On (in)tractability of connection and cut problems[*]

Alexsander A. de Melo[1]
Orientadores: Celina M. H. de Figueiredo[1]**, Uéverton S. Souza**[2]**, Ana Silva**[3]

[1]Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, RJ, Brasil

[2]Universidade Federal Fluminense (UFF)
Niterói, RJ, Brasil

[3]Univesidade Federal do Ceará (UFC)
Fortaleza, CE, Brasil

{aamelo,celina}@cos.ufrj.br, ueverton@ic.uff.br, anasilva@mat.ufc.br

Abstract. *This work addresses connection and cut problems from the viewpoint of graph classes and computational complexity, classic and parameterized. Regarding connection problems, we investigate the so-called* TERMINAL CONNECTION *problem (TCP), which can be seen as a generalisation of the classical* STEINER TREE *problem. We propose several complexity results for TCP, when restricted to specific graph classes, and some of its input parameters are fixed. As for cut problems, we analyse the complexity of the classical* MAXCUT *problem. We introduce the first complexity classification for the problem on interval graphs of bounded interval count. In addition, we prove the* NP*-completeness of* MAXCUT *on permutation graphs, settling a question posed by David S. Johnson in the* Ongoing Guide to NP-completeness*, which has been open since 1985. Finally, we consider the problem of computing the zig-zag number of a directed graph, which is a directed width measure defined over cuts of a graph.*

1. Introduction

Connection and cut problems on graphs have been widely studied over the years, mainly due to the fact that such problems are closely related to a great variety of real-world applications, playing an important role in the field of network design. Generally, connection problems aim to obtain a minimum/maximum number of required elements whose inclusion yields a connected graph satisfying certain conditions, while cut problems aim to obtain a minimum/maximum number of required elements whose removal yields a (disconnected) graph with more connected components. Both settings are ways of measuring the robustness of the network.

As an example of connection problem, consider the scenario in which a group of special nodes (*e.g.* wireless devices) in a communication network needs to broadcast messages among themselves, but some of such nodes are physically unreachable. Then, in order to establish a viable communication route, additional nodes of the network (*e.g.* repeaters) might be required, which in turn implies an extra associated cost. Hence, the

[*]The present document summarises the contributions of the PhD thesis entitled *On (in)tractability of connection and cut problems* [de Melo 2022]. This work was partially supported by CNPq (GD 140399/2017-8) and CAPES (Finance Code 0001; PDSE 88881.187636/2018-01).

objective is to obtain a feasible configuration that contains all of the special nodes and that, at the same time, minimises the number of additional nodes used. On the other hand, concerning cut problems, consider as an example the problem of designing fault tolerant networks, where the objective is to have a fully operable system even after the failure of some of its components. Informally, the robustness of a system can be evaluated by measuring the connectivity between any two subgroups of nodes in the corresponding network. In other words, it can be described as a function of the maximum number of distinct existing links between any two subgroups of nodes of the network.

In this work, we investigate the computational complexity, classic and parameterized, of connection and cut problems. We solve many open questions posed by the community; in particular, we settle the computational complexity of a long-standing problem asked by David S. Johnson in the *Ongoing Guide to NP-completeness* [Johnson 1985]. In what follows, we discuss our contributions in more details.

One of the most fundamental and well-known connection problems is STEINER TREE, which was proved to be NP-hard by Karp in his seminal paper [Karp 1972]. Since then, STEINER TREE has been extensively studied from the point of view of graph classes and computational complexity, being one of the eleven problems selected by David S. Johnson to appear in the *Ongoing Guide to NP-completeness* [Johnson 1985]. As a variant of STEINER TREE, and having as motivation applications in information security, network routing and telecommunications, the TERMINAL CONNECTION problem (TCP) was introduced recently [Dourado et al. 2014]. However, TCP was also proved to be NP-complete on general graphs [Dourado et al. 2014]. We then investigate the complexity of the problem when restricted to specific graph classes and some of its input parameters are fixed. We propose several polynomial-time algorithms and hardness proofs. In Section 2, we formally define the STEINER TREE and TERMINAL CONNECTION problems, and we summarise our main contributions, focusing particularly on results that separate the complexity of TCP from the complexity of STEINER TREE. These results were presented at the *47th International Conference on Current Trends in Theory and Practice of Computer Science* [de Melo et al. 2021a], and they were published as a full paper in *RAIRO - Theoretical Informatics and Applications* [de Melo et al. 2023]. Further obtained results for TCP and related variants were also published as full papers in *Journal of Computer and System Sciences* [de Melo et al. 2020] and in *Networks* [de Melo et al. 2021b], and as an extended abstract in *Matemática Contemporânea* [de Melo et al. 2022].

As for cut problems, MAXCUT is one of the most classical graph theory problems, and it is known to be NP-complete since the seventies [Garey et al. 1976]. Nonetheless, only recently its restriction to interval graphs has been announced to be hard [Adhikary et al. 2021]. Yet, the complexity on the even more restricted class of unit interval graphs, *i.e.* interval graphs of interval count 1, still remains unknown. In fact, many flawed proofs of polynomial-time solvability for the problem on the class of unit interval graphs have been presented [Bodlaender et al. 1999, Boyaci et al. 2017], just to be disproved closely after [Bodlaender et al. 2004, Kratochvíl et al. 2020]. We present the first complexity result for MAXCUT on interval graphs of bounded interval count, by proving that the problem remains NP-complete on interval graphs of interval count 4. Our contribution is a relevant improvement towards filling the complexity gap between interval and unit interval graphs. As an additional result, we prove that MAX-

CUT is also NP-complete on permutation graphs, which, along with interval graphs, is one of the most important and extensively studied geometric intersection graph classes. This result also closes a long-standing question appearing in the *Ongoing Guide to NP-completeness* [Johnson 1985]. In Section 3, we formally define the MAXCUT problem and give a brief idea of our NP-complete proofs. Our results for the problem on interval graphs were presented at the *46th International Symposium on Mathematical Foundations of Computer Science-MFCS 2021* [de Figueiredo et al. 2021] and have been recently published as a full paper in *Discrete & Computational Geometry* [de Figueiredo et al. 2023b]. Concerning permutation graphs, our results have been recently published as a full paper in *Journal of Graph Theory* [de Figueiredo et al. 2023a].

Structural parameters have been crucial in the development of parameterized complexity theory. Many problems that are hard on general graphs become tractable when parameterized by such parameters [Courcelle 1990]. Nevertheless, one of their limitations is the fact that, when dealing with directed graphs, the direction of edges are not taken into account. Then, Johnson, Robertson, Seymour and Thomas initiated a quest for the development of width measures that explicitly consider the direction of edges [Johnson et al. 2001]. This motivated the development of several width measures for directed graphs, such as directed path-width [Barát 2006] and directed tree-width [Johnson et al. 2001]. Moreover, defined over cuts of directed graphs, the notion of zig-zag number was introduced in [de Oliveira Oliveira 2013] as a generalization of directed path-width and an attempt to provide a unified algorithmic framework based on monadic second-order logic. However, very little was known about the complexity of computing the zig-zag number of a directed graph, with many questions remaining open. We present the first results concerning these questions. More specifically, we analyse the complexity of the k-ZIG-ZAG NUMBER problem, which consists in deciding whether the input directed graph has zig-zag number at most k, for a fixed $k \geq 0$. We prove that k-ZIG-ZAG NUMBER is in NP, and that 2-ZIG-ZAG NUMBER is already an NP-hard problem. In Section 4, we define the notion of zig-zag number of directed graphs and present our contributions regarding the complexity of k-ZIG-ZAG NUMBER. These results were published as a full paper in *Discrete and Applied Mathematics* [Dourado et al. 2022].

2. Connection problems

Let G be a graph and W be a non-empty subset of the vertex set of G. A *connection tree* T of G for W is a tree subgraph of G that contains every vertex in W, and whose leaves belong to W. In a connection tree T for W, the vertices belonging to W are called *terminal*, and the vertices belonging to $V(T) \setminus W$ are called *non-terminal* and are classified into two types according to their respective degrees in T, namely: the non-terminal vertices of degree exactly 2 in T are called *linkers* and the non-terminal vertices of degree at least 3 in T are called *routers*. This defines a partition of the vertex set of a connection tree into terminal vertices, linkers and routers. Figure 1 illustrates the notions of connection tree, linkers and routers.

The STEINER TREE problem has as input a connected graph G, a non-empty subset W of $V(G)$, and a non-negative integer k, and it asks whether there exists a connected subgraph T of G that contains every vertex in W and satisfies $|V(T) \setminus W| \leq k$. Note that, if such a connected subgraph exists, then it admits a spanning tree with at most k vertices not belonging to W. Thus, STEINER TREE can be alternatively defined in terms

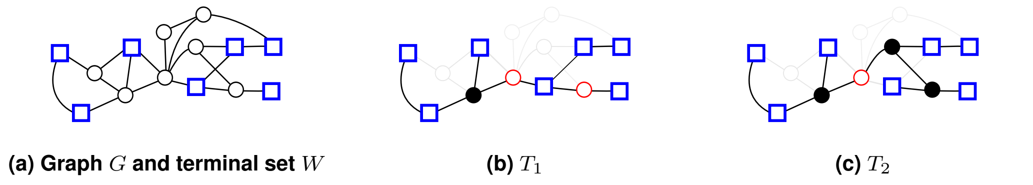

(a) Graph G and terminal set W (b) T_1 (c) T_2

Figure 1. A graph G, a terminal set W (blue squares), and connection trees of G for W, each with a distinct number of linkers (red circles) and routers (solid black circles).

of connection tree, by asking whether the input graph G has a connection tree T for W with at most k non-terminal vertices. However, STEINER TREE does not distinguish the non-terminal terminal vertices among themselves. This motivates the definition of the TERMINAL CONNECTION problem (TCP, for short), which has as input a connected graph G, a non-empty terminal set $W \subseteq V(G)$, and two non-negative integers ℓ and r, and it asks whether there exists a connection tree T of G for W that has at most ℓ linkers and at most r routers. TCP was proved to be polynomial-time solvable (more specifically, to be in XP) when the parameters ℓ and r are both fixed, whereas it is NP-complete even if exactly one of the parameters $\ell \geq 0$ or $r \geq 0$ is fixed [Dourado et al. 2014].

An important observation about TCP and STEINER TREE is the fact that there is a Turing reduction from STEINER TREE to TCP that preserves the structure of the input graph, namely: (G, W, k) is a yes-instance of STEINER TREE if and only if (G, W, ℓ, r) is a yes-instance of TCP for some pair $\ell, r \in \{0, \ldots, k\}$ whose sum is exactly k. Hence, if TCP (with ℓ and r arbitrarily large) is polynomial-time solvable on some graph class $\mathcal{G}$, then so is STEINER TREE. Nevertheless, if either $\ell \geq 0$ or $r \geq 0$ is fixed, then possibly TCP is polynomial-time solvable on $\mathcal{G}$, while STEINER TREE (for arbitrarily large k) remains NP-complete on $\mathcal{G}$. Moreover, there might exist a graph class $\mathcal{G}$ on which STEINER TREE is polynomial-time solvable whereas TCP is NP-complete.

We confirm the existence of such separating classes. First, we prove that, on *split* graphs, TCP is in XP when parameterized by r, given that $r \geq 1$. This tells us that TCP is polynomial-time solvable for fixed $r \geq 1$, whereas STEINER TREE is known to be NP-complete [White et al. 1985]. In a nutshell, given a graph G on n vertices, we establish structural properties for the sought tree. Then, relying on these, our algorithm for TCP enumerates in time $n^{\mathcal{O}(r)}$ each possible candidate for router set R, such that $|R| \leq r$ and $R \subseteq V(G) \setminus W$. Finally, we apply matching techniques to decide whether G admits a suitable connection tree for W whose router set is exactly R and that has at most ℓ linkers. We also show that the described algorithm is asymptotically optimum, unless widely believed complexity assumptions fail, by giving a parameterized reduction from SET COVER, a well-known W[2]-hard problem. As a second separating result, we prove through a polynomial-time reduction from HAMILTONIAN PATH that TCP also remains NP-complete on RDV graphs even if $r \geq 0$ is fixed. This contrasts with the computational complexity of STEINER TREE, which is known to be polynomial-time solvable on *strongly chordal* graphs [White et al. 1985], a superclass of RDV graphs. Moreover, we show that the clique-width of the graph constructed in this reduction from HAMILTONIAN PATH is at most one unit greater than the clique-width of the original graph. Consequently, building on this same reduction and on the fact that HAMILTONIAN PATH parameterized by clique-width is W[1]-hard, we obtain that TCP parameterized by clique-width is also

W[1]-hard, whereas STEINER TREE parameterized by clique-width is known to be in FPT [Bergougnoux and Kanté 2019].

On the other hand, agreeing with the computational complexity of STEINER TREE [Colbourn and Stewart 1990], we prove that TCP is linear-time solvable on *cographs*. This is done by providing a dynamic programming algorithm over the nodes of the cotree of the input cograph. In addition, also agreeing with the complexity of STEINER TREE [Müller and Brandstädt 1987], we prove through a polynomial-time reduction from VERTEX COVER that TCP is NP-complete on *chordal bipartite graphs*.

The NP-completeness proof for TCP on chordal bipartite graphs were published in [de Melo et al. 2022], and all the other mentioned results were published in [de Melo et al. 2020, de Melo et al. 2021a, de Melo et al. 2023]. In particular, in [de Melo et al. 2020], we analyse the complexity of S-TCP, the strict variant of TCP. This variant is also used as an auxiliary problem to solve TCP on split graphs.

3. Maximum cut

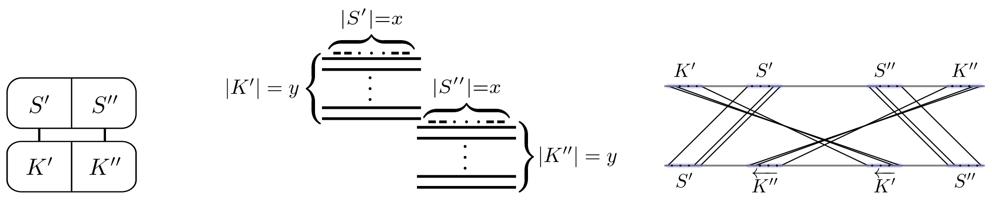
A *cut* of a graph G is a partition $[A, B]$ of its vertex set $V(G)$ into two non-empty disjoint parts $A, B \subseteq V(G)$, and the *cut-set* of G associated with a cut $[A, B]$ is the set of edges of G with an endpoint in A and the other endpoint in B. The MAXCUT problem is the decision problem that has as input a graph G and a non-negative integer k, and which asks whether there exists a cut in G whose cut-set has cardinality at least k. As previously mentioned, MAXCUT is a classical NP-complete problem, studied over the years on distinct graph classes. Nevertheless, its complexity on *interval graphs* has only been settled recently [Adhikary et al. 2021], and on the more restricted class of *unit interval graphs*, *i.e.* interval graphs of *interval count* 1, it still remains open. Moreover, since the *Ongoing Guide to NP-completeness* [Johnson 1985], it has been unknown whether the problem is NP-complete when constrained to *permutation graphs*.

The notion of *interval count* of an interval graph was introduced in the eighties *cf.* [Leibowitz et al. 1982], and it is defined as the minimum number of distinct interval lengths needed to represent the graph among all its interval models. Besides being an interesting problem by itself, understanding the interval count can be of value for the investigation of problems that are hard on general interval graphs.

We provide the first complexity classification for MAXCUT on interval graphs of bounded interval count, showing that it is also NP-complete on interval graphs of interval count 4. This was presented at *MFCS 2021* [de Figueiredo et al. 2021] and has been recently published in *Discrete & Computational Geometry* [de Figueiredo et al. 2023b]. Additionally, we prove the NP-completeness of MAXCUT on permutation graphs, a result that has been recently published in *Journal of Graph Theory* [de Figueiredo et al. 2023a]. In order to prove these results, we propose polynomial-time reductions from MAXCUT restricted to *cubic graphs*. The key gadget of our reduction is the notion of (x, y)-*grained gadget*, which we introduce as a generalization of the so-called V-gadgets and E-gadgets described in [Adhikary et al. 2021].

An (x, y)-*grained gadget* is a split graph H formed by a stable set $S' \cup S''$ and a clique $K' \cup K''$, with $|S'| = |S''| = x$ and $|K'| = |K''| = y$, such that K' (resp. K'') is complete to S' (resp. S''), and there is no edge between K' (resp. K'') and S'' (resp. S').

Grained gadgets are both interval and permutation graphs. Indeed, Figure 2 depicts an (x, y)-grained gadget and its interval model and permutation representation. The central property of such a gadget is the fact that, for suitable values of x and y, if G is a supergraph of an (x, y)-grained gadget whose neighbours are arranged in a well-structured way, then, in any maximum cut of G, the vertices in $K' \cup S''$ are placed in a same part of the cut, opposite to the part containing the vertices in $K'' \cup S'$. This provides a handy way of establishing how maximum cuts of the original graph are related to maximum cuts of our reduction graphs. Then, relying on this property, we use grained gadgets as building blocks in our polynomial-time reductions, which are briefly described next.



(a) Grained gadget **(b) Interval model** **(c) Permutation representation**

Figure 2. An (x, y)-grained gadget and its interval model and permutation representation, respectively.

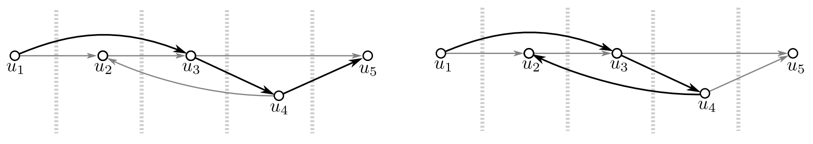
Given a cubic graph G and orderings $\pi_V = (v_1, \ldots, v_n)$ and $\pi_E = (e_1, \ldots, e_m)$ of the vertex set and edge set of G, respectively, we construct an interval model $\mathcal{M}$ of interval count 4, such that G has a cut-set of size at least k if and only if the interval graph corresponding to $\mathcal{M}$ has a cut-set of size at least $\phi(n, k)$, for a suitable function ϕ. Intuitively, to construct $\mathcal{M}$, we partition the real line into m mutually disjoint regions, such that the j-th region is related to the edge e_j and holds the information whether e_j is in the cut-set of G. To accomplish this, the edge e_j is represented by a grained gadget $\mathcal{E}_j$, which must be contained within the j-th region, and each vertex v_i is represented by a grained gadget $\mathcal{H}_i^j$ and special intervals, called *link intervals*, connecting $\mathcal{H}_i^j$ to $\mathcal{H}_i^{j+1}$. The aim of such link intervals is to propagate, from a region to the next, the information about to which part of the cut the respective vertex v_i belongs. In addition, in order to represent that a vertex v_i is an endpoint of an edge e_j, we add in a convenient way a couple of intervals connecting $\mathcal{H}_i^j$ to $\mathcal{E}_j$, called *incidence intervals*, We show that it is possible to construct such a model $\mathcal{M}$ having only four distinct interval lengths. Hence, we obtain NP-completeness of MAXCUT on interval graphs of interval count 4.

Our proof for the NP-completeness of MAXCUT on permutation graphs employs a similar approach. More specifically, given G, π_V and π_E as in the previous paragraph, we construct a permutation model $\{\Pi, \Pi'\}$, whose associated permutation graph G' has a cut-set of size at least $\varphi(n, k)$ if and only if G has a cut-set of size at least k, where φ is a suitable function. In the permutation model $\{\Pi, \Pi'\}$, each vertex v_i of G is represented by a grained gadget $\mathcal{H}_i$ and each edge e_j of G is represented by a grained gadget $\mathcal{E}_j$, such that $\mathcal{H}_i$ appears on the left of $\mathcal{H}_{i+1}$, $\mathcal{H}_n$ appears on the left of $\mathcal{E}_1$, and $\mathcal{E}_j$ appears on the left of $\mathcal{E}_{j+1}$. In addition, in order to represent that a vertex v_i is an endpoint of an edge e_j, we add a couple of vertices relating $\mathcal{H}_i$ to $\mathcal{E}_j$, so that e_j belongs to a maximum cut of G if and only if the grained gadgets of its endpoints are partitioned in an opposite way in the corresponding maximum cut of G'. It immediately follows from our construction that the obtained graph is a permutation graph.

4. Zig-zag number

The notion of *zig-zag number* was introduced in [de Oliveira Oliveira 2013] as a generalisation for directed path-width, aiming to provide a unified algorithmic framework for directed graphs based on monadic second-order logic. However, the problem of computing the zig-zag number of a directed graph was completely unexplored. We then investigate the computational complexity of k-ZIG-ZAG NUMBER, the problem of deciding whether a given directed graph has zig-zag number at most k, for fixed $k \geq 0$. Intuitively, the zig-zag number of a directed graph measures how much the directed cycles of this graph are nested. Next, we present a formal definition for this notion.

Let G be a directed graph on n vertices and $\pi = (u_1, \dots, u_n)$ be an ordering of the vertex set of G. For each $i \in [n-1]$, we let $S_G(\pi, i)$ denote the *i-th cut-set of G with respect to π*, *i.e.* the set of all edges of G between the vertices in $\{u_1, \dots, u_i\}$ and the vertices in $\{u_{i+1}, \dots, u_n\}$. For each directed path P of G, we let $\mathsf{zn}(G, \pi, P)$ be the maximum number of edges of P that belong to the cut-set $S_G(\pi, i)$, where the maximum is taken over all $i \in [n-1]$. Then, we let $\mathsf{zn}(G, \pi)$ be the maximum $\mathsf{zn}(G, \pi, P)$ over all directed paths P of G. Finally, we define the *zig-zag number of G*, denoted by $\mathsf{zn}(G)$, as the minimum $\mathsf{zn}(G, \pi)$ over all orderings π. Figure 3 exemplifies these notions.



(a) $\mathsf{zn}(G, \pi, P_1) = 1$ **(b)** $\mathsf{zn}(G, \pi, P_2) = 2$

Figure 3. Directed graph G, ordering $\pi = (u_1, \dots, u_n)$, and directed paths P_1 and P_2 (in bold), such that $\mathsf{zn}(G, \pi, P_1) = 1$ and $\mathsf{zn}(G, \pi, P_2) = 2$, respectively.

It is immediate from the definition of zig-zag number that a directed graph has zig-zag number 0 if and only if it does not contain any edge. Moreover, one can verify that every directed acyclic graph with at least one edge has zig-zag number 1. Indeed, it is known that a directed graph G is directed acyclic if and only if it admits a *topological ordering*. Thus, one can verify that, if G is a directed acylic graph and π corresponds to a topological ordering of G, then $\mathsf{zn}(G, \pi) = 1$. In other words, graphs of zig-zag number at least 2 must contain directed cycles. On the other hand, every directed graph G with a directed cycle of length at least 3 necessarily has zig-zag number at least 2. In this case, for each ordering $\pi = (u_1, \dots, u_n)$, there always exist three distinct vertices $a, b, c \in V(G)$ such that (a, b, c) is a directed path of G, where $a <_\pi b$ and $c <_\pi b$. Next, we present a sketch of our proofs about the computational complexity of k-ZIG-ZAG NUMBER.

First, we prove that k-ZIG-ZAG NUMBER is in NP for each fixed k. Unlike most natural decision problems, settling k-ZIG-ZAG NUMBER in NP turned out to be an interesting quest. This is due to the fact that the definition of zig-zag number involves the alternation of an existential and a universal quantifiers, and thus, a naive application of the definition only leads to a Σ_2^{P}-upper bound for the problem. To circumvent this and, then, settle the problem in NP, we show how to replace the inner universal quantifier, which iterates over all directed paths, with an XP-time deterministic computation corresponding to a guessed linear order of the vertices of the input graph and the integer k.

More specifically, we reduce the problem of deciding whether $\mathsf{zn}(G, \pi) \geq k + 1$, for a guessed ordering π, to the REACHABILITY problem in a suitably defined directed acyclic graph $D_G(\pi, k)$. This graph is constructed considering all possible sequences $S'_1, \ldots, S'_{n-1}$ of *subcuts* $S'_i \subseteq S_G(\pi, i)$ of size at most $k+1$, such that their union induces a directed path P. The neat idea behind such sequences is that, when enumerating all possible subcuts S'_i, it is sufficient to only consider neighbouring subcuts S'_{i+1} and S'_{i+1}. This owns to the fact that a non-trivial directed graph P is a directed path if and only if it satisfies the following simple four conditions: (i) P has exactly one vertex of in-degree 0 and out-degree 1; (ii) P has exactly one vertex of in-degree 1 and out-degree 0; (iii) all the other vertices of P have in-degree 1 and out-degree 1; (iv) P is weakly connected. Then, using a depth-first search algorithm, we are able to decide in $n^{\mathcal{O}(k)}$-time whether the graph $D_G(\pi, k)$ has a directed path of size $n-1$, which in turn is equivalent to deciding whether the input directed graph G has zig-zag number at least $k+1$. Consequently, we obtain that k-ZIG-ZAG NUMBER is in NP for every fixed $k \geq 0$.

Now, through a polynomial-time reduction from POSITIVE NOT ALL EQUAL 3SAT (PNAE 3SAT, for short), we prove that 2-ZIG-ZAG NUMBER is NP-hard. It is worth mentioning that, for $k > 2$, it is still unknown whether k-ZIG-ZAG NUMBER is NP-hard. Given an instance $I = (X, \mathcal{C})$ of PNAE 3SAT, we construct a directed graph G_I comprising a directed cycle $H_i = (u_i^1, u_i^2, u_i^3)$ for each variable $x_i \in X$, and a directed cycle $\widetilde{H}_j = (v_j^1, v_j^2, v_j^3)$ for each clause $C_j \in \mathcal{C}$. Also, if x_i is the l-th literal in C_j, then G_I further contains the directed edges (u_i^1, v_j^l) and (u_i^3, v_j^l). We establish the existence of a satisfying truth assignment for I if and only if there is an ordering of zig-zag number at most 2 for the vertices of G_I. The main idea of our proof is to explore the internal relative orderings of the vertices of each directed cycle of G_I, and the relative placements among the subgraphs H_{l_1}, H_{l_2}, H_{l_3}, and $\widetilde{H}_j$ for each clause $C_j = \{x_{l_1}, x_{l_2}, x_{l_3}\} \in \mathcal{C}$.

The results presented in this section have been published in *Discrete Applied Mathematics* [Dourado et al. 2022].

5. Concluding remarks

We have investigated connection and cut problems on graphs from the viewpoint of computational complexity and graph classes, proposing various contributions for these two groups of problems. In particular, we have answered a long-standing open question for the MAXCUT problem, appearing in the *Ongoing Guide to NP-completeness* [Johnson 1985]. In addition to the described contributions, many other results were obtained during the doctoral studies, including the investigation of width measures for directed graphs from formal languages and logic standpoints [de Melo and de Oliveira Oliveira 2022]. The obtained results were published in high-standard journals and conferences in the fields of algorithms and graph theory, counting nine journal papers and six conference papers. In the thesis' appendix, we also provide a thorough revision of the *Ongoing Guide to NP-completeness* [de Figueiredo et al. 2022]. In that revision, in addition to an updated version of the original guide [Johnson 1985], we also provide the state of the art of parameterized complexity of the problems, thus putting in evidence the granularity provided by the Parameterized Complexity Theory[1].

[1]See Parameterized Complexity News, March 2022, Vol.18, No. 1, ISSN 2203-109X.

References

Adhikary, R., Bose, K., Mukherjee, S., and Roy, B. (2021). Complexity of maximum cut on interval graphs. In *37th International Symposium on Computational Geometry, SoCG 2021*, volume 189 of *LIPIcs*, pages 7:1–7:11.

Barát, J. (2006). Directed path-width and monotonicity in digraph searching. *Graphs and Combinatorics*, 22(2):161–172.

Bergougnoux, B. and Kanté, M. (2019). Fast exact algorithms for some connectivity problems parameterized by clique-width. *Theoretical Computer Science*, 782:30 – 53.

Bodlaender, H. L., de Figueiredo, C. M. H., Gutierrez, M., Kloks, T., and Niedermeier, R. (2004). SIMPLE MAX-CUT for split-indifference graphs and graphs with few P_4s. In *Experimental and Efficient Algorithms, Third International Workshop, WEA 2004*, volume 3059 of *Lecture Notes in Computer Science*, pages 87–99.

Bodlaender, H. L., Kloks, T., and Niedermeier, R. (1999). SIMPLE MAX-CUT for unit interval graphs and graphs with few P_4s. *Electronic Notes in Discrete Mathematics*, 3:19–26.

Boyaci, A., Ekim, T., and Shalom, M. (2017). A polynomial-time algorithm for the maximum cardinality cut problem in proper interval graphs. *Information Processing Letters*, 121:29–33.

Colbourn, C. J. and Stewart, L. K. (1990). Permutation graphs: connected domination and Steiner trees. *Discrete Mathematics*, 86(1-3):179–189.

Courcelle, B. (1990). The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75.

de Figueiredo, C. M. H., de Melo, A. A., de S. Oliveira, F., and Silva, A. (2021). Maximum cut on interval graphs of interval count four is NP-complete. In *46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021*, volume 202 of *LIPIcs*, pages 38:1–38:15.

de Figueiredo, C. M. H., de Melo, A. A., de S. Oliveira, F., and Silva, A. (2023a). Maxcut on permutation graphs is NP-complete. *Journal of Graph Theory*. doi.org/10.1002/jgt.22948 (forthcoming).

de Figueiredo, C. M. H., de Melo, A. A., de S. Oliveira, F., and Silva, A. (2023b). Maximum cut on interval graphs of interval count four is NP-complete. *Discrete & Computational Geometry*. doi.org/10.1007/s00454-023-00508-x (forthcoming).

de Figueiredo, C. M. H., de Melo, A. A., Sasaki, D., and Silva, A. (2022). Revising Johnson's table for the 21st century. *Discrete Applied Mathematics*, 323:184–200.

de Melo, A. A. (2022). *On (in)tractability of connection and cut problems*. PhD thesis, Universidade Federal do Rio de Janeiro.

de Melo, A. A., de Figueiredo, C. M. H., and Souza, U. S. (2021a). On the terminal connection problem. In *47th International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2021*, volume 12607 of *Lecture Notes in Computer Science*, pages 278–292.

de Melo, A. A., de Figueiredo, C. M. H., and Souza, U. S. (2021b). On undirected two-commodity integral flow, disjoint paths and strict terminal connection problems. *Networks*, 77(4):559–571.

de Melo, A. A., de Figueiredo, C. M. H., and Souza, U. S. (2022). The strict terminal connection problem on chordal bipartite graphs. *Matemática Contemporânea*, 48(14):137–145.

de Melo, A. A., de Figueiredo, C. M. H., and Souza, U. S. (2023). On the computational difficulty of the terminal connection problem. *RAIRO - Theoretical Informatics and Applications*.

de Melo, A. A. and de Oliveira Oliveira, M. (2022). Second-order finite automata. *Theory of Computing Systems*, 66(4):861–909.

de Melo, A. A., Figueiredo, C. M. H., and Souza, U. S. (2020). A multivariate analysis of the strict terminal connection problem. *Journal of Computer and System Sciences*, 111:22–41.

de Oliveira Oliveira, M. (2013). Subgraphs satisfying MSO properties on z-topologically orderable digraphs. In *Parameterized and Exact Computation, IPEC 2013*, volume 8246 of *Lecture Notes in Computer Science*, pages 123–136.

Dourado, M. C., de Figueiredo, C. M. H., de Melo, A. A., de Oliveira Oliveira, M., and Souza, U. S. (2022). Computing the zig-zag number of directed graphs. *Discrete Applied Mathematics*, 312:86–105.

Dourado, M. C., Oliveira, R. A., Protti, F., and Souza, U. S. (2014). Design of connection networks with bounded number of non-terminal vertices. *Matemática Contemporânea*, 42(14):39–47.

Garey, M. R., Johnson, D. S., and Stockmeyer, L. J. (1976). Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267.

Johnson, D. S. (1985). The NP-completeness column: An ongoing guide. *Journal of Algorithms*, 6(3):434–451.

Johnson, T., Robertson, N., Seymour, P. D., and Thomas, R. (2001). Directed tree-width. *Journal of Combinatorial Theory, Series B*, 82(1):138–154.

Karp, R. M. (1972). *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA.

Kratochvíl, J., Masarík, T., and Novotná, J. (2020). U-bubble model for mixed unit interval graphs and its applications: The maxcut problem revisited. In *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020*, volume 170 of *LIPIcs*, pages 57:1–57:14.

Leibowitz, R., Assmann, S. F., and Peck, G. W. (1982). The interval count of a graph. *SIAM Journal on Algebraic Discrete Methods*, 3(4):485–494.

Müller, H. and Brandstädt, A. (1987). The NP-completeness of Steiner tree and dominating set for chordal bipartite graphs. *Theoretical Computer Science*, 53(2-3):257–265.

White, K., Farber, M., and Pulleyblank, W. (1985). Steiner trees, connected domination and strongly chordal graphs. *Networks*, 15(1):109–124.