss the algorithm in Section 2.

1.1 Related work

Finding a maximum c-IS is known to be NP-complete [15] and, for graphs on n vertices, there does not even exist an $O(n^{1-\varepsilon})$-approximation for any $c \in \mathbb{N}_{\geq 1}$ and $\varepsilon > 0$, unless P = NP [18, 22, 31]. Moreover, when considering general graphs, computing a maximum c-IS is equivalent to computing a maximum 1-IS, under polynomial-time reductions [25].

Considering $c = 1$, the maximum independent set (MIS) is relevant in many applications and therefore well studied, also for special graph classes. In this sense, constant factor approximations exist for bounded degree graphs and families of geometric intersection graphs [8]. Furthermore, polynomial-time approximation schemes (PTAS) exist for families of graphs that are closed under taking minors [16], e.g., planar graphs [5]. For many other graph classes, a maximum weight independent set may be found in polynomial time. Famous examples include claw-free graphs [12, 24], P_5-free graphs [21] and perfect graphs [17]. For chordal graphs, and thus in particular for interval graphs, maximum weight independent sets can be computed in linear time [14]. More generally, the maximum weight independent set can be computed in polynomial time for graphs that do not contain a k-prism or an induced C_n with $n \geq 5$ [10].

For $c = 2$, the problem is also known as Maximum (Induced) Bipartite Subgraph, Graph Bipartization, or Odd Cycle Transversal. A maximum 2-IS can be found in polynomial time in planar graphs with maximum degree three, while it is NP-complete for cubic graphs and planar graphs with maximum degree four [4, 9]. Additionally, polynomial-time algorithms exist for many other graph classes; including split graphs, permutation graphs, tolerance graphs and circular-arc graphs [25, 30]. We emphasize that there are graph classes, for which a maximum 1-IS can be computed in polynomial time, while finding a maximum 2-IS is NP-hard. Examples are circle graphs [2, 26, 27] and perfect graphs, as computing a maximum 2-IS is NP-hard for the subclass of clique-separable graphs [1].

For general c, it is known that the class of chordal graphs allows for a polynomial-time algorithm if c is not part of the input [29]. If c is part of the input, polynomial-time algorithms exist for the two subfamilies of perfect graphs of i-triangulated graphs [1] and interval graphs [29].

1.2 Fundamental Concepts

Let $G = (V, E)$ be a graph. As usual, we denote the *(open) neighborhood* of a vertex v by $N(v) := \{u \in V \mid uv \in E\}$ and the *closed neighborhood* by $N[v] := N(v) \cup \{v\}$. For a subset $U \subset V$, $G[U]$ denotes the subgraph induced by U.

Independent sets. A subset of vertices $U \subset V$ is an *independent set* of G, if no two vertices of U share an edge in G. For a fixed $c \in \mathbb{N}$, a *c-independent set* (c-IS) $\mathcal{I}$ of G is a union of c independent sets $\mathcal{I}_1, ..., \mathcal{I}_c$ of G; its size is given by the number of vertices, i.e., $|\mathcal{I}| := |\bigcup_i \mathcal{I}_i|$. A c-IS is *maximum* if no other c-IS has larger size. We denote the size of a maximum c-IS of

G by $\alpha_c(G)$. As mentioned before, we are particularly interested in c-ISs with many isolated vertices. To this end, we call a maximum c-IS $\mathcal{I}$ of G *max-iso*, if G has no maximum c-IS with more isolated vertices.

Interval graphs. In a (unit) interval representation of a graph $G = (V, E)$, every vertex is represented by a (unit) interval in $\mathbb{R}^1$ such that two intervals intersect if and only if the corresponding vertices share an edge. If a graph has a (unit) interval representation, then it is a (unit) interval graph. Without loss of generality, we assume that no two intervals are identical. A unit interval representation has a natural ordering of the intervals by their start points. If the interval of a vertex v starts before the interval of w, then we say v is smaller than w; we write $v \prec w$. We write $v \preceq w$ if either $v \prec w$, or $v = w$, i.e., v and w refer to the same vertex. An ordering $v_1, \ldots, v_n$ of the vertices of G is a *left-right-order* if $v_i \prec v_{i+1}$ for all i. We shortly recall a simple greedy algorithm to compute a maximum c-IS for a unit interval graph $G = (V, E)$ with left-right-order $v_1, \ldots, v_n$ with a runtime in $O(n)$ [29]: We start with $\mathcal{I} = \emptyset$ and consider the vertices by increasing index. Vertex v_i is added to $\mathcal{I}$, if this maintains a c-IS in G. A useful property of the greedy solution $\mathcal{I}$ for G is the following: For every vertex $u \in V$, it holds that $\mathcal{I}' := \{v \in \mathcal{I} \mid v \prec u\}$ is a maximum c-IS in $G[\{v \in V \mid v \prec u\}]$. Clearly, an analogous statement holds when using the reversed (right-left) order of the vertices.

2 The Algorithm

In this section, we study the computation of max-iso c-ISs for unit interval graphs. We start by establishing a simple test that states if it is reasonable to isolate a specific vertex for our purposes. A straight-forward implementation would yield a quadratic algorithm. We additionally show how to derive a linear-time algorithm.

▶ **Lemma 2.1.** *Let G be a connected unit interval graph, $\mathcal{I}_w$ a max-iso c-IS of G and w be the smallest isolated vertex in $\mathcal{I}_w$. Consider a vertex $v \prec w$. If $\alpha_c(G - N(v)) = \alpha_c(G)$, then $G - N(v)$ contains a max-iso c-IS of G.*

Proof-Sketch. We consider a maximum c-IS $\mathcal{I}_v$ in $G - N(v)$, i.e., $\mathcal{I}_v$ has size $\alpha_c(G)$. If $w \in \mathcal{I}_v$, we observe that exchanging the sets of vertices that are larger than w in $\mathcal{I}_v$ and $\mathcal{I}_w$ (denoted by $\mathcal{I}_v^{\succ}, \mathcal{I}_w^{\succ}$) by each other, yields a c-IS; here we use the fact that G is a unit interval graph. For a schematic illustration consider Figure 2. Therefore, it holds that $|\mathcal{I}_v^{\succ}| = |\mathcal{I}_w^{\succ}|$ and consequently $\mathcal{I}' := (\mathcal{I}_v \setminus \mathcal{I}_v^{\succ}) \cup \mathcal{I}_w^{\succ}$ is a maximum c-IS in G, where the isolated vertices are the same as in $\mathcal{I}_w$, except for isolating now v instead of w. Thus, $\mathcal{I}'$ is max-iso.

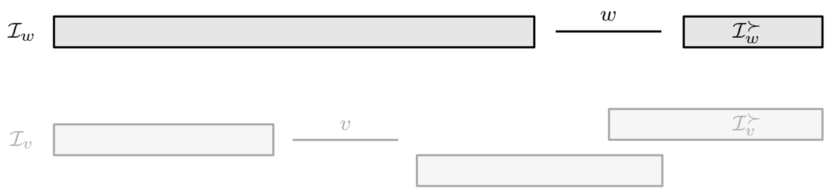

Figure 2 Illustration for the proof of Lemma 2.1. If $w \in \mathcal{I}_w, \mathcal{I}_v$, then replacing $\mathcal{I}_w^{\succ}$ and $\mathcal{I}_v^{\succ}$ by each other in $\mathcal{I}_w$ or $\mathcal{I}_v$ maintains a c-IS.

Afterwards, we show that there exists a maximum c-IS in $G - N(v)$ that contains w. In particular, this implies that $v \notin N(w)$. To this end, we carefully analyse the structure of $G[\mathcal{I}_v \cup \mathcal{I}_w]$ and, assuming that there is no such maximum c-IS, obtain a contradiction. ◀

▶ Remark. Lemma 2.1 does not hold for interval graphs in general: Consider the interval graph G depicted in Figure 3. Note that it contains several c-cliques and a unique $(2c-1)$-clique $C = \{u, b_1, \ldots, b_{c-1}, r_1, \ldots, r_{c-1}\}$. In order to obtain a maximum c-IS, we have to delete $c-1$ vertices of C. There exist various choices. However, when we keep at least one b_i and one r_j, then the resulting c-IS has no isolated vertex. Deleting the vertices $b_1, \ldots, b_{c-1}$ isolates vertex v, while deleting the vertices $r_1, \ldots, r_{c-1}$ isolates the vertices $w_1, \ldots, w_k$ and thus yields the unique max-iso c-IS of G. Consequently, as $N(v) = \{b_1, \ldots, b_{c-1}\}$, the graph $G - N(v)$ contains a maximum c-IS of G but no max-iso c-IS. This implies that the decision of isolating v is not only suboptimal but even arbitrarily bad as the number of isolated vertices is 1 vs k in the optimum. Consequently, Lemma 2.1 cannot be used to derive a constant-factor approximation algorithm for general interval graphs.

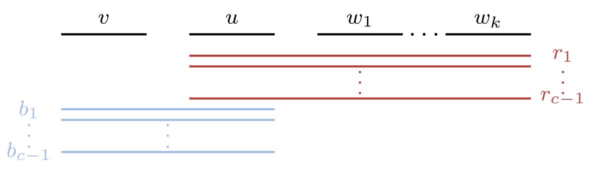

Figure 3 An interval graph G and its first vertex v such that $\alpha_c(G - N(v)) = \alpha_c(G)$, but $G - N(v)$ contains no max-iso c-IS of G. Thus, Lemma 2.1 does not generalize to interval graphs.

▶ Remark. It is crucial in Lemma 2.1 that v is a valid candidate for a smallest isolated vertex. In other words, even if $\alpha_c(G - N(v)) = \alpha_c(G)$ for some vertex v, it is not necessarily true that $G - N(v)$ contains a max-iso c-IS of G. For an example, consider the unit interval graph G depicted in Figure 4. Observe that there are two disjoint $(2c-1)$-cliques in G, namely, $C_i = \{u_i, b^i_1, \ldots, b^i_{c-1}, r^i_1, \ldots, r^i_{c-1}\}$ for $i \in \{1, 2\}$. In order to obtain a maximum c-IS, we have to delete $c-1$ vertices from C_1 and C_2, respectively. The unique maximum c-IS $\mathcal{I}$ that isolates v is obtained by deleting all r^i_j; note that this implies that there is no other isolated vertex in $\mathcal{I}$. In contrast, the maximum c-IS $\mathcal{I}'$ obtained by deleting all b^i_j has two isolated vertices, namely w_1 and w_2. Thus, when applying Lemma 2.1, it is crucial that v is a candidate for a smallest isolated vertex (in its connected component).

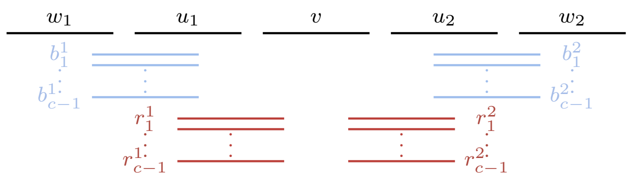

Figure 4 A unit interval graph G and vertex v with $\alpha_c(G - N(v)) = \alpha_c(G)$, but $G - N(v)$ contains no max-iso c-IS of G.

As mentioned before, Lemma 2.1 yields a simple quadratic algorithm: For a given left-right-order and increasing i, we check iteratively whether $\alpha_c(G) = \alpha_c(G - N(v_i))$. If so, we delete $N(v_i)$ from G and repeat with incremented i. In each step, we use Lemma 2.1 for the connected component of v_i. Finally, we compute a maximum c-IS of the modified graph and return it. As computing a maximum c-IS takes linear time (for a given left-right-order), we obtain a simple quadratic algorithm.

In the following, we show how to obtain a linear-time algorithm for unit interval graphs.

▶ **Lemma 2.2.** *For every unit interval graph on n vertices with given left-right-order, Algorithm 1 can be implemented to compute a max-iso c-IS with a running time in $O(n)$.*

Algorithm 1 Algorithm for computing a max-iso c-IS in unit interval graphs

```
Require: Unit interval Graph $G = (V, E)$ with left-right-order $v_1, \dots, v_n$
Ensure: max-iso $c$-IS of $G$
 1: $U := V$
 2: $L := \emptyset$
 3: $R :=$ Greedy maximum $c$-IS with reversed order $v_n, \dots, v_1$
 4: while $U \neq \emptyset$ do
 5:     Identify the smallest vertex $v$ from $U$ and delete it from $U$
 6:     if $|(L \cup R) \cap N[v]| = 1$ then          ▷ if true, $v$ will be isolated
 7:         $U = U \setminus N[v]$
 8:         $L = (L \setminus N[v]) \cup \{v\}$
 9:         $R = R \setminus N[v]$
10:     else if $|L \cap N[v]| < c$ then
11:         $L = L \cup \{v\}$
12:         Delete the smallest vertex from $R$
13:     end if
14: end while
15: return $L$
```

Proof-Sketch. The key idea of the algorithm is to maintain a leftmost partial solution L (before the current vertex) and a rightmost partial (greedy) solution R (after the current vertex) which allows for constant update time in each iteration. This is schematically depicted in Figure 5. When the algorithm ends, it holds that L is a max-iso c-IS for the given graph. More precisely, in each iteration of the while-loop, we test whether the smallest yet unconsidered vertex v (selected in line 5) can be isolated (in line 6). If so, we delete its neighborhood to guarantee that it is isolated and add v to our partial solution L (in lines 7-9). If not, we test whether it can be added greedily to L (in line 10).

Throughout the algorithm, we maintain a set of invariants. To this end, consider an iteration where v is selected in line 5 and let S denote the set of vertices for which line 6 evaluated to true so far. Before the iteration, the following invariants hold:

- $L \cup R$ is a maximum c-IS in G in which vertices from S are isolated.
- $L \subset \{u \in V \mid u \prec v\} =: V_L$ and $R \subset \{u \in V \mid v \preceq u\} =: V_R$
- $L \setminus N[v]$ is a maximum c-IS in $G[V_L - N[v]]$ (in which vertices from S are isolated); moreover, after the largest isolated vertex $u \in L$, vertices are added greedily to L.
- $R \setminus N[v]$ is a (greedy) maximum c-IS in $G[V_R - N[v]]$ (for right-left order).

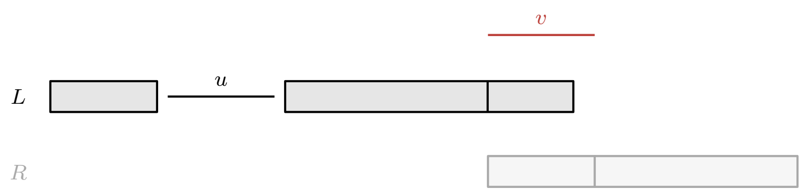

Figure 5 State of L and R before the iteration where v is selected in line 5 of Algorithm 1.

These properties allow us to check efficiently the size of a largest c-IS where the vertices $S \cup \{v\}$ are isolated, because $(L \cup R \cup \{v\}) \setminus N(v)$ is such a largest c-IS. In line 6, we test whether this size is equal to $\alpha_c(G) = |L \cup R|$ and if so, we isolate v. ◀

References

1 Louigi Addario-Berry, W.S. Kennedy, Andrew D. King, Zhentao Li, and Bruce Reed. Finding a maximum-weight induced k-partite subgraph of an i-triangulated graph. *Discrete Applied Mathematics*, 158(7):765–770, 2010. doi:10.1016/j.dam.2008.08.020.

2 Alberto Apostolico, Mikhail J. Atallah, and Susanne E. Hambrusch. New clique and independent set algorithms for circle graphs. *Discrete Applied Mathematics*, 36(1):1–24, 1992. doi:10.1016/0166-218X(92)90200-T.

3 Ozkan Arapoglu and Orhan Dagdeviren. An asynchronous self-stabilizing maximal independent set algorithm in wireless sensor networks using two-hop information. In *2019 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–5, 2019. doi:10.1109/ISNCC.2019.8909189.

4 Mourad BaÏou and Francisco Barahona. Maximum weighted induced bipartite subgraphs and acyclic subgraphs of planar cubic graphs. *SIAM Journal on Discrete Mathematics*, 30(2):1290–1301, 2016. doi:10.1137/140980053.

5 Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994. doi:10.1145/174644.174650.

6 Matthias Bentert, René van Bevern, and Rolf Niedermeier. Inductive k-independent graphs and c-colorable subgraphs in scheduling: a review. *Journal of Scheduling*, 22(1):3–20, 2018. doi:10.1007/s10951-018-0595-8.

7 Moritz Buchem, Linda Kleist, and Daniel Schmidt genannt Waldschmidt. Scheduling with machine conflicts. In *Approximation and Online Algorithms: 20th International Workshop, (WAOA 2022)*, page 36–60, 2022. doi:10.1007/978-3-031-18367-6_3.

8 Timothy M Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, pages 333–340, 2009. doi:10.1145/1542362.1542420.

9 Hyeong-Ah Choi, Kazuo Nakajima, and Chong S. Rim. Graph bipartization and via minimization. *SIAM Journal on Discrete Mathematics*, 2(1):38–47, 1989. doi:10.1137/0402004.

10 Maria Chudnovsky, Marcin Pilipczuk, Michał Pilipczuk, and Stéphan Thomassé. On the maximum weight independent set problem in graphs without induced cycles of length at least five. *SIAM Journal on Discrete Mathematics*, 34(2):1472–1483, 2020. doi:10.1137/19M1249473.

11 Duncan Eddy and Mykel J. Kochenderfer. A maximum independent set method for scheduling earth observing satellite constellations, 2020. arXiv:2008.08446.

12 Yuri Faenza, Gianpaolo Oriolo, and Gautier Stauffer. Solving the weighted stable set problem in claw-free graphs via decomposition. *Journal of the ACM (JACM)*, 61(4):1–41, 2014. doi:10.1145/2629600.

13 Pierre Fouilhoux and A. Ridha Mahjoub. Solving vlsi design and dna sequencing problems using bipartization of graphs. *Computational Optimization and Applications*, 51(2):749–781, 2012. doi:10.1007/s10589-010-9355-1.

14 András Frank. Some polynomial algorithms for certain graphs and hypergraphs. In *Proceedings of the 5th British combinatorial conference, congressus numerantium*, volume XV, pages 211—-226, 1975.

15 M. R. Garey and D. S. Johnson. "Strong" NP-completeness results: Motivation, examples, and implications. *J. ACM*, 25(3):499–508, 1978. doi:10.1145/322077.322090.

16 Martin Grohe. Local tree-width, excluded minors, and approximation algorithms. *Combinatorica*, 23(4):613–632, 2003. doi:10.1007/s00493-003-0037-9.

17 M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981. doi:10.1007/BF02579273.

18 Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182(1):105–142, 1999. doi:10.1007/BF02392825.

19 Deborah Joseph, Joao Meidanis, and Prasoon Tiwari. Determining dna sequence similarity using maximum independent set algorithms for interval graphs. In *Algorithm Theory — SWAT '92*, pages 326–337, 1992. doi:10.1007/3-540-55706-7_29.

20 Alper Köse and Berna Özbek. Resource allocation for underlaying device-to-device communications using maximal independent sets and knapsack algorithm. In *IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–5, 2018. doi:10.1109/PIMRC.2018.8580784.

21 Daniel Lokshantov, Martin Vatshelle, and Yngve Villanger. Independent set in p_5-free graphs in polynomial time. In *Symposium on discrete algorithms (SODA)*, pages 570–581, 2014. doi:10.1137/1.9781611973402.4.

22 Carsten Lund and Mihalis Yannakakis. The approximation of maximum subgraph problems. In *Automata, Languages and Programming*, pages 40–51, 1993. doi:10.1007/3-540-56939-1_60.

23 M. Marek-Sadowska. An unconstrained topological via minimization problem for two-layer routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 3(3):184–190, 1984. doi:10.1109/TCAD.1984.1270074.

24 George J. Minty. On maximal independent sets of vertices in claw-free graphs. *Journal of Combinatorial Theory, Series B*, 28(3):284–304, 1980. doi:10.1016/0095-8956(80)90074-X.

25 G. Narasimhan. *The Maximum k-Colorable Subgraph Problem.* PhD thesis, University of Wisconsin-Madison, 1989. URL: https://research.cs.wisc.edu/techreports/1989/TR864.pdf.

26 Nicholas Nash and David Gregg. An output sensitive algorithm for computing a maximum independent set of a circle graph. *Information Processing Letters*, 110(16):630–634, 2010. doi:10.1016/j.ipl.2010.05.016.

27 M. Sarrafzadeh and D.T. Lee. A new approach to topological via minimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 8(8):890–900, 1989. doi:10.1109/43.31548.

28 Christof Spanring. Axiom of Choice, Maximal Independent Sets, Argumentation and Dialogue Games. In *2014 Imperial College Computing Student Workshop*, volume 43 of *Open Access Series in Informatics (OASIcs)*, pages 91–98, 2014. doi:10.4230/OASIcs.ICCSW.2014.91.

29 Mihalis Yannakakis and Fanica Gavril. The maximum k-colorable subgraph problem for chordal graphs. *Information Processing Letters*, 24(2):133–137, 1987. doi:10.1016/0020-0190(87)90107-4.

30 Susan S. Yeh and Andrea S. LaPaugh. Algorithms for finding a maximum bipartite subgraph for special classes of graphs. 1988. technical report. URL: https://www.cs.princeton.edu/research/techreps/TR-149-88.

31 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 681–690, 2006. doi:10.1145/1132516.1132612.