Some Complexity Considerations on the Uniqueness of Graph Colouring

[1]OLIVIER HUDRY, [2]ANTOINE LOBSTEIN
[1]Institut polytechnique de Paris, T´el´ecom Paris, 91123 Palaiseau
[2]Laboratoire Interdisciplinaire des Sciences du Num´erique
(UMR 9015), CNRS, Universit´e Paris-Saclay, 91400 Orsay
FRANCE

Abstract: For some well-known $\mathcal{NP}$-complete problems, linked to the colourability of a graph, we study the variation which consists in asking about the *uniqueness* of a solution (up to permutations of the colours). In particular, we show that the decision problems Unique k-Colouring (U-k-COL) with $k \geqslant 3$ and Unique Colouring (U-COL), have equivalent complexities, up to polynomials, as Unique Satisfiability (U-SAT) and Unique One-in-Three Satisfiability (U-1-3-SAT) by establishing polynomial reductions relating these four problems. As a consequence, all are co-$\mathcal{NP}$-hard (or, equivalently, $\mathcal{NP}$-hard with respect to Turing reductions) and belong to the complexity class $\mathcal{DP}$. We also consider the problem Unique Optimal Colouring (U-OCOL) and show that it belongs to $\mathcal{L}^{\mathcal{NP}}$ (also denoted Θ_2).

Key-Words: Complexity Theory, co-$\mathcal{NP}$-hardness, $\mathcal{NP}$-hardness, Decision Problems, Polynomial Reduction, Turing Reduction, Uniqueness of Solution, Graph Theory, Graph Colouring, Partition into Independent Sets, Boolean Satisfiability, Satisfiability Problems.

Received: November 14, 2022. Revised: May 13, 2023. Accepted: June 7, 2023. Published: July 5, 2023.

1 Introduction

1.1 Goal, Outline and Results

In the theory of complexity, decision problems are stated with a question that admits only the answer YES or NO; the question can be stated in the very general following form: "Given an instance I and a property Pr on I, is Pr true for I?" where Pr can be expressed as: "Is there a structure satisfying a given characteristic?". In our study, we add the extra word "unique" to the latter question: "Is there a *unique* structure satisfying a given characteristic?" Then we investigate the complexity of these newly defined problems. In this paper, we pay attention to the uniqueness of the vertex-colouration of graphs (up to colour permutations). For this, we consider the uniqueness for some variations of the well-known problem SAT.

In Section 1.2, we give some notation and definitions from graph theory and then, in Section 1.3, the basic background for the complexity theory; in Section 1.4, we present some well-known Satisfiability problems, together with their complexities. In Section 2, we study how these complexities vary if we consider the question of the uniqueness of a solution for colouring problems. We prove that the colouring problems called below U-k-COL ($k \geqslant 3$) and U-COL have equivalent complexities as U-SAT and U-1-3-SAT; consequently, they all belong to the class $\mathcal{DP}$ and are co-$\mathcal{NP}$-hard (or, equivalently, $\mathcal{NP}$-hard with respect to Turing reductions). We also show that U-OCOL belongs to the class $\mathcal{L}^{\mathcal{NP}}$. We present some concluding remarks in Section 3.

We similarly revisited some famous problems, from the viewpoint of uniqueness of solution: Vertex Cover and Dominating Set (as well as its generalization to domination within distance r) [17], Hamiltonian Cycle [18], and r-Identifying Code together with r-Locating-Dominating Code [16].

Note that uniqueness of solutions, which may be seen as part of the wider issue of the number of solutions of a problem, had been studied earlier in a few papers (see, e.g., [3], [4], [5], [9], [12], [13], [20], [21], [22], [27]). In particular, the complexity of the uniqueness for SAT problems has been studied in [20], which provides a "dichotomy theorem" characterizing the variants of SAT for which the uniqueness is polynomial.

1.2 Notation and Definitions for Graphs

For graph theory, we refer to, e.g., [2] or [8]. We shall denote by $G = (V, E)$ a finite, simple, undirected graph with vertex set V and edge set E, where an *edge* between $x \in V$ and $y \in V$ is indifferently denoted by xy or yx. The *order* of the graph is its number of vertices.

An *independent set*, or *stable set*, is a subset $V^* \subseteq V$ such that for all $u \in V^*$, $v \in V^*$, uv is not an edge of G: $uv \notin E$. If k is an integer, $k \geqslant 1$, a *k-colouring* of G is a function $f : V \to \{1, 2, \ldots, k\}$ such that $f(u) \neq f(v)$ whenever $uv \in E$. Two vertices with the same value on the function f are said to share the same colour. Obviously, a k-colouring of G exists if and only if one can partition V into k independent sets, with a correspondence between an independent set and a set of vertices sharing the

same colour. When it exists, such a partition of V will be called a *k-IS-partition*. Since fewer than k colours may be needed, we might be led to use the word "partition" in a sense broader than usual, with empty sets allowed. The smallest k such that there is a k-IS-partition is called the *chromatic number* $\chi(G)$ of G. A colouring with a number of colours equal to $\chi(G)$ is said to be *optimal*.

1.3 Notation and Definitions for Complexity

We expound here the notions of complexity that will be needed in the sequel. We refer the reader to, e.g., [1], [10], [19] or [23] for more on this topic. Here, we consider complexity only with respect to computing time and not with respect to memory space.

A *decision problem* is of the type "Given an instance I and a property Pr on I, is Pr true for I?", and has only two solutions, YES or NO. The complexity class $\mathcal{P}$ denotes the set of problems which can be solved by a *polynomial* (time) algorithm, and the complexity class $\mathcal{NP}$ the set of problems which can be solved by a *nondeterministic polynomial* algorithm.

A *polynomial reduction* from a decision problem π_1 to a decision problem π_2 is a polynomial algorithm that maps any instance of π_1 into an equivalent instance of π_2, that is, an instance of π_2 admitting the same answer as the instance of π_1; in this case, we write $\pi_1 \leqslant_m^p \pi_2$, or simply $\pi_1 \leqslant \pi_2$. Cook [6] proved that there is a problem in $\mathcal{NP}$, namely "Satisfiability" or simply SAT (see below in Section 1.4), to which every other problem in $\mathcal{NP}$ can be polynomially reduced. Thus, in a sense, SAT is a "hardest" problem inside $\mathcal{NP}$. Other problems share this property in $\mathcal{NP}$ and are called *$\mathcal{NP}$-complete* problems; their class is denoted by $\mathcal{NP}$-complete or $\mathcal{NP}$-$\mathcal{C}$.

The way to show that a decision problem π is $\mathcal{NP}$-complete is, once it is proved to be in $\mathcal{NP}$, to choose some $\mathcal{NP}$-complete problem and to polynomially reduce it to π. From a practical viewpoint, the $\mathcal{NP}$-completeness of a problem π implies that we do not know any polynomial algorithm solving π, and that, under the assumption $\mathcal{P} \neq \mathcal{NP}$, which is widely believed to be true, no such algorithm exists: the time required can grow exponentially with the size of the instance (for example, when the instance is a graph, its size is polynomially linked to the order of the graph; see below, Section 2).

The *complement* of a decision problem, "Given I and Pr, is Pr true for I?", is "Given I and Pr, is Pr false for I?". The class *co-$\mathcal{NP}$* (respectively, *co-$\mathcal{NP}$-complete* or *co-$\mathcal{NP}$-$\mathcal{C}$*) is the class of the problems which are the complement of a problem in $\mathcal{NP}$ (respectively, in $\mathcal{NP}$-$\mathcal{C}$).

For problems which are not necessarily decision problems, a *Turing reduction* from a problem π_1 to a problem π_2 is an algorithm $\mathcal{A}$ that solves π_1 using a (hypothetical) subprogram $\mathcal{S}$ solving π_2 such that, if $\mathcal{S}$ were a polynomial algorithm for π_2, then $\mathcal{A}$ would be a polynomial algorithm for π_1; in this case, we write $\pi_1 \leqslant_T \pi_2$. Thus, in this sense, π_2 is "at least as hard" as π_1. A problem π is *$\mathcal{NP}$-hard* (respectively, *co-$\mathcal{NP}$-hard*) if there is a Turing reduction from some $\mathcal{NP}$-complete (respectively, co-$\mathcal{NP}$-complete) problem to π [10, p. 113].

Remark 1 *Note that with these definitions, $\mathcal{NP}$-hard and co-$\mathcal{NP}$-hard coincide [10, p. 114].*

The notions of completeness and hardness can of course be extended to classes other than $\mathcal{NP}$ or co-$\mathcal{NP}$. Note that $\mathcal{NP}$-hardness and co-$\mathcal{NP}$-hardness are defined differently in [7] and [15] for instance: there, a problem π is *$\mathcal{NP}$-hard* (respectively, *co-$\mathcal{NP}$-hard*) if there is a *polynomial* reduction from some $\mathcal{NP}$-complete (respectively, co-$\mathcal{NP}$-complete) problem to π; then, with these definitions, $\mathcal{NP}$-hardness and co-$\mathcal{NP}$-hardness do not coincide any more and this may lead to confusion (see Section 3). For the problems studied here, our results show that U-k-COL ($k \geqslant 3$) and U-COL are co-$\mathcal{NP}$-hard for both polynomial and Turing reductions, while they are $\mathcal{NP}$-hard only for Turing reductions.

We will use two more classes. They received different notations, or were shown to be the same as classes defined differently, hence the multiple notations (see, e.g., [14]).

• The class $\mathcal{L}^{\mathcal{NP}}$ [19] contains the decision problems which can be solved by applying, with a number of calls which is logarithmic with respect to the size of the instance, a subprogram able to solve an appropriate problem in $\mathcal{NP}$ (usually, an $\mathcal{NP}$-complete problem). It is also denoted by $\mathcal{P}^{\mathcal{NP}[\mathrm{O}(\log n)]}$, or $\mathcal{P}^{\mathcal{NP}[\log]}$, or Θ_2, or Θ_2^P, or still $\mathcal{P}^{||\mathcal{NP}}$; in the sequel, we will use the notation $\mathcal{L}^{\mathcal{NP}}$, easier to remember. This class should not be confused with the class $\mathcal{L}$, which is the class of decision problems solvable by a Turing machine restricted to use an amount of memory logarithmic in the size of the input.

• Finally, let us define $\mathcal{DP}$ [25] (or $\mathcal{DIF}^{\mathcal{P}}$ [3] or $\mathcal{BH}_2$ [19], [28], ...) as the class of languages (or problems) L such that there are two languages $L_1 \in \mathcal{NP}$ and $L_2 \in$ co-$\mathcal{NP}$ satisfying $L = L_1 \cap L_2$. This class is not to be confused with $\mathcal{NP} \cap$ co-$\mathcal{NP}$ (see the warning in, e.g., [23, p. 412]); actually, $\mathcal{DP}$ contains $\mathcal{NP} \cup$ co-$\mathcal{NP}$ and is contained in $\mathcal{L}^{\mathcal{NP}}$ (see Figure 1).

Membership to $\mathcal{P}$, $\mathcal{NP}$, co-$\mathcal{NP}$, $\mathcal{DP}$, or $\mathcal{L}^{\mathcal{NP}}$ gives an upper bound on the complexity of a problem (this problem is not more difficult than ...), whereas a hardness result gives a lower bound (this problem

is at least as difficult as ...). Still, such results are conditional in some sense; if for example $\mathcal{P} = \mathcal{NP}$, they would lose their interest.

1.4 Satisfiability Problems

We first recall some problems related to Boolean Satisfiability, and present what is known about their complexities.

We consider a set $\mathcal{X}$ of n *Boolean variables* x_i ($1 \leqslant i \leqslant n$). A *literal* is a variable x or its complement (or negated variable) $\overline{x}$; let $\overline{\mathcal{X}}$ denote the set of the complement variables of $\mathcal{X}$: $\overline{\mathcal{X}} = \{\overline{x} \text{ with } x \in \mathcal{X}\}$. A *clause* defined on $\mathcal{X}$ is a subset of $\mathcal{X} \cup \overline{\mathcal{X}}$. We consider a set $\mathcal{C}$ of m *clauses* c_j ($1 \leqslant j \leqslant m$); $\mathcal{C}$ is also called a *Boolean formula*. Each clause c_j contains κ_j *literals*. Equivalently, $\mathcal{C}$ can read as a logical formula:

$$\mathcal{C} = \wedge_{1 \leqslant j \leqslant m} c_j, \text{ with } c_j = \vee_{1 \leqslant i \leqslant \kappa_j} \ell_i^j,$$

with $\ell_i^j \in \mathcal{X} \cup \overline{\mathcal{X}}$ for $1 \leqslant j \leqslant m$ and $1 \leqslant i \leqslant \kappa_j$.

This form is called a *normal conjunctive form*. A *truth assignment* for $\mathcal{X}$ sets the variable x_i to TRUE, also denoted by T, and its complement to FALSE (or F), or *vice-versa.* A truth assignment is said to *satisfy* the clause c_j if c_j contains at least one true literal, and to satisfy the set of clauses $\mathcal{C}$ if every clause contains at least one true literal. The following decision problems, for which the size of the instance is polynomially linked to $n + m$, are classical problems in complexity.

Problem SAT (Satisfiability):
Instance: A set $\mathcal{X}$ of variables, a collection $\mathcal{C}$ of clauses over $\mathcal{X}$.
Question: Is there a truth assignment for $\mathcal{X}$ that satisfies $\mathcal{C}$?

Problem 1-3-SAT (One-in-Three Satisfiability):
Instance: A set $\mathcal{X}$ of variables, a collection $\mathcal{C}$ of clauses over $\mathcal{X}$, each clause containing exactly three different literals.
Question: Is there a truth assignment for $\mathcal{X}$ such that each clause of $\mathcal{C}$ contains *exactly one* true literal?

We shall say that a clause (respectively, a set of clauses) is 1-3-*satisfied* by an assignment if this clause (respectively, every clause in the set) contains exactly one true literal.

Problem $\overline{\text{SAT}}$ (co-Satisfiability):
Instance: A set $\mathcal{X}$ of variables, a collection $\mathcal{C}$ of clauses over $\mathcal{X}$.
Question: Is it true that no truth assignment for $\mathcal{X}$ satisfies $\mathcal{C}$?

The problem SAT is one of the basic and most well-known $\mathcal{NP}$-complete problems [6], [10, p. 39 and p. 259]. It follows that the problem $\overline{\text{SAT}}$ is co-$\mathcal{NP}$-complete.

The problem 1-3-SAT is also $\mathcal{NP}$-complete [26, Lemma 3.5], [10, p. 259].

The problems U-SAT and U-1-3-SAT are obtained from the above problems SAT and 1-3-SAT respectively by adding the word "unique" in the question. The problem U-SAT has been studied as far back as 1982 ([3], [24]). The problems U-SAT and U-1-3-SAT share the same complexity. More precisely, we have:

$$\text{U-SAT} \leqslant \text{U-1-3-SAT} \leqslant \text{U-SAT}. \qquad (1)$$

They are both $\mathcal{NP}$-hard and both belong to $\mathcal{DP}$.

Remark 2 *In [23], it is stated that "U-SAT is not believed to be $\mathcal{DP}$–complete". It is shown in [3] that there exists one oracle under which U-SAT is not $\mathcal{DP}$–complete; and one oracle under which it is, if $\mathcal{NP}$ and co-$\mathcal{NP}$ are distinct.*

2 Uniqueness for Colouring Problems

We now turn to the colouring decision problems, stated in their two usual forms. First, for a given integer $k \geqslant 1$:

Problem k-COL (k-Colouring):
Instance: A graph G.
Question: Does G admit a k-colouring?

It is well known that this problem is trivial for $k = 1$, polynomial for $k = 2$ and $\mathcal{NP}$-complete for $k \geqslant 3$, see [11], [10, p. 191]. We can also state the problem COL with k belonging to the instance:

Problem COL (Colouring):
Instance: A graph G, an integer k.
Question: Does G admit a k-colouring?

The problem COL is also $\mathcal{NP}$-complete.

In their spirit, the problems U-k-COL and U-COL are derived from the above problems k-COL and COL respectively in a similar manner as previously done for satisfiability problems.

Anyway, if we define the problem of the existence of a unique k-colouring in a graph by simply adding the word "unique" in the statements of the problems k-COL or COL, it is quite obvious that the answer will always be NO for $k > 1$, since any permutation (other than the identity) on the values of a k-colouring provides another k-colouring. So, rather than saying that we look for a unique colouring up to permutations, it is more relevant to consider the problems stated in terms of IS-partitions. Also, if the graph is not connected, the answer will be NO for $k > 1$. Thus, the problems that we study assume the graphs to be connected, as stated below:

Let k be a fixed integer with $k \geqslant 1$.
Problem U-k-COL (Unique k-IS-Partition):
Instance: A *connected* graph G.
Question: Does G admit a *unique* k-IS-partition?

Problem U-COL (Unique IS-Partition):
Instance: A *connected* graph G, an integer k.
Question: Does G admit a *unique* k-IS-partition?

We add the following problem, which would be irrelevant without the word "unique" in its question, and which will be dealt with in Section 2.3:

Problem U-OCOL (Unique Optimal IS-Partition):
Instance: A *connected* graph $G = (V, E)$.
Question: Is there a unique way of partitioning V into a minimum number of independent sets?

Note that, like 1-COL and 2-COL, U-1-COL and U-2-COL are, quite obviously, polynomial.

As the graphs are assumed to be connected, we may assume also, without loss of generality, that they are encoded by their adjacency matrices. Thus, the size of a graph G with n vertices is equal to n^2.

2.1 Preliminary Results on 3-Colourings

The following lemma is somehow inspired by the proof of the $\mathcal{NP}$-completeness of 3-COL, see Theorem 2.1 and Figure 1 in [11]. Note that the proof from [11] cannot convey uniqueness, even up to permutations of colours.

Lemma 3 *Consider the graph $G_0 = (V_0, E_0)$ described by Figure 2.*

(a) Any 3-IS-partition of $\{a, b, d\}$ such that these three vertices belong to the same independent set, cannot be extended to a 3-IS-partition in G_0.

(b) Any 3-IS-partition of $\{a, b, d\}$ such that these three vertices belong to exactly two independent sets, can be uniquely extended to a 3-IS-partition in G_0.

Proof. Without loss of generality, we assume that in any 3-IS-partition of V_0 into S_1, S_2, S_3, the vertex v_2 belongs to S_3. Then a, b and d can belong only to S_1 or S_2.

(a) First, we assume that the vertices a, b and d belong to S_1. Then, because of the triangle a, w_1, v_2, we have $w_1 \in S_2$. Similarly, $z_2 \in S_2$, $z_3 \in S_2$. Then $z_1 \in S_3$, and step by step, $w_2 \in S_1$, $w_3 \in S_3$, $v_1 \in S_2$, $w_4 \in S_1$, $y_6 \in S_2$, $y_2 \in S_3$, $y_1 \in S_2$, $y_4 \in S_1$, and $y_5 \in S_3$. But now the neighbours of y_3 are $y_6 \in S_2$, $y_5 \in S_3$ and $d \in S_1$, which makes it impossible to have a 3-IS-partition. Obviously, the conclusion is the same if a, b and d belong to S_2, with the roles of S_1 and S_2 permuted.

(b) Assume first that $a \in S_1$, $b \in S_1$, $d \in S_2$. Then $\{w_1, z_2\} \subset S_2$, $z_3 \in S_1$, $z_1 \in S_3$, $w_2 \in S_1$, $v_1 \in S_2$, $w_3 \in S_3$, $w_4 \in S_1$, $y_6 \in S_2$, $y_2 \in S_3$, $y_1 \in S_2$, $y_4 \in S_1$, $y_5 \in S_3$ and $y_3 \in S_1$, which constitutes a 3-IS-partition, obtained in a unique way. The same is true for $a \in S_2$, $b \in S_2$, $d \in S_1$, with the roles of S_1 and S_2 permuted.

For $a \in S_1$, $b \in S_2$, $d \in S_1$, we simply give the three sets S_1, S_2 and S_3, since it is straightforward to check that there is only one way to obtain them: $S_1 = \{a, d, z_2, w_3, w_4, y_5, y_2\}$, $S_2 = \{b, z_1, z_3, w_1, v_1, y_6, y_4\}$, $S_3 = \{v_2, w_2, y_3, y_1\}$. The case with $a \in S_2$, $b \in S_1$, $d \in S_2$ is the same, with the roles of S_1 and S_2 permuted.

For $a \in S_2$, $b \in S_1$, $d \in S_1$, the partition $S_1 = \{b, d, w_1, w_3, z_1, w_4, y_5, y_1\}$, $S_2 = \{a, z_2, z_3, v_1, y_6, y_4\}$, $S_3 = \{v_2, w_2, y_3, y_2\}$ is the only 3-IS-partition. The case with $a \in S_1$, $b \in S_2$, $d \in S_2$ is the same, with the roles of S_1 and S_2 permuted. $\triangle$

Remark 4 *We can see from the previous proof that w_4, which is linked to v_1 and v_2, belongs to S_1 as soon as two of the three vertices a, b, d belong to S_1 — and $w_4 \in S_2$ if two of a, b, d belong to S_2. This implies that $v_1 \in S_2$ in the first case, $v_1 \in S_1$ in the latter case.*

We are now ready to show that U-SAT and U-COL have equivalent complexities (up to polynomials).

2.2 Equivalence of Uniqueness of Colouring and of Satisfiability

We are going to prove the following polynomial reductions:

- U-1-3-SAT $\leqslant$ U-3-COL (Theorem 5),
- for $k \geqslant 3$, U-3-COL $\leqslant$ U-k-COL $\leqslant$ U-COL (Proposition 6),
- and U-COL $\leqslant$ U-SAT (Theorem 7).

Theorem 5 *There exists a polynomial reduction from U-1-3-SAT to U-3-COL: U-1-3-SAT $\leqslant$ U-3-COL.*

Proof. Consider an instance of U-1-3-SAT consisting in a set $\mathcal{C}$ of m clauses over n variables x_i ($1 \leqslant i \leqslant n$). We associate to it an instance of U-3-COL, i.e. a graph G, as follows. The vertex set W of G is defined by:

$$\begin{aligned} W = &\{x_i, \overline{x}_i : 1 \leqslant i \leqslant n\} \cup \{v_1, v_2\} \\ &\cup \{z_{i,j} : 1 \leqslant i \leqslant 3, 1 \leqslant j \leqslant m\} \\ &\cup \{w_{i,j} : 1 \leqslant i \leqslant 4, 1 \leqslant j \leqslant m\} \\ &\cup \{y_{i,j} : 1 \leqslant i \leqslant 6, 1 \leqslant j \leqslant m\}. \end{aligned}$$

The order of G is $2n + 2 + 13m$. For every j in $\{1, \ldots, m\}$, we denote by V_j^- the set

$$\begin{aligned} V_j^- = &\{z_{i,j} : 1 \leqslant i \leqslant 3\} \\ &\cup \{w_{i,j} : 1 \leqslant i \leqslant 4\} \\ &\cup \{y_{i,j} : 1 \leqslant i \leqslant 6\}, \end{aligned}$$

and by V_j the set

$$V_j = V_j^- \cup \{v_{1,j}, v_{2,j}\}.$$

For each clause $c_j = \{a_j, b_j, d_j\}$ $(1 \leqslant j \leqslant m)$ where a_j, b_j and d_j belong to $\{x_i, \overline{x}_i : 1 \leqslant i \leqslant n\}$, we take a copy $G_j = (V_j \cup \{a_j, b_j, d_j\}, E_j)$ of the graph $G_0 = (V_0, E_0)$ from the previous lemma, with $a_j = a$, $b_j = b$, $d_j = d$, and complete identification of the other vertices of V_0 to the vertices of V_j. Then we merge all the vertices $v_{1,j}$, $1 \leqslant j \leqslant m$, into one vertex v_1, i.e., the new vertex v_1 replaces the $v_{1,j}$'s and is linked to all their neighbours; we proceed similarly to create a new vertex v_2. The vertices v_1 and v_2 are now common for all the clauses c_j and subgraphs G_j — we still call these subgraphs G_j, with a slight notational abuse.

We add the edges $x_i\overline{x}_i$, x_iv_2 and $\overline{x}_iv_2$, $1 \leqslant i \leqslant n$, and we have our graph G; see Figure 3 for a small example.

Let us now show that the answer is the same for the initial instance of U-1-3-SAT and the instance of U-3-COL.

(1) Let us first assume that the answer to U-1-3-SAT is YES: there is a unique truth assignment 1-3-satisfying the clauses of $\mathcal{C}$. We construct a valid 3-IS-partition $W = S_1 \cup S_2 \cup S_3$ in the following way: we put v_2 in S_3, we put in S_2 the literals that have the assignment TRUE, and in S_1 those which are FALSE. Since each clause contains exactly one true literal, we are in condition (b) of Lemma 3, and *from now on*, there is a *unique* way for obtaining a 3-IS-partition of W, by proceeding in each graph G_j as in the proof of the lemma for V_0: note that we cannot proceed independently in each graph G_j, which could lead to more than one partition, because of the vertices v_1 and v_2 which are shared by all the graphs G_j.

Can we have another 3-IS-partition $W = S_1^* \cup S_2^* \cup S_3^*$? Still assuming, without loss of generality, that $v_2 \in S_3^*$, this 3-IS-partition, like every 3-IS-partition in G, induces, because of the triangles $x_i\overline{x}_iv_2$, a valid truth assignment $\mathcal{A}$ for the variables x_i, $1 \leqslant i \leqslant n$, by setting $\mathcal{A}(x_i) = \text{T}$ if $x_i \in S_2^*$ and $\mathcal{A}(x_i) = \text{F}$ if $x_i \in S_1^*$. If we study, in a subgraph G_{j_0}, the vertices a_{j_0}, b_{j_0} and d_{j_0}, we know, by Lemma 3(a), that they cannot all belong to the same set, S_1^* or S_2^*. If two of them belong to S_1^*, then, by Remark 4, $w_{4,j_0} \in S_1^*$ and $v_1 \in S_2^*$. This in turn implies that *all* the vertices $w_{4,j}$, $1 \leqslant j \leqslant m$, belong to S_1^*, and then that, for *all* j, two among the three vertices a_j, b_j and d_j belong to S_1^*. Similarly, if two of a_{j_0}, b_{j_0} and d_{j_0} belong to S_2^*, then for *all* j, two of a_j, b_j and d_j belong to S_2^*. Therefore, the assignment $\mathcal{A}$ is such that (a) in *all* the clauses, two of the three vertices a_j, b_j, d_j belong to S_1^*, or (b) in *all* the clauses, two of the three vertices a_j, b_j, d_j belong to S_2^*. This proves that the assignment $\mathcal{A}$, or its complement, 1-3-satisfies all the clauses. Since such an assignment was assumed to be unique, the 3-IS-partition $W = S_1^* \cup S_2^* \cup S_3^*$ is the same as $W = S_1 \cup S_2 \cup S_3$.

Thus a YES answer to U-1-3-SAT implies a YES answer to U-3-COL.

(2) Assume next that the answer to U-1-3-SAT is NO: this may be either because no truth assignment 1-3-satisfies the instance, or because at least two assignments do; in the latter case however, this would lead to at least two 3-IS-partitions, and a NO answer to U-3-COL. So we are left with the case when the set of clauses $\mathcal{C}$ cannot be 1-3-satisfied. But if a 3-IS-partition of W exists, then we have seen, with $S_1^* \cup S_2^* \cup S_3^*$ above, how to construct an assignment which 1-3-satisfies all the clauses; therefore, no 3-IS-partition can exist.

We have proved that, in all cases, the answer to U-3-COL is also NO.

Thus the answer is the same for the two instances.

Moreover, the complexity of the reduction is directly linked to the size of G. Since the order of G is $2n + 13m + 2$, the size of G is in $\text{O}((n+m)^2)$ while the size of the instance of U-1-3-SAT is at least $n + m$. So the reduction is polynomial.

In conclusion, we have what we wanted:

$$\text{U-1-3-SAT} \leqslant \text{U-3-COL}. \qquad \triangle$$

Reducing U-3-COL into U-k-COL for $k \geqslant 4$ and then U-k-COL for $k \geqslant 3$ to U-COL is much easier, as shown by Proposition 6:

Proposition 6 *For every integer $k \geqslant 3$, there exists a polynomial reduction from U-3-COL to U-k-COL: U-3-COL $\leqslant$ U-k-COL, and from U-k-COL to U-COL: U-k-COL $\leqslant$ U-COL.*

Proof. To go from U-ℓ-COL to U-$(\ell+1)$-COL for any integer ℓ, the trick is standard: a graph G admits a (unique) ℓ-IS-partition if and only if the graph obtained from G by adding, in polynomial time, one vertex connected to all the vertices of G, admits a (unique) $(\ell+1)$-IS-partition. Starting from $\ell = 3$, we can reach any fixed $k \geqslant 3$.

The problem U-COL, where k, the number of colours, is not fixed but is part of the instance, is at least as hard as U-k-COL, for any fixed integer k: to any instance G of U-k-COL, we can associate the instance consisting of G and k for U-COL, also in polynomial time. $\triangle$

When dealing with the usual problems like COL and SAT, it is not necessary to establish the reduction from COL to SAT explicitly: the belonging of COL to $\mathcal{NP}$ and the $\mathcal{NP}$-completeness of SAT are sufficient

to involve that such a reduction does exist. In other words, because SAT is known to be $\mathcal{NP}$-complete, the properties COL $\in \mathcal{NP}$ and SAT $\leqslant$ COL are sufficient to show that SAT and COL belong to the same complexity class, namely $\mathcal{NP}$-$\mathcal{C}$. For U-COL and U-SAT, it is not sufficient to reduce U-SAT to U-COL to be able to conclude that these two problems have the same complexity: we may only conclude from such a reduction that U-COL is at least as difficult as U-SAT. So, in order to have the reciprocal statement, we now reduce U-COL into U-SAT polynomially.

Theorem 7 *There exists a polynomial reduction from U-COL to U-SAT: U-COL $\leqslant$ U-SAT.*

Proof. We start from an instance of U-COL, ie. a connected graph $G = (V, E)$ and an integer k, with $V = \{x^1, \ldots, x^{|V|}\}$ (where $|V|$ denotes the order of G); we assume that $k \geqslant 3$ and $|V| \geqslant 3$. We create the set of variables $\mathcal{X} = \{x_i^h : 1 \leqslant h \leqslant |V|, 1 \leqslant i \leqslant k\}$ and the following clauses:

(0) $\{x_1^1\}$;

(i) for $1 \leqslant h \leqslant |V|$, clauses of size k: $\{x_1^h, x_2^h, \ldots, x_k^h\}$;

(ii) for $1 \leqslant h \leqslant |V|$ and $1 \leqslant i < j \leqslant k$, clauses of size two: $\{\overline{x}_i^h, \overline{x}_j^h\}$;

(iii) for every edge $x^h x^{h'} \in E$, k clauses of size two: $\{\overline{x}_1^h, \overline{x}_1^{h'}\}$, $\{\overline{x}_2^h, \overline{x}_2^{h'}\}$, ..., $\{\overline{x}_{k-1}^h, \overline{x}_{k-1}^{h'}\}$, $\{\overline{x}_k^h, \overline{x}_k^{h'}\}$;

(iv) for $2 \leqslant h \leqslant |V|$ and $1 \leqslant i \leqslant k$, clauses

We shall say that $\{\overline{x}_i^h\}$ is the first part of c_i^h, $\{x_i^1, \ldots, x_i^{h-1}\}$ its second part, and $\{x_{i-1}^1, \ldots, x_{i-1}^{h-1}\}$ its third part. When $i = 1$, c_i^h reduces to its first and second parts. All these clauses form the instance of U-SAT.

The role of the variables and of the clauses will appear below in the proof.

Note that the number of variables and clauses is polynomial with respect to the order of G, in particular because we may take $k \leqslant |V|$. As the complexity of the reduction is polynomially related to this number of variables and clauses, the reduction is polynomial.

Let us show now that the answer is kept by the reduction.

(1) We assume that there is a k-IS-partition of V into k sets $S_1, S_2, \ldots, S_k$. If necessary, we redefine this partition, with renamed sets S_1^*, S_2^*, ..., S_k^*, in the following way: we put x^1 in S_1^*; then if x^2 was in the same set as x^1, we also put x^2 in S_1^*, otherwise we put it in S_2^*; more generally, if x^p was in the same set as some x^q, $q < p$, we put x^p in the same set as x^q, otherwise we put it in S_t^*, where t is the smallest index that has not been used yet for a set S^*; and so on, until we have processed all the vertices. In other words, we re-order the sets $S_1, S_2, \ldots, S_k$ according to the order of the smallest superscript of their elements. In particular, x^1 belongs to the first set.

Example. $k = 5$, $|V| = 14$, $S_1 = \{x^5, x^2, x^8\}$, $S_2 = \{x^3, x^{14}, x^4\}$, $S_3 = \{x^{10}, x^1, x^{12}\}$, $S_4 = \{x^7, x^9, x^{11}\}$, $S_5 = \{x^{13}, x^6\}$. *After re-ordering as described above, we have* $S_1^* = \{x^1, x^{10}, x^{12}\}$, $S_2^* = \{x^2, x^5, x^8\}$, $S_3^* = \{x^3, x^4, x^{14}\}$, $S_4^* = \{x^6, x^{13}\}$, $S_5^* = \{x^7, x^9, x^{11}\}$.

If we have a k-IS-partition $S_1, S_2, \ldots, S_k$ ordered as above, we can define a truth assignment $\mathcal{A}_1$ by setting, for every variable x_i^h, $\mathcal{A}_1(x_i^h) = \mathrm{T}$ if and only if $x^h \in S_i$, and this assignment satisfies all the clauses; indeed:

(0) $\{x_1^1\}$ is satisfied thanks to the re-ordering;

(i) each clause $\{x_1^h, x_2^h, \ldots, x_k^h\}$ contains at least one true literal, the contrary meaning that the vertex x^h belongs to no set of the k-IS-partition;

(ii) each clause $\{\overline{x}_i^h, \overline{x}_j^h\}$ contains at least one true literal, the contrary meaning that the vertex x^h belongs to two sets S_i and S_j;

(iii) each clause $\{\overline{x}_i^h, \overline{x}_i^{h'}\}$ contains at least one true literal, the contrary meaning that two neighbours in G belong to the same set S_i;

(iv) let us consider c_i^h for given h and i, $2 \leqslant h \leqslant |V|$, $1 \leqslant i \leqslant k$, and assume that it is *not* satisfied by $\mathcal{A}_1$. Then the first part of c_i^h implies that $x^h \in S_i$, the second part that $\{x^1, \ldots, x^{h-1}\} \cap S_i = \emptyset$; if $i = 1$, this is impossible, since $x^1 \in S_1$. So $i > 1$ and c_i^h has a third part, which, when not satisfied, implies that $\{x^1, \ldots, x^{h-1}\} \cap S_{i-1} = \emptyset$. But then x^h should have been put in S_{i-1} (or possibly even earlier) when re-ordering the k-IS-partition, and we have a contradiction. Therefore, all the clauses c_i^h are satisfied by $\mathcal{A}_1$. We can conclude that any k-IS-partition gives a truth assignment satisfying all the clauses constructed for U-SAT.

Assume now that this k-IS-partition is unique, i.e., we have a YES answer for U-COL. We claim that there is only one assignment satisfying the instance of U-SAT. Assume on the contrary that another assignment, $\mathcal{A}_2$, also satisfies it. Then, thanks to the clauses described in (i) and (ii), for every h, at least one literal x_i^h is set TRUE by $\mathcal{A}_2$, and for every pair $\{i, j\}$, $i \neq j$, at least one of $\overline{x}_i^h$ or $\overline{x}_j^h$ is set TRUE, which means that at most one x_i^h is set TRUE: so for every h, *exactly* one x_i^h is TRUE. Using this, let us construct a partition $S_1^+, \ldots, S_k^+$ using the following rule: $x^h \in S_i^+$ if and only if $\mathcal{A}_2(x_i^h) = \mathrm{T}$.

Now because of the clauses $\{\overline{x}_i^h, \overline{x}_i^{h'}\}$ corresponding to neighbours x^h and $x^{h'}$ in G, at least one of x_i^h or $x_i^{h'}$ is set FALSE by $\mathcal{A}_2$: this means that two neighbours cannot be in the same set and guarantees that the partition $S_1^+, \ldots, S_k^+$ is a k-IS-partition of V, and, by assumption, it must coincide with S_1, S_2, $\ldots$, S_k, *up to permutations of the subscripts*. This is where the clauses introduced in Step (iv) intervene: without them, a single k-IS-partition could give more than one assignment, differing only according to the permutations on the subscripts of the sets of the k-IS-partition.

We are going to prove that $S_1 = S_1^+$, $S_2 = S_2^+$, $\ldots$, $S_k = S_k^+$. Because of the clause $\{x_1^1\}$, we have $\mathcal{A}_2(x_1^1) =$ T, implying that $x^1 \in S_1^+$ and $S_1 = S_1^+$. Now assume that there are two sets S_p^+ and S_q^+ such that $1 < p$, $q = p + 1$, and the element with smallest superscript in S_p^+, x^{p_1}, has superscript *greater* than the element with smallest superscript in S_q^+, x^{q_1}: $p_1 > q_1$. Consider the clause

$$\begin{aligned} c_q^{q_1} &= c_{p+1}^{q_1} \\ &= \{\overline{x}_{p+1}^{q_1}, x_{p+1}^1, \ldots, x_{p+1}^{q_1-1}, x_p^1, \ldots, x_p^{q_1-1}\}. \end{aligned}$$

Now $\mathcal{A}_2(x_{p+1}^{q_1}) =$ T, and none of the vertices $x^1, \ldots, x^{q_1-1}$, which all have superscripts smaller than q_1 and p_1, can belong to $S_q^+ = S_{p+1}^+$ nor to S_p^+. This implies that $c_q^{q_1}$ cannot be satisfied by $\mathcal{A}_2$, a contradiction. It follows that the k-IS-partition $S_1^+, \ldots, S_k^+$ is ordered according to the smallest superscript of the elements in its sets, i.e., it has the same set order as the k-IS-partition $S_1, \ldots, S_k$, which was our claim, and consequently $\mathcal{A}_1 = \mathcal{A}_2$, i.e., we have also a YES answer to U-SAT.

Example. Let G be a triangle: $k = 3$, $G = (V, E)$ with $V = \{x^1, x^2, x^3\}$, $E = \{x^1x^2, x^1x^3, x^2x^3\}$. The clauses are:
(0) $\{x_1^1\}$;
(i) $\{x_1^1, x_2^1, x_3^1\}$, $\{x_1^2, x_2^2, x_3^2\}$, *and* $\{x_1^3, x_2^3, x_3^3\}$;
(ii) $\{\overline{x}_1^1, \overline{x}_2^1\}$, $\{\overline{x}_1^1, \overline{x}_3^1\}$, $\{\overline{x}_2^1, \overline{x}_3^1\}$, $\{\overline{x}_1^2, \overline{x}_2^2\}$, $\{\overline{x}_1^2, \overline{x}_3^2\}$, $\{\overline{x}_2^2, \overline{x}_3^2\}$, $\{\overline{x}_1^3, \overline{x}_2^3\}$, $\{\overline{x}_1^3, \overline{x}_3^3\}$, *and* $\{\overline{x}_2^3, \overline{x}_3^3\}$;
(iii) $\{\overline{x}_1^1, \overline{x}_1^2\}$, $\{\overline{x}_2^1, \overline{x}_2^2\}$, $\{\overline{x}_3^1, \overline{x}_3^2\}$, $\{\overline{x}_1^1, \overline{x}_1^3\}$, $\{\overline{x}_2^1, \overline{x}_2^3\}$, $\{\overline{x}_3^1, \overline{x}_3^3\}$, $\{\overline{x}_1^2, \overline{x}_1^3\}$, $\{\overline{x}_2^2, \overline{x}_2^3\}$, *and* $\{\overline{x}_3^2, \overline{x}_3^3\}$.
If we stop here, two assignments $\mathcal{A}_1, \mathcal{A}_2$ are possible, one corresponding to $x^1 \in S_1$, $x^2 \in S_2$, $x^3 \in S_3$: $\mathcal{A}_1(x_1^1) = \mathcal{A}_1(x_2^2) = \mathcal{A}_1(x_3^3) = T$, $\mathcal{A}_1(x_2^1) = \mathcal{A}_1(x_3^1) = \mathcal{A}_1(x_1^2) = \mathcal{A}_1(x_3^2) = \mathcal{A}_1(x_1^3) = \mathcal{A}_1(x_2^3) = F$, the other corresponding to $x^1 \in S_1$, $x^2 \in S_3$, $x^3 \in S_2$: $\mathcal{A}_2(x_1^1) = \mathcal{A}_2(x_3^2) = \mathcal{A}_2(x_2^3) = T$, $\mathcal{A}_2(x_2^1) = \mathcal{A}_2(x_3^1) = \mathcal{A}_2(x_1^2) = \mathcal{A}_2(x_2^2) = \mathcal{A}_2(x_1^3) = \mathcal{A}_2(x_3^3) = F$. However, only $\mathcal{A}_1$ *satisfies the clauses (iv), which are:* $c_1^2 = \{\overline{x}_1^2, x_1^1\}$, $c_1^3 = \{\overline{x}_1^3, x_1^1, x_1^2\}$, $c_2^2 = \{\overline{x}_2^2, x_2^1, x_1^1\}$, $c_2^3 = \{\overline{x}_2^3, x_2^1, x_2^2, x_1^1, x_1^2\}$, $c_3^2 = \{\overline{x}_3^2, x_3^1, x_2^1\}$, *and* $c_3^3 = \{\overline{x}_3^3, x_3^1, x_3^2, x_2^1, x_2^2\}$. *Indeed, c_3^2 is not satisfied by $\mathcal{A}_2$.*

(2) Assume now that the answer to U-COL is negative. If it is negative because there are at least two k-IS-partitions of V, then we have at least two assignments satisfying the instance of U-SAT: we have seen above how to construct a suitable assignment from a k-IS-partition, and different partitions lead to different assignments. If there is no k-IS-partition, then there is no assignment satisfying U-SAT, because such an assignment would give a k-IS-partition, as we have seen above in the proof with $\mathcal{A}_2$. So in both cases, a NO answer to U-COL implies a NO answer to U-SAT.

In conclusion, the reduction saves the answer and is polynomial: thus U-COL $\leqslant$ U-SAT. △

As U-SAT and U-1-3-SAT have equivalent complexities (see the relationship (1) above), we directly obtain the conclusion stated by the following theorem as a consequence of the transitivity of the relation $\leqslant$ and of Theorem 5, Proposition 6 and Theorem 7:

Theorem 8 *The problems U-SAT, U-1-3-SAT, U-k-COL for every integer $k \geqslant 3$ and U-COL have equivalent complexities, up to polynomials. All are co-$\mathcal{NP}$-hard (and $\mathcal{NP}$-hard by Remark 1) and belong to the class $\mathcal{DP}$.* △

Note that it could have been shown directly that U-k-COL ($k \geqslant 1$) and U-COL belong to $\mathcal{DP}$. Indeed, for U-k-COL, we have to exhibit two languages $L_1 \in \mathcal{NP}$ and $L_2 \in$ co-$\mathcal{NP}$ such that the set of YES instances of U-k-COL is $L_1 \cap L_2$. To reach this aim, it is sufficient to define L_1 as $\{G$: there is at least one k-IS-partition in $G\}$ and L_2 as $\{G$: there is at most one k-IS-partition in $G\}$. We can proceed similarly for U-COL.

2.3 Location of U-OCOL

Proposition 9 below provides an upper bound on the complexity of U-OCOL in the complexity classes hierarchy.

Proposition 9 *The problem U-OCOL belongs to the class $\mathcal{L}^{\mathcal{NP}}$.*

Proof. Let us design an algorithm $\mathcal{A}_2$ solving U-OCOL based on an algorithm $\mathcal{A}_1$ solving COL, which belongs to $\mathcal{NP}$, in such a way that the number of calls to $\mathcal{A}_1$ is logarithmic with respect to the size of the instance; for this, consider any instance I of U-OCOL: I is defined by a graph G. So, if n denotes the order of G, the size $|I|$ of I is equal to n^2.

Step 1. Consider the instances of COL of the form (G, k) where G is the graph of I and k an integer between 1 and n. By applying a standard dichotomous process on k between 1 and n, we can build an algorithm $\mathcal{A}_2$ based on $\mathcal{A}_1$ and outputting the chromatic number $\chi(G)$ of G. In $\mathcal{A}_2$, the number of calls to $\mathcal{A}_1$ is about $\log_2 n$, i.e. is upper-bounded by $\log_2 |I|$.

So we may perform Step 1 with a logarithmic number (with respect to $|I|$) of calls to an algorithm solving a problem in $\mathcal{NP}$.

Step 2. Once we have computed $\chi(G)$, we question the instance $(G, \chi(G))$ of U-COL, and we get the answer to U-OCOL.

Since U-COL belongs to $\mathcal{DP}$ and because $\mathcal{DP}$ is a subset of $\mathcal{L}^{\mathcal{NP}}$, this step can also be performed with a logarithmic number (with respect to $|I|$) of calls to an algorithm solving a problem in $\mathcal{NP}$.

Conclusion: all in all, by performing Step 1 and then Step 2, we obtain an algorithm solving U-OCOL with a logarithmic number of calls to an algorithm solving a problem in $\mathcal{NP}$, which is the definition of the membership to U-OCOL to $\mathcal{L}^{\mathcal{NP}}$. $\triangle$

Note that using an algorithm for U-COL without knowing the chromatic number leads nowhere, because a NO answer cannot be interpreted unambiguously: either k is smaller than the chromatic number, and there is no colouring, or k is greater than or equal to the chromatic number, but there is more than one colouring.

3 Conclusion

Theorem 8 states that the four problems U-SAT, U-1-3-SAT, U-k-COL with $k \geqslant 3$ and U-COL are equivalent; they lie somewhere in the area of Figure 1 located below the $\mathcal{DP}$ line and above the $\mathcal{NP}$-hard dashed line; according to [23] (cf. Remark 2), they are probably not $\mathcal{DP}$-complete.

For the problem U-OCOL, we have a poorer result (Proposition 9): U-OCOL belongs to $\mathcal{L}^{\mathcal{NP}}$.

In [3], the authors wonder whether U-SAT is $\mathcal{NP}$-hard; but what they mean seems to be: "does there exist a *polynomial* reduction from an $\mathcal{NP}$-complete problem to U-SAT?" i.e., they use the *second* definition of $\mathcal{NP}$-hardness, based on polynomial reductions (see the paragraph after Remark 1).

They show that this would be the case if and only if U-SAT is $\mathcal{DP}$-complete. Because of the results stated in Theorem 8, the same applies to U-k-COL for $k \geqslant 3$ and for U-COL.

This constitutes a part of the concluding open problems.

Open problems.

1. Determine a better location for U-k-COL ($k \geqslant 3$), U-COL and U-OCOL in the classes of complexity.
2. Are U-k-COL ($k \geqslant 3$) and U-COL $\mathcal{DP}$-complete?
3. Is U-OCOL $\mathcal{L}^{\mathcal{NP}}$-complete?

References:

[1] J.-P. BARTHÉLEMY, G. D. COHEN and A. C. LOBSTEIN: *Algorithmic Complexity and Communication Problems*, University College of London: London, 1996.

[2] C. BERGE: *Graphes*, Gauthier-Villars: Paris, 1983. English translation: *Graphs*, North-Holland Publishing Co.: Amsterdam, 1985.

[3] A. BLASS and Y. GUREVICH: On the unique satisfiability problem, *Information and Control*, Vol. 55, pp. 80-88, 1982.

[4] M. BLIDIA, M. CHELLALI, R. LOUNES and F. MAFFRAY: Characterizations of trees with unique minimum locating-dominating sets, *Journal of Combinatorial Mathematics and Combinatorial Computing*, Vol. 76, pp. 225–232, 2011.

[5] C. CALABRO, R. IMPAGLIAZZO, V. KABANETS and R. PATURI: The complexity of Unique k-SAT: an isolation lemma for k-CNFs, *Journal of Computer and System Sciences*, Vol. 74, pp. 386–393, 2008.

[6] S. A. COOK: The complexity of theorem-proving procedures, *Proceedings of 3rd Annual ACM Symposium on Theory of Computing*, pp. 151–158, 1971.

[7] T. CORMEN: Algorithmic complexity, in: K. H. Rosen (ed.) *Handbook of discrete and combinatorial mathematics*, pp. 1077–1085, CRC Press: Boca Raton, 2000.

[8] R. DIESTEL: *Graph Theory*, Springer-Verlag: Berlin, 2005.

[9] M. FISCHERMANN: Block graphs with unique minimum dominating sets, *Discrete Mathematics*, Vol. 240, pp. 247–251, 2001.

[10] M. R. GAREY and D. S. JOHNSON: *Computers and Intractability, a Guide to the Theory of NP-Completeness*, Freeman: New York, 1979.

[11] M. R. GAREY, D. S. JOHNSON and L. STOCKMEYER: Some simplified *NP*-complete graph problems, *Theoretical Computer Science*, Vol. 1(3), pp. 237–267, 1976.

[12] G. GUNTHER, B. HARTNELL, L. R. MARKUS and D. RALL: Graphs with unique minimum dominating sets, *Congressus Numerantium*, Vol. 101, pp. 55–63, 1994.

[13] P. HANSEN and B. JAUMARD: Uniquely solvable quadratic Boolean equations, *Discrete Applied Mathematics*, Vol. 12(2), pp. 147-154, 1985.

[14] L. A. HEMACHANDRA: The strong exponential hierarchy collapses, *Journal of Computer and System Sciences*, Vol. 39, pp. 299–322, 1989.

[15] L. HEMASPAANDRA: Complexity classes, in: K. H. Rosen (ed.) *Handbook of discrete and combinatorial mathematics*, pp. 1085–1090, CRC Press: Boca Raton, 2000.

[16] O. HUDRY and A. LOBSTEIN: Unique (optimal) solutions: complexity results for identifying and locating-dominating codes, *Theoretical Computer Science*, Vol. 767, pp. 83–102, 2019.

[17] O. HUDRY and A. LOBSTEIN: Complexity of unique (optimal) solutions in graphs: Vertex Cover and Domination, *Journal of Combinatorial Mathematics and Combinatorial Computing*, Vol. 110, pp. 217–240, 2019.

[18] O. HUDRY and A. LOBSTEIN: On the complexity of determining whether there is a unique Hamiltonian cycle in a graph, *WSEAS Transactions on Mathematics*, Vol. 21, pp. 433–446, 2022.

[19] D. S. JOHNSON: A catalog of complexity classes, in: J. van Leeuwen (ed.), *Handbook of theoretical computer science, Vol. A: Algorithms and complexity*, Chapter 2, Elsevier: Amsterdam, 1990.

[20] L. JUBAN: Dichotomy theorem for the generalized unique satisfiability problem, in: G. Ciobanu and G. Paun (eds.) *Fundamentals of computation theory* (FCT'99), pp. 327-337, LNCS 1684: Berlin, 1999.

[21] M. MINOUX: The Unique Horn-Satisfiability problem and quadratic Boolean equations, *Annals of Mathematics and Artificial Intelligence*, Vol. 6, pp. 253-266, 1992.

[22] C. H. PAPADIMITRIOU: On the complexity of unique solutions, *Journal of the Association for Computing Machinery*, Vol. 31, pp. 392-400, 1984.

[23] C. H. PAPADIMITRIOU: *Computational Complexity*, Addison-Wesley: Reading, 1994.

[24] C. H. PAPADIMITRIOU and M. YANNAKAKIS: The complexity of facets (and some facets of complexity), *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*, San Francisco, May 1982.

[25] C. H. PAPADIMITRIOU and M. YANNAKAKIS: The complexity of facets (and some facets of complexity), *Journal of Computer and System Sciences*, Vol. 28, pp. 244–259, 1984.

[26] T. J. SCHAEFER: The complexity of satisfiability problems, *Proceedings of 10th Annual ACM Symposium on Theory of Computing*, pp. 216–226, 1978.

[27] L. G. VALIANT and V. V. VAZIRANI: NP is as easy as detecting unique solutions, *Theoretical Computer Science*, Vol. 47 (1), pp. 85-93, 1986.

[28] Complexity Zoo, MediaWiki, Available at: https://complexityzoo.net/Complexity_Zoo (Accessed June 12, 2023)

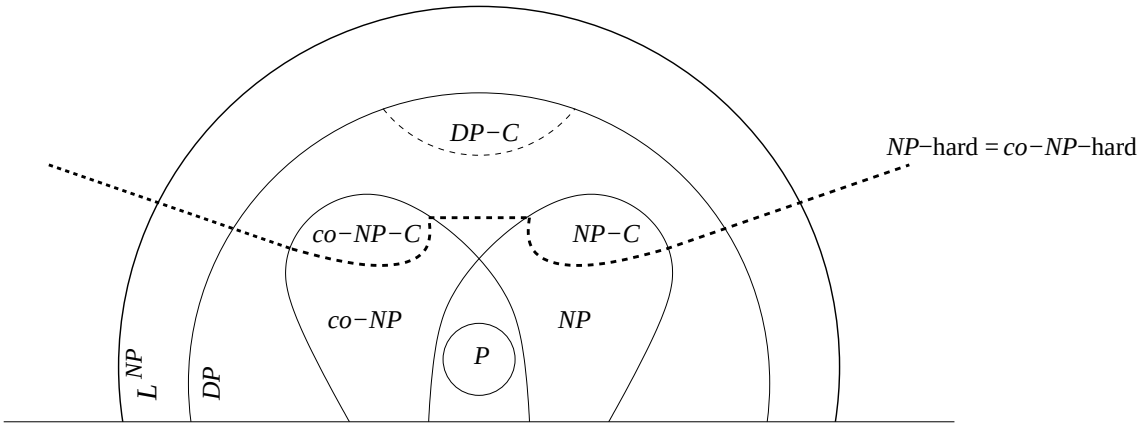

Figure 1: Some classes of complexity.

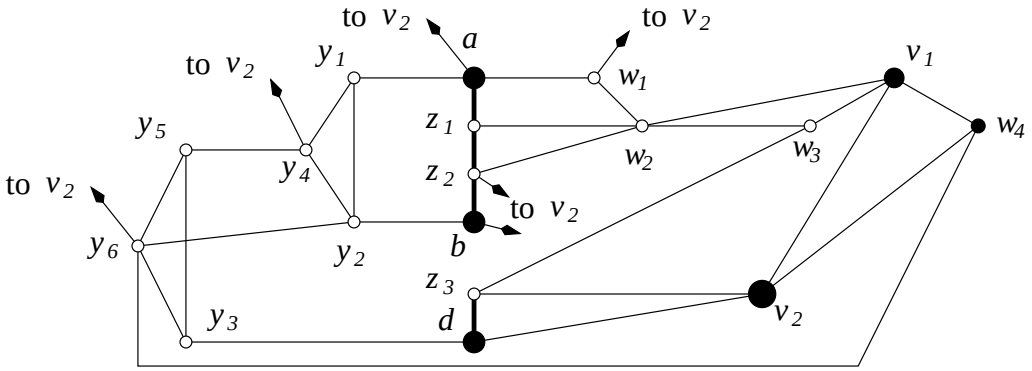

Figure 2: The graph G_0 of Lemma 3. Because of their particular roles, the vertices v_1, v_2, w_4, a, b and d are represented by black circles.

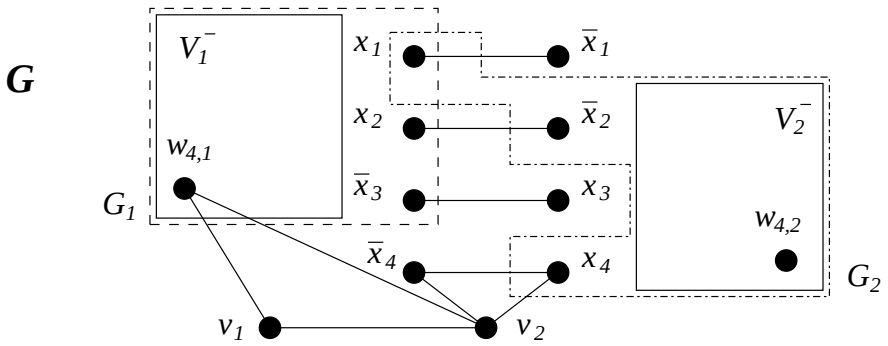

Figure 3: A small example for Theorem 5: $c_1 = \{x_1, x_2, \overline{x}_3\}$, $c_2 = \{x_1, \overline{x}_2, x_4\}$. Some vertices or edges are missing.

Contribution of individual authors to the creation of a scientific article (ghostwriting policy)

The two authors contribute equally to all the aspects of this work.

Sources of funding for research presented in a scientific article or scientific article itself

None.

Conflict of interest

None.