Complexité de l'unicité de solutions dans des problèmes de graphes

On the complexity of the uniqueness of solutions in graph problems

---

Olivier Hudry
Antoine Lobstein

2017D001

avril 2017

Département Informatique et réseaux
Groupe MIC2 : Mathématiques de l'Information, des Communications et du Calcul

Complexité de l'unicité de solutions dans des problèmes de graphes

Olivier Hudry
Département Informatique et Réseaux
LTCI, Télécom ParisTech, Université Paris-Saclay
46 rue Barrault, 75634 Paris Cedex 13 - France
& Antoine Lobstein
Centre National de la Recherche Scientifique
Laboratoire de Recherche en Informatique, UMR 8623,
Université Paris-Sud, Université Paris-Saclay
Bâtiment 650 Ada Lovelace, 91405 Orsay Cedex - France

olivier.hudry@telecom-paristech.fr
antoine.lobstein@lri.fr

Résumé : Ce rapport est constitué de quatre articles soumis à des revues internationales de mathématiques avec comité de lecture, sur un sujet général commun, à savoir l'étude de la complexité de problèmes traitant de l'unicité de la solution, pour des problèmes classiques en théorie de la complexité, que ce soient des problèmes de satisfiabilité de formules booléennes, ou des problèmes de graphes. Plus précisément :

le premier article traite des problèmes de satisfiabilité SAT, k-SAT, $k \geq 2$, et 1-3-SAT, ainsi que du problème de l'existence de colorations dans un graphe donné *(Sections 1–4 dans ce rapport)* ;

le deuxième article traite de problèmes de recouvrement d'arêtes par des sommets d'une part, et de problèmes d'ensembles dominants (ou codes dominateurs) d'autre part, dans un graphe donné *(Sections 5–8)* ;

le troisième article traite de problèmes de codes identifiants, et de codes localisateurs-dominateurs, dans un graphe donné *(Sections 9–13)* ;

le quatrième et dernier article traite de problèmes de cycles et de chaînes

hamiltoniens, dans un graphe donné, orienté ou non-orienté *(Sections 14–16)*.

Pour tous ces problèmes, nous essayons de trouver des localisations précises à l'intérieur de l'ensemble des classes de complexité, et d'établir des équivalences entre problèmes, principalement à l'aide de réductions polynomiales d'un problème à un autre.

Mots-clés : Théorie de la complexité : NP-complétude, NP-difficulté, Unicité d'une solution (optimale), Réduction polynomiale ; Satisfiabilité booléenne ; Théorie des graphes : Coloration de graphes, Domination, Codes dominateurs, Recouvrements par des sommets, Codes localisateurs-dominateurs, Codes identifiants, Graphes sans jumeaux, Cycles et chaînes hamiltoniens.

On the Complexity of the Uniqueness of Solutions in Graph Problems

Olivier Hudry
LTCI, Télécom ParisTech, Université Paris-Saclay
46 rue Barrault, 75634 Paris Cedex 13 - France
& Antoine Lobstein
Centre National de la Recherche Scientifique
Laboratoire de Recherche en Informatique, UMR 8623,
Université Paris-Sud, Université Paris-Saclay
Bâtiment 650 Ada Lovelace, 91405 Orsay Cedex - France

olivier.hudry@telecom-paristech.fr, antoine.lobstein@lri.fr

Abstract: This Report contains four articles submitted to international journals of mathematics with reviewing process, all about the same general topic, namely, the study of the complexity of problems dealing with the uniqueness of a solution, for classical problems in complexity theory, either Boolean satisfiability problems, or graph problems. More specifically:

the first article deals with the satisfiability problems SAT, k-SAT, $k \geq 2$, and 1-3-SAT, as well as the problem of the existence of colourings in a given graph *(Sections 1–4 in this Report)*;

the second article deals with the vertex cover problem and the dominating set (or code) problem in a given graph *(Sections 5–8)*;

the third article deals with the locating-dominating and the identifying problems, in a given graph *(Sections 9–13)*;

the fourth and last article deals with the problem of finding Hamiltonian cycles or paths in a directed, oriented or undirected given graph *(Sections 14–16)*.

We try to locate accurately these problems inside the classes of complexity, and to establish equivalences between problems, in particular by constructing polynomial reductions from one problem to another.

Key Words: Complexity Theory: NP-Completeness, NP-Hardness, Uniqueness of an (Optimal) Solution, Polynomial Reduction; Boolean Satisfiability Problems; Graph Theory: Graph Colouring, Domination, Dominating Codes, Vertex Covers, Locating-Dominating Codes, Identifying Codes, Twin-Free Graphs, Hamiltonian Cycle and Path.

Some Complexity Considerations on the Uniqueness of Solutions for Satisfiability and Colouring Problems

Olivier Hudry
LTCI, Télécom ParisTech, Université Paris-Saclay
46 rue Barrault, 75634 Paris Cedex 13 - France

& Antoine Lobstein
Centre National de la Recherche Scientifique
Laboratoire de Recherche en Informatique, UMR 8623,
Université Paris-Sud, Université Paris-Saclay
Bâtiment 650 Ada Lovelace, 91405 Orsay Cedex - France

olivier.hudry@telecom-paristech.fr
antoine.lobstein@lri.fr

Abstract

For some well-known *NP*-complete problems, linked to the satisfiability of Boolean formulas and the colourability of a graph, we study the variation which consists in asking about the *uniqueness* of a solution. In particular, we show that five decision problems, Unique Satisfiability (U-SAT), Unique k-Satisfiability (U-k-SAT), $k \geq 3$, Unique One-in-Three Satisfiability (U-1-3-SAT), Unique k-Colouring (U-k-COL), $k \geq 3$, and Unique Colouring (U-COL), have equivalent complexities, up to polynomials —when dealing with colourings, we forbid permutations of colours. As a consequence, all are *NP*-hard and belong to the class *DP*. We also consider the problems U-2-SAT, U-2-COL and Unique Optimal Colouring (U-OCOL).

Key Words: Complexity Theory, *NP*-hardness, Decision Problems, Polynomial Reduction, Turing Reduction, Uniqueness of Solution, Boolean Satisfiability, Graph Theory, Graph Colouring, Partition into Independent Sets

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 $\mathcal{PR}$ on I, is $\mathcal{PR}$ true for I?" where $\mathcal{PR}$ can be expressed as: "Is there a set satisfying a given characteristic?". In our study, we add the small extra word "unique" to the latter question: "Is there a *unique* set satisfying a given characteristic?" Then we investigate the complexity of these newly defined problems. The problems considered here are of two types: Boolean satisfiability of formulas, and graph colouring (up to colour permutations).

In Section 1.2, we give some notation and definitions from graph theory and then, in Section 1.3, the basic background for the theory of *NP*-completeness; in Section 1.4, we present some well-known decision problems, together with their complexities. In Sections 2 and 3, we study how these complexities vary if we consider the question of the uniqueness of a solution, for satisfiability and colouring problems. We prove that the problems called below: U-SAT, U-1-3-SAT, U-k-SAT, U-k-COL and U-COL have equivalent complexities; consequently, they all belong to the class *DP* and are *NP*-hard. We also show that U-2-SAT is polynomial (as U-2-COL trivially is), and that U-OCOL belongs to the class L^{NP}. We present some concluding remarks in Section 4.

In a forthcoming work, we similarly revisit 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) [35] *(= Sections 5–8 in this Report)*, r-Identifying Code together with r-Locating-Dominating Code [36] *(Sections 9–13)*, and Hamiltonian Cycle [37] *(Sections 14–16)*.

1.2 Basic Notation and Definitions for Graphs

For graph theory, we refer to, e.g., [7] or [16] —and to [5] for directed graphs.

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, $|V|$.

An *independent set*, or *stable set*, is a subset $V^* \subseteq V$ such that for all $u \in V^*$, $v \in V^*$, $uv \notin E$. If k is an integer, $k \geq 1$, 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 have 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 with 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* of G. A colouring with a number of colours equal to the chromatic number is said to be *optimal*.

In Section 2.1, we shall need directed graphs. A *directed graph* (or simply *digraph*) $\overrightarrow{G} = (V, A)$ has *directed edges* or *arcs* $(x \to y) \in A$, $x \in V$, $y \in V$. A *strongly connected component* (SCC) of $\overrightarrow{G}$ is a maximal set (for inclusion) $S \subseteq V$ such that for all x, $y \in S$, there is a directed path from x to y, and a directed path from y to x. If S_1 and S_2 are two SCCs with an arc going from a vertex in S_1 to a vertex in S_2 (for short, we say that there in an arc from S_1 to S_2), then S_2 is a *successor* of S_1, and S_1 is a *predecessor* of S_2; note that in this case, there can be no arc from S_2 to S_1, by definition. Two digraphs $\overrightarrow{G_1} = (V, A_1)$ and $\overrightarrow{G_2} = (V, A_2)$ are *isomorphic* if there is a permutation p on V such that $(x \to y) \in A_1$ if and only if $(p(x) \to p(y)) \in A_2$.

1.3 Necessary Notions in Complexity

We expound here, not too formally, the notions of complexity that will be needed in the sequel. We refer the reader to, e.g., [6], [21], [38] or [45] for more on this topic.

A *decision problem* is of the type "Given an instance I and a property $\mathcal{PR}$ on I, is $\mathcal{PR}$ true for I?", and has only two solutions, YES or NO. The class P will denote the set of problems which can be solved by a *polynomial* (time) algorithm, and the class 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 transformation 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 shall write $\pi_1 \to_p \pi_2$. Cook [14] proved that there is one problem in NP, namely "Satisfiability" or simply SAT (see below in Section 1.4), to which every other problem in NP can be polynomially reduced. Thus, in a sense, SAT is the "hardest" problem inside NP. Other problems share this property in NP and are called *NP-complete* problems; their class is denoted by *NP-C*. The way to show that a decision problem π is NP-complete is, once it is proved to be in NP, to choose some NP-complete problem π_1

and to polynomially reduce it to π. From a practical viewpoint, the *NP*-completeness of a problem π implies that we do not know any polynomial algorithm solving π, and that, under the assumption $P \neq NP$, 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).

The *complement* of a decision problem, "Given I and $\mathcal{PR}$, is $\mathcal{PR}$ true for I?", is "Given I and $\mathcal{PR}$, is $\mathcal{PR}$ false for I?". The class *co-NP* (respectively, *co-NP-C*) is the class of the problems which are the complement of a problem in *NP* (respectively, *NP-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 shall write $\pi_1 \rightarrow_T \pi_2$. Thus, in this sense, π_2 is "at least as hard" as π_1. A problem π is *NP-hard* (respectively, *co-NP-hard*) if there is a Turing reduction from some *NP*-complete (respectively, co-*NP*-complete) problem to π [21, p. 113].

Remark 1 *Note that with these definitions, NP-hard and co-NP-hard coincide [21, p. 114].*

The notions of completeness and hardness can of course be extended to classes other than *NP* or co-*NP*. *NP*-hardness is defined differently in [15] and [27]: there, a problem π is *NP-hard* if there is a *polynomial* reduction from some *NP*-complete problem to π; this may lead to confusion (see Section 4).

We also introduce the classes P^{NP} (also known as Δ_2 in the hierarchy of classes) and L^{NP} (also denoted by $P^{NP[O(\log n)]}$ or Θ_2): they contain the decision problems which can be solved by applying, with a number of calls which is polynomial (respectively, logarithmic) with respect to (wrt) the size of the instance, a subprogram able to solve an appropriate problem in *NP* (usually, an *NP*-complete problem).

Finally, let us define *DP* [46] (or DIF^P [8] or BH_2 [38], [52], ...) as the class of languages (or problems) L such that there are two languages $L_1 \in NP$ and $L_2 \in$ co-*NP* satisfying $L = L_1 \cap L_2$. This class is not to be confused with $NP \cap$ co-*NP* (see the warning in, e.g., [45, p. 412]); actually, *DP* contains $NP \cup$ co-*NP* and is contained in L^{NP} (see Figure 1).

Membership to *P*, *NP*, co-*NP*, *DP*, L^{NP} or P^{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

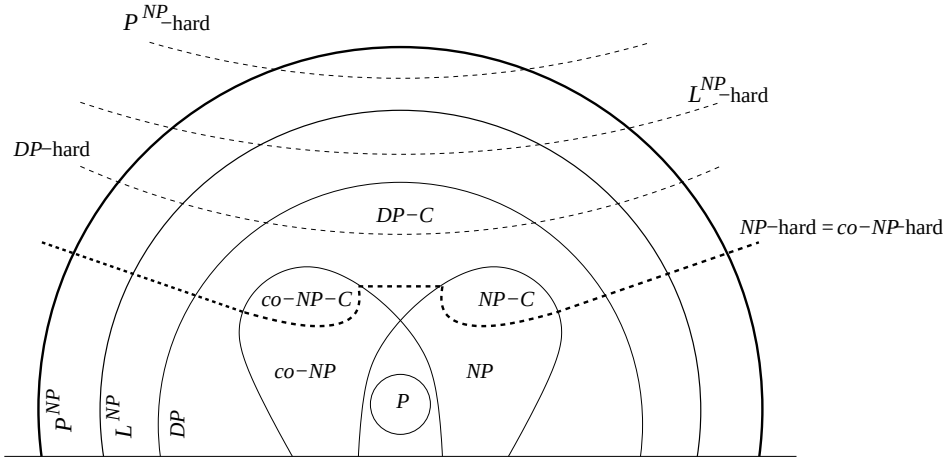

Figure 1: Some classes of complexity.

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

1.4 Satisfiability and Colouring Problems

We first introduce 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 and a set $\mathcal{C}$ of m *clauses* ($\mathcal{C}$ is also called a *Boolean formula*); each clause c_j contains κ_j *literals*, a literal being a variable x_i or its complement (or negated variable) $\overline{x}_i$. Equivalently, $\mathcal{C}$ can read as a logical formula

$$\mathcal{C} = \wedge_{1\leq j\leq m} c_j, \text{ with } c_j = \vee_{1\leq i\leq \kappa_j} \ell_i. \tag{1}$$

This form is called a *normal conjunctive form*; other forms can be used when more convenient, see the proof of Proposition 12. 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 $\mathcal{X}$, each clause

containing at least two different literals.
Question: Is there a truth assignment for $\mathcal{X}$ that satisfies $\mathcal{C}$?

Remark 2 *When useful, we can assume, without loss of generality (wlog), that at least one clause contains only negated variables; indeed, if this is not the case, i.e., no clause contains only negated variables, then obviously the assignment giving TRUE to all the variables satisfies all the clauses, so that this instance is trivial and can be discarded. We shall use this remark in order to simplify the proof of Proposition 12.*

The following problem is stated for any fixed integer $k \geq 2$, the case $k = 1$ being trivial:

Problem k-SAT (k-Satisfiability):
Instance: A set $\mathcal{X}$ of variables, a collection $\mathcal{C}$ of clauses over $\mathcal{X}$, each clause containing exactly k different literals.
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, or two different literals and the constant FALSE.
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 co-SAT (co-Satisfiability):
Instance: A set $\mathcal{X}$ of variables, a collection $\mathcal{C}$ of clauses over $\mathcal{X}$, each clause containing at least two different literals.
Question: Is it true that no truth assignment for $\mathcal{X}$ satisfies $\mathcal{C}$?

In the same way, we can define co-k-SAT for $k \geq 2$.

The problems SAT and 3-SAT are two of the basic and most well-known *NP*-complete problems [14], [21, p. 39, p. 46 and p. 259]. More generally, k-SAT is *NP*-complete for $k \geq 3$ and polynomial for $k = 2$. It follows that the problems co-SAT and co-k-SAT, $k \geq 3$, are co-*NP*-complete.

The problem 1-3-SAT, which is obvioulsy in *NP*, is also *NP*-complete [48, Lemma 3.5], [21, p. 259]. Because we shall need it later on for Proposition 6, we give one polynomial reduction from 3-SAT to 1-3-SAT:

Consider an instance of 3-SAT. For each clause $c_i = \{\ell_1, \ell_2, \ell_3\}$, where ℓ_1, ℓ_2, ℓ_3 are literals stemming from $\mathcal{X}$, one creates six variables a_i, b_i, d_i, e_i, f_i

and g_i, and five clauses $c_{i,1} = \{\ell_1, a_i, e_i\}$, $c_{i,2} = \{\ell_2, b_i, e_i\}$, $c_{i,3} = \{a_i, b_i, f_i\}$, $c_{i,4} = \{d_i, e_i, g_i\}$, $c_{i,5} = \{\ell_3, d_i, \text{FALSE}\}$, for 1-3-SAT.

First, note that if $\ell_1 = \ell_2 = \ell_3 = \text{F}$, then there is no truth assignment (for a_i, b_i, d_i, e_i, f_i and g_i) 1-3-satisfying all the clauses $c_{i,j}$, $1 \leq j \leq 5$; indeed, clause $c_{i,5}$ implies that $d_i = \text{T}$, which implies by clause $c_{i,4}$ that $e_i = \text{F}$, which in turn implies, by clauses $c_{i,1}$ and $c_{i,2}$, that $a_i = b_i = \text{T}$, and then $c_{i,3}$ contains at least two true literals.

Then, from the Truth Table below, which shows under each clause $c_{i,j}$ the only true literal that it contains, we can see that there is an assignment satisfying c_i if and only if there is an assignment 1-3-satisfying each clause $c_{i,j}$.

	ℓ_1	ℓ_2	ℓ_3	a_i	b_i	d_i	e_i	f_i	g_i	$c_{i,1}$	$c_{i,2}$	$c_{i,3}$	$c_{i,4}$	$c_{i,5}$
1	T	T	T	F	F	F	F	T	T	ℓ_1	ℓ_2	f_i	g_i	ℓ_3
2	T	T	F	F	F	T	F	T	F	ℓ_1	ℓ_2	f_i	d_i	d_i
3	T	F	T	F	T	F	F	F	T	ℓ_1	b_i	b_i	g_i	ℓ_3
4	F	T	T	T	F	F	F	F	T	a_i	ℓ_2	a_i	g_i	ℓ_3
5	T	F	F	F	T	T	F	F	F	ℓ_1	b_i	b_i	d_i	d_i
6	F	T	F	T	F	T	F	F	F	a_i	ℓ_2	a_i	d_i	d_i
7	F	F	T	F	F	F	T	T	F	e_i	e_i	f_i	e_i	ℓ_3

From this, we can conclude that there is a truth assignment for $\mathcal{X}$ satisfying the collection $\mathcal{C}$ of m clauses we started from if and only if there is a truth assignment for $\mathcal{X}^* = \mathcal{X} \cup \{a_i, b_i, d_i, e_i, f_i, g_i : 1 \leq i \leq m\}$ such that each of the $5m$ clauses $c_{i,j}$, $1 \leq i \leq m$, $1 \leq j \leq 5$, is 1-3-satisfied. This is sufficient to prove that 1-3-SAT is *NP*-complete.

Remark 3 *Actually, one can get rid of FALSE in every clause $c_{i,5}$, and have clauses only of "real" size three: create three new variables α, β, γ, which are added to $\mathcal{X}^*$, and three new clauses $\{\alpha, \beta, \gamma\}$, $\{\alpha, \overline{\beta}, \overline{\gamma}\}$ and $\{\overline{\alpha}, \overline{\beta}, \gamma\}$, which are added to the $5m$ clauses $c_{i,j}$. It is then straightforward to check that there is one, and only one, way of* 1-3*-satisfying the new clauses: one must take $\alpha = F$, $\beta = T$, $\gamma = F$. From this, we can deduce that each clause $c_{i,5} = \{\ell_3, d_i, FALSE\}$, $1 \leq i \leq m$, can be replaced by, e.g., $\{\ell_3, d_i, \alpha\}$, in order to obtain a set of clauses of size three which can be* 1-3*-satisfied if and only if there is a truth assignment satisfying the clauses c_i, $1 \leq i \leq m$, from which we started.*

So from now on, we shall use 1-3-SAT exclusively in the following form.

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 now turn to the colouring decision problem, stated in its two usual forms; first, for a given integer $k \geq 1$:

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 *NP*-complete for $k \geq 3$, see [22], [21, p. 191]. We can also state the problem with k belonging to the instance, in which case it is also *NP*-complete, since it still belongs to *NP*:

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

Finally, we denote by U-SAT, U-k-SAT, U-1-3-SAT, U-k-COL and U-COL the five problems obtained from the above five problems SAT, k-SAT, 1-3-SAT, k-COL and COL by putting the word "unique" in the question, and we add the following one, which would be meaningless without the word "unique" in its question:

Problem U-OCOL (Unique Optimal Colouring):
Instance: A graph G.
Question: Does G admit a unique optimal colouring?

As already mentioned, the uniqueness of colourings must be considered up to colour permutations, see the beginning of Section 3.

2 Uniqueness of Solutions: Satisfiability

2.1 Polynomiality of U-2-SAT

Theorem 4 *The problem U-2-SAT is polynomial.*

Proof. In [1], from an instance of 2-SAT with n variables x_i and a set $\mathcal{C}$ of m clauses c_j, the following digraph $\overrightarrow{G} = (V, A)$, with $2n$ vertices $x_i, \overline{x}_i$ and $2m$ arcs, is constructed: for each clause $c_j = \{\ell_1, \ell_2\}$, one puts the arcs $(\overline{\ell}_1 \to \ell_2)$ and $(\overline{\ell}_2 \to \ell_1)$ in A.

This graph has the following *duality property*: it is isomorphic to the graph obtained by reversing the arcs and negating the names of the vertices. Therefore, every strongly connected component S of $\overrightarrow{G}$ has a dual component $\overline{S}$ which is the subgraph induced by the negated vertices of S —it may be that $S = \overline{S}$. If S_1 is a predecessor of S_2, then $\overline{S}_1$ is a successor of $\overline{S}_2$, and *vice-versa*.

It is then shown in [1] that the instance of 2-SAT can be satisfied if and only if, for all $i \in \{1, \ldots, n\}$, no SCC of $\overrightarrow{G}$ contains both x_i and $\overline{x}_i$. The latter can be checked in polynomial time [51] wrt the order of the graph $\overrightarrow{G}$, which is itself polynomial wrt the size of the instance of 2-SAT; this proves that 2-SAT is polynomial.

We shall use the very same construction for U-2-SAT, and check the SCCs of $\overrightarrow{G}$. If one SCC contains some pair $x_i, \overline{x}_i$, then the answer to 2-SAT, and *a fortiori* to U-2-SAT, is NO. So from now on, we assume that no pair $x_i, \overline{x}_i$ belongs to the same SCC. This implies a YES answer for 2-SAT. But what about U-2-SAT?

Assume first that there is a directed path from some x_i to $\overline{x}_i$ (if it is the other way round, the argument is the same): $(x_i \to \ell_1) \in A$, $(\ell_1 \to \ell_2) \in A$, $\ldots$, $(\ell_s \to \overline{x}_i) \in A$. This means that the clauses $\{\overline{x}_i, \ell_1\}$, $\{\overline{\ell}_1, \ell_2\}$, $\ldots$, $\{\overline{\ell}_s, \overline{x}_i\}$ all belong to $\mathcal{C}$. Obviously, this leads to $x_i = \mathrm{F}$ (no choice for x_i). If this is true for all $i \in \{1, \ldots, n\}$, then the answer to U-2-SAT is YES: only one assignment satisfies the set $\mathcal{C}$ of clauses. So from now on, we assume that there is at least one pair $x_i, \overline{x}_i$ such that there is no directed path from x_i to $\overline{x}_i$, nor from $\overline{x}_i$ to x_i. This implies that no arc exists going from S_{x_i} to $S_{\overline{x}_i}$, nor from $S_{\overline{x}_i}$ to S_{x_i}.

Let $\sigma(\overrightarrow{G})$ be the number of SCCs of $\overrightarrow{G}$. A *topological order* (TO for short) Φ of the SCCs of $\overrightarrow{G}$, from 1 to $\sigma(\overrightarrow{G})$, is such that if S_1 is a predecessor of S_2, then $\Phi(S_1) < \Phi(S_2)$; in general, there are many ways to choose one TO, and there exists always at least one. This is used in [1] in order to define an assignment of the variables: if S_{x_i} is the SCC containing x_i (and $S_{\overline{x}_i} = \overline{S}_{x_i}$ is the SCC containing $\overline{x}_i$), set $x_i = \mathrm{F}$ if and only if $\Phi(S_{x_i}) < \Phi(S_{\overline{x}_i})$; with the assumption that the answer to 2-SAT is YES, every TO defines an assignment satisfying $\mathcal{C}$. But for U-2-SAT, we have to go further.

Let $\Phi(S_{x_i}) = k$ and $\Phi(S_{\overline{x}_i}) = \ell$, with, say, $k < \ell$. This means that there is an assignment satisfying $\mathcal{C}$, with $x_i = \mathrm{F}$. If $k + 1 = \ell$, take $\Phi^* = \Phi$ except for $\Phi^*(S_{x_i}) = \ell$ and $\Phi^*(S_{\overline{x}_i}) = k$: Φ^* still is a TO, because we assumed that there is no arc between S_{x_i} and $S_{\overline{x}_i}$, and we have another assignment satisfying $\mathcal{C}$, this time with $x_i = \mathrm{T}$, i.e., a NO answer to U-2-SAT. So from now on, we assume that $k + 1 < \ell$.

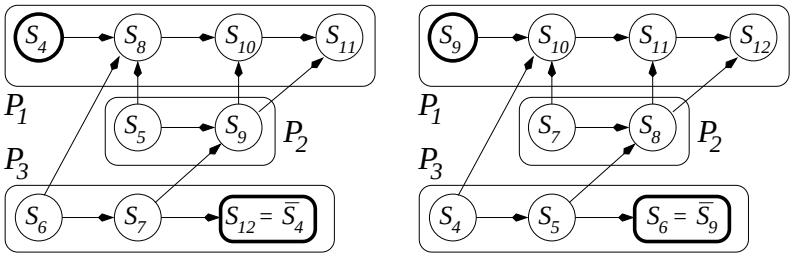

Figure 2: An example for the proof of Theorem 4, with $k = 4$ and $\ell = 12$; on the left, the Φ-order inside P_1, P_2 and P_3, on the right, the Φ^*-order.

We set $S_{x_i} = S_k$, $S_{\overline{x}_i} = S_\ell$, and, for $j \in \{k+1, \ldots, \ell-1\}$, $S_j = \Phi^{-1}(j)$. Because we use a TO, S_k has no predecessor among these SCCs, and S_ℓ has no successor. We shall partition these $\ell - k + 1$ SCCs into three sets, P_1, P_2 and P_3, according to the following rules (note that P_2 can be empty):

(i) in P_1, we put S_k and every SCC of index $j_1 \in \{k+1, \ldots, \ell-1\}$ such that there is a path from S_k to S_{j_1};

(ii) in P_2, we put every SCC of index $j_2 \in \{k+1, \ldots, \ell-1\}$ such that there is no path from S_k to S_{j_2}, nor from S_{j_2} to S_ℓ;

(iii) in P_3, we put S_ℓ and every SCC of index $j_3 \in \{k+1, \ldots, \ell-1\}$ such that there is a path from S_{j_3} to S_ℓ.

There is no path from P_1 to P_2, since a path from some $S_{j_1} \in P_1$ to some $S_{j_2} \in P_2$ means that S_{j_2} should have been put in P_1; similarly, there is no path from P_2 to P_3. By assumption, there is no path from P_1 to P_3, since this would lead to a path from x_i to $\overline{x}_i$. Paths from P_2 to P_1, from P_3 to P_2, and from P_3 to P_1 are possible.

Now we re-order the SCCs, starting from P_3 and ending with P_1; we define Φ^* as follows:

(0) $\Phi^* = \Phi$ outside $\{S_k, \ldots, S_\ell\}$;

(i) if S_{q_1}, S_{q_2}, ..., $S_{q_{|P_3|}}(= S_\ell)$ belong to P_3, with $k < q_1 < q_2 < \ldots < q_{|P_3|} = \ell$, we set $\Phi^*(S_{q_j}) = (k-1) + j$, for $j \in \{1, \ldots, |P_3|\}$;

(ii) if S_{t_1}, S_{t_2}, ..., $S_{t_{|P_2|}}$ belong to P_2, with $k < t_1 < t_2 < \ldots < t_{|P_2|} < \ell$, we set $\Phi^*(S_{t_j}) = (k-1) + |P_3| + j$, for $j \in \{1, \ldots, |P_2|\}$;

(iii) if $S_{w_1}(= S_k)$, S_{w_2}, ..., $S_{w_{|P_1|}}$ belong to P_1, with $k = w_1 < w_2 < \ldots < w_{|P_1|} < \ell$, we set $\Phi^*(S_{w_j}) = (k-1)+|P_3|+|P_2|+j$, for $j \in \{1, \ldots, |P_1|\}$.

See Figure 2 for an example. Because of the forbidden paths aforementioned and because Φ is a TO, Φ^* is also a TO, and we have now $\Phi^*(S_{x_i}) > \Phi^*(S_{\overline{x}_i})$, which means that there is a second assignment satisfy-

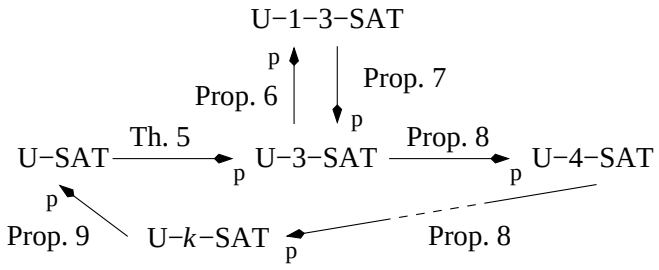

Figure 3: The polynomial reductions between satisfiability problems.

ing $\mathcal{C}$, this time with $x_i = \mathrm{T}$, i.e., a NO answer to U-2-SAT.

Recapitulating,

(a) if at least one pair $x_i, \overline{x}_i$ is contained in the same SCC, then the answer to U-2-SAT is NO (no assignment satisfying $\mathcal{C}$ exists); otherwise,

(b) if for all $i \in \{1, \ldots, n\}$, there is a path from x_i to $\overline{x}_i$ or from $\overline{x}_i$ to x_i, then the answer to U-2-SAT is YES (a unique assignment satisfying $\mathcal{C}$ exists);

(c) if there is at least one pair $x_i, \overline{x}_i$ such that there is no directed path from x_i to $\overline{x}_i$, nor from $\overline{x}_i$ to x_i, then the answer to U-2-SAT is NO (at least two assignments satisfying $\mathcal{C}$ exist).

All this can be checked in polynomial time. △

2.2 Equivalence of U-Satisfiability Problems

Consider the problem 1-3-SAT. The fact that an *NP*-complete problem, here 3-SAT, can be polynomially reduced to it, together with the fact that 1-3-SAT belongs to *NP*, suffices to prove that the complexities of these two problems are equivalent, up to polynomials: in this case, there is no need to give a polynomial reduction from 1-3-SAT to 3-SAT. In other situations, such as the problems that, in the sequel, we shall be interested in, it can be useful, or even necessary, to establish "cyclic" reductions such as $\pi_1 \rightarrow_p \pi_2 \rightarrow_p \pi_3 \rightarrow_p \pi_1$, in order to prove the equivalence of these three problems. This is what we shall do in this Section for our satisfiability problems, establishing the chain of polynomial reductions given by Figure 3. As a consequence, all these problems are equivalent, up to polynomials, see Theorem 10.

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

Proof. We consider an instance of U-SAT with n variables x_i and m clauses c_j. We assume that there is no clause of size one. We also assume that at least one clause does not have size three, otherwise we simply keep all the variables and clauses, and we are done.

We keep all the variables x_i, $1 \leq i \leq n$.

We create three new variables α, β, γ and seven clauses $\{\alpha, \beta, \gamma\}$, $\{\alpha, \beta, \overline{\gamma}\}$, $\{\alpha, \overline{\beta}, \gamma\}$, $\{\alpha, \overline{\beta}, \overline{\gamma}\}$, $\{\overline{\alpha}, \beta, \gamma\}$, $\{\overline{\alpha}, \beta, \overline{\gamma}\}$, and $\{\overline{\alpha}, \overline{\beta}, \gamma\}$. Note that these clauses can be simultaneously satisfied if and only if $\alpha = \beta = \gamma = \mathrm{T}$. Only α will be used in the sequel, the other two variables only help to have clauses of size three.

Let c_j be any clause of the instance we start from, $1 \leq j \leq m$; c_j has size κ_j, $\kappa_j \geq 2$, but for simplicity we use $\kappa_j = \kappa$, and c_j reads $c_j = \{\ell_{j,1}, \ldots, \ell_{j,\kappa}\}$.

(a) If $\kappa = 2$, we create the clause $\{\ell_{j,1}, \ell_{j,2}, \overline{\alpha}\}$;

(b) If $\kappa = 3$, we keep the clause $\{\ell_{j,1}, \ell_{j,2}, \ell_{j,3}\}$;

(c) If $\kappa > 3$, we create a new variable $y_{c_j,1}$ and the clauses $\{y_{c_j,1}, \ell_{j,3}, \ldots, \ell_{j,\kappa}\}$, $\{y_{c_j,1}, \overline{\ell}_{j,1}, \overline{\alpha}\}$, $\{y_{c_j,1}, \overline{\ell}_{j,2}, \overline{\alpha}\}$, and $\{\overline{y}_{c_j,1}, \ell_{j,1}, \ell_{j,2}\}$. The first clause has size $\kappa - 1$, and we iterate the process until it has size three; at each step we create one variable and four clauses, three of them (at least) with size three. At the end, we keep only the clauses of size three. In this step, we have created $\kappa - 3$ new variables.

Example. $\kappa = 5$ *and* $c_j = \{x_{j,1}, \overline{x}_{j,2}, \overline{x}_{j,3}, x_{j,4}, x_{j,5}\}$. *First, we create* $\{y_{c_j,1}, \overline{x}_{j,3}, x_{j,4}, x_{j,5}\}$, *which we will not keep, and* $\{y_{c_j,1}, \overline{x}_{j,1}, \overline{\alpha}\}, \{y_{c_j,1}, x_{j,2}, \overline{\alpha}\}, \{\overline{y}_{c_j,1}, x_{j,1}, \overline{x}_{j,2}\}$. *Then we create* $\{y_{c_j,2}, x_{j,4}, x_{j,5}\}, \{y_{c_j,2}, \overline{y}_{c_j,1}, \overline{\alpha}\}, \{y_{c_j,2}, x_{j,3}, \overline{\alpha}\}, \{\overline{y}_{c_j,2}, y_{c_j,1}, \overline{x}_{j,3}\}$.

(1) Assume first that there is a unique way of satisfying all the clauses of the instance of U-SAT. We keep the same truth assignment for the variables x_i, $1 \leq i \leq n$, we set $\alpha = \beta = \gamma = \mathrm{T}$, and for all $j \in \{1, \ldots, m\}$, we set $y_{c_j,1} = \mathrm{T}$ if and only if at least one of $\ell_{j,1}$ and $\ell_{j,2}$ is equal to TRUE, $y_{c_j,2} = \mathrm{T}$ if and only if at least one of $y_{c_j,1}$ and $\ell_{j,3}$ is equal to TRUE, and so on. It is quite straightforward to check that this new assignment satisfies the set of clauses we have just constructed for U-3-SAT.

Is it the only way? Assume on the contrary that we have two assignments, $\mathcal{A}_1$ and $\mathcal{A}_2$, satisfying the instance of U-3-SAT.

First, note that any assignment $\mathcal{A}$ that satisfies the instance of U-3-SAT also satisifies, when restricted to the variables x_i, the instance of U-SAT. Indeed,

(a) Consider a clause of size two $c_j = \{\ell_{j,1}, \ell_{j,2}\}$; because necessarily $\mathcal{A}(\alpha) = \mathrm{T}$, if $\{\ell_{j,1}, \ell_{j,2}, \overline{\alpha}\}$ is satisfied by the values given to $\ell_{j,1}$ and $\ell_{j,2}$,

then this is also true for c_j;

(b) The clauses of size three in U-SAT need no explanation;

(c) Assume on the contrary that for $\kappa > 3$, the clause $c_j = \{\ell_{j,1}, \ldots, \ell_{j,\kappa}\}$ in U-SAT is not satisfied. Then, because the clause $\{\overline{y}_{c_j,1}, \ell_{j,1}, \ell_{j,2}\}$ in U-3-SAT is satisfied, we have $\mathcal{A}(y_{c_j,1}) = \mathrm{F}$, then similarly, because of the clause $\{\overline{y}_{c_j,2}, y_{c_j,1}, \ell_{j,3}\}$ in U-3-SAT, we have $\mathcal{A}(y_{c_j,2}) = \mathrm{F}$, and so on until the last new variable of type y, $\mathcal{A}(y_{c_j,\kappa-3}) = \mathrm{F}$, but then the clause $\{y_{c_j,\kappa-3}, \ell_{j,\kappa-1}, \ell_{j,\kappa}\}$ in U-3-SAT is not satisfied, a contradiction.

Therefore, the two assignments $\mathcal{A}_1$ and $\mathcal{A}_2$ give the same values to the variables x_i, $1 \leq i \leq n$, for otherwise we would have two ways of satisfying the instance of U-SAT, which would contradict our assumption on uniqueness. Since the values of α, β, γ are forced, $\mathcal{A}_1$ and $\mathcal{A}_2$ can differ only on the variables of type y.

Suppose first that they differ on $y_{c_j,1}$ for some j, with, say, $\mathcal{A}_1(y_{c_j,1}) = \mathrm{T}$ and $\mathcal{A}_2(y_{c_j,1}) = \mathrm{F}$. Then the three clauses $\{y_{c_j,1}, \overline{\ell}_{j,1}, \overline{\alpha}\}$, $\{y_{c_j,1}, \overline{\ell}_{j,2}, \overline{\alpha}\}$, $\{\overline{y}_{c_j,1}, \ell_{j,1}, \ell_{j,2}\}$, together with the fact that we have $\mathcal{A}_1(\alpha) = \mathcal{A}_2(\alpha) = \mathrm{T}$, show that $\mathcal{A}_2(\ell_{j,1}) = \mathcal{A}_2(\ell_{j,2}) = \mathrm{F}$, and $\mathcal{A}_1(\ell_{j,1}) = \mathrm{T}$ or $\mathcal{A}_1(\ell_{j,2}) = \mathrm{T}$, i.e., $\mathcal{A}_1$ and $\mathcal{A}_2$ differ on $\ell_{j,1}$ or $\ell_{j,2}$, a contradiction. So $\mathcal{A}_1(y_{c_j,1}) = \mathcal{A}_2(y_{c_j,1})$. Step by step, we can then show that $\mathcal{A}_1(y_{c_j,2}) = \mathcal{A}_2(y_{c_j,2})$, and so on until $\mathcal{A}_1(y_{c_j,\kappa-3}) = \mathcal{A}_2(y_{c_j,\kappa-3})$, so that finally $\mathcal{A}_1 = \mathcal{A}_2$ and there is only one solution for the instance of U-3-SAT.

(2) If the answer to the instance of U-SAT is NO, then either there are at least two solutions, which give at least two solutions to U-3-SAT, as previously seen, or there is no solution to U-SAT, in which case there is no solution to U-3-SAT, for otherwise one solution to U-3-SAT would also give, by restriction to the variables x_i, one solution to U-SAT. Thus the answer to the instance of U-3-SAT is also NO. △

Proposition 6 *There exists a polynomial reduction from U-3-SAT to U-1-3-SAT: U-3-SAT* $\rightarrow_p$ *U-1-3-SAT.*

Proof. The reduction is the one described in Section 1.4. Indeed, one can observe that this construction is more than just a reduction from 3-SAT to 1-3-SAT: in the Truth Table, once ℓ_1, ℓ_2, ℓ_3 are given, there is *only one way* to give a truth assignment to the new six variables in such a way that the new five clauses are 1-3-satisfied. First, note that $d_i = \overline{\ell}_3$ and that, once a_i, b_i, d_i and e_i have been assigned a value, then f_i and g_i are forced. Let us consider just one example, the other lines not being more difficult: if $\ell_1 = \ell_2 = \mathrm{F}$ and $\ell_3 = \mathrm{T}$ (line 7 of the Table), then $d_i = \mathrm{F}$ and, assuming that $a_i = \mathrm{F}$, the rest is forced: $e_i = \mathrm{T}$ and $b_i = \mathrm{F}$; but if $a_i = \mathrm{T}$, then $e_i = \mathrm{F}$, $b_i = \mathrm{T}$, and $c_{i,3}$ contains at least two true literals.

So, to any solution that 1-3-satisfies the five clauses $c_{i,j}$ we can associate a solution that satisfies c_i, and conversely; in other words, there are seven solutions that 1-3-satisfy the five clauses $c_{i,j}$ and there are seven solutions that satisfy c_i.

The conservation of the number of solutions is still verified if we consider the collections of clauses of the two instances, and the following has been proved: the number of solutions for the instance of 3-SAT we started from is equal to the number of solutions for the constructed instance of 1-3-SAT. In particular, there is a unique solution to the instance of 3-SAT if and only if there is a unique solution to the instance of 1-3-SAT, i.e., the answer to the initial instance, now seen as an instance of U-3-SAT, is YES if and only if the answer to the corresponding instance for U-1-3-SAT, is YES. △

Proposition 7 *There exists a polynomial reduction from U-1-3-SAT to U-3-SAT: U-1-3-SAT* $\rightarrow_p$ *U-3-SAT.*

Proof. We keep all the variables x_i, $1 \leq i \leq n$, of the instance of U-1-3-SAT.

We create three new variables α, β, γ and seven clauses $\{\alpha, \beta, \gamma\}$, $\{\alpha, \beta, \overline{\gamma}\}$, $\{\alpha, \overline{\beta}, \gamma\}$, $\{\alpha, \overline{\beta}, \overline{\gamma}\}$, $\{\overline{\alpha}, \beta, \gamma\}$, $\{\overline{\alpha}, \beta, \overline{\gamma}\}$, and $\{\overline{\alpha}, \overline{\beta}, \gamma\}$. As remarked previously, these clauses can be simultaneously satisfied if and only if $\alpha = \beta = \gamma = \mathrm{T}$; only α will be used in the sequel.

Let c_j be any clause of the instance we start from, $1 \leq j \leq m$: $c_j = \{\ell_1, \ell_2, \ell_3\}$. We keep c_j and add three new clauses, $\{\overline{\ell}_1, \overline{\ell}_2, \overline{\alpha}\}$, $\{\overline{\ell}_1, \overline{\ell}_3, \overline{\alpha}\}$, $\{\overline{\ell}_2, \overline{\ell}_3, \overline{\alpha}\}$.

Assume first that there is exactly one truth assignment, $\mathcal{A}_1$, 1-3-satisfying the instance of U-1-3-SAT. Keeping the same assignment for the variables x_i and setting $\mathcal{A}_1(\alpha) = \mathcal{A}_1(\beta) = \mathcal{A}_1(\gamma) = \mathrm{T}$, we obviously satisfy all the clauses for U-3-SAT. Is it possible to have a second assignment, $\mathcal{A}_2$, that satisfies the instance of U-3-SAT? If yes, necessarily $\mathcal{A}_2(\alpha) = \mathcal{A}_2(\beta) = \mathcal{A}_2(\gamma) = \mathrm{T}$; then, because of the clauses $\{\overline{\ell}_1, \overline{\ell}_2, \overline{\alpha}\}$, $\{\overline{\ell}_1, \overline{\ell}_3, \overline{\alpha}\}$, $\{\overline{\ell}_2, \overline{\ell}_3, \overline{\alpha}\}$, at most one of the three literals ℓ_1, ℓ_2, ℓ_3 can be set true by $\mathcal{A}_2$, but also at least one, because of the clause $\{\ell_1, \ell_2, \ell_3\}$. So $\mathcal{A}_2$, when restricted to the variables x_i, also 1-3-satisfies the instance of U-1-3-SAT, and $\mathcal{A}_2 \neq \mathcal{A}_1$ is impossible.

If there is no assignment 1-3-satisfying the instance of U-1-3-SAT, then there is no assignment satisfying the instance of U-3-SAT, since this assignment, when restricted to the variables x_i, would be an assignment 1-3-satisfying the original instance, as we have already seen. If there is more than one assignment 1-3-satisfying the instance of U-1-3-SAT, then obviously there is more than one assignment satisfying the instance of U-3-SAT.

So there is a YES answer for the instance of U-1-3-SAT if and only if there is a YES answer for the corresponding instance of U-3-SAT. △

Proposition 8 *For every integer $k \geq 4$, there exists a polynomial reduction from U-$(k-1)$-SAT to U-k-SAT: U-$(k-1)$-SAT $\rightarrow_p$ U-k-SAT.*

Proof. We consider an instance of U-$(k-1)$-SAT with n variables x_i and m clauses c_j, of size $k-1$.

We keep all the variables x_i, $1 \leq i \leq n$.

We introduce k new variables $\alpha_1, \ldots, \alpha_k$ and $2^k - 1$ clauses: these are all the clauses of size k containing either α_1 or $\overline{\alpha}_1$, ..., α_k or $\overline{\alpha}_k$, with the exception of $\{\overline{\alpha}_1, \ldots, \overline{\alpha}_k\}$; since k is fixed, the number of clauses is a constant, so this is polynomial wrt the size of the instance we start from. Note that these clauses can be simultaneously satisfied if and only if $\alpha_1 = \ldots = \alpha_k = \mathrm{T}$. When building other clauses of size k, we shall use only α_1; the aim of the other variables $\alpha_2, \ldots, \alpha_k$ is simply to have clauses of the right size.

The reduction is simple: besides the $2^k - 1$ aforementioned clauses, each clause $c_j = \{\ell_1, \ldots, \ell_{k-1}\}$ in the instance of U-$(k-1)$-SAT is transformed into $\{\ell_1, \ldots, \ell_{k-1}, \overline{\alpha}_1\}$. It is now easy to check that there is a YES answer to the instance of U-$(k-1)$-SAT if and only if there is a YES answer to the resulting instance of U-k-SAT. △

The above reduction also works for $k = 3$, but U-2-SAT $\rightarrow_p$ U-3-SAT gives no useful information, since U-2-SAT is polynomial (Theorem 4).

Proposition 9 *For every integer $k \geq 3$, there exists a polynomial reduction from U-k-SAT to U-SAT: U-k-SAT $\rightarrow_p$ U-SAT.*

Proof. Simply use the identity reduction. △

The above reduction also works for $k = 2$, but U-2-SAT $\rightarrow_p$ U-SAT gives no useful information, since U-2-SAT is polynomial.

We have now proved that all our satisfiability problems are equivalent:

Theorem 10 *For every integer $k \geq 3$, the problems U-SAT, U-k-SAT and U-1-3-SAT are equivalent, up to polynomials.*

Proof. Because the chain of polynomial reductions described at the beginning of Section 2.2 has been proved, by Theorem 5 and Propositions 6–9. △

2.3 Location of U-Satisfiability Problems

Now that we know that they are equivalent, we can use previous works to show that the three problems U-SAT, U-k-SAT and U-1-3-SAT are *NP*-hard, and that they belong to *DP*.

Proposition 11 *There exists a Turing reduction from* 3-*SAT to co*-3-*SAT:* 3-*SAT* $\rightarrow_T$ *co*-3-*SAT.*

Proof. Simply use the identity reduction; the answers YES and NO are permuted. △

Proposition 12 *[8] There exists a polynomial reduction from co-SAT to U-SAT: co-SAT* $\rightarrow_p$ *U-SAT.*

Proof. We give a proof which is slightly simplified compared to [8].

We consider an instance of co-SAT with n variables x_i and a set $\mathcal{C}$ of m clauses c_j, with at least one of them containing only negated variables (cf. Remark 2). We write $\mathcal{C}$ in its normal conjunctive form like in (1): $\mathcal{C} = \wedge_{1 \leq j \leq m} c_j$, and, instead of constructing clauses for the instance of U-SAT, we construct the logical formula

$$\mathcal{C}^* = \mathcal{C} \vee (\wedge_{1 \leq i \leq n} x_i),$$

which could, through logical equivalences, be led back to its normal conjunctive form as in (1); now the instance of U-SAT simply consists of $\mathcal{C}^*$.

If the answer to the instance of co-SAT is YES, i.e., no assignment satisfies $\mathcal{C}$, then obviously the *only* way to satisfy $\mathcal{C}^*$ is to set TRUE every x_i, so the answer to U-SAT is YES.

If the answer to co-SAT is NO, then there are at least two ways of satisfying $\mathcal{C}^*$:

(a) use an assignment that satisfies $\mathcal{C}$;

(b) set $x_i =$ TRUE for all $i \in \{1, \ldots, n\}$.

The assignment of the variables x_i in (b) cannot be the same as in (a), thanks to the assumption that at least one clause in $\mathcal{C}$ contains only negated variables. So the answer to U-SAT is NO. △

Corollary 13 *For every integer* $k \geq 3$*, the decision problems U-SAT, U-k-SAT and U*-1-3-*SAT are NP-hard (and co-NP-hard by Remark 1).*

Proof. By Proposition 11, co-3-SAT is *NP*-hard, which implies that co-SAT also is, as well as U-SAT, by Proposition 12; then both U-k-SAT, $k \geq 3$, and U-1-3-SAT are *NP*-hard, by Theorem 10. △

Proposition 14 *For every integer* $k \geq 3$*, the decision problems U-SAT, U-k-SAT and U*-1-3-*SAT belong to the class DP.*

Proof. The result for U-SAT is stated in [45, p. 415] and is easy to prove. We have to exhibit two languages, $L_1 \in NP$ and $L_2 \in$ co-*NP*, such that

the set of YES instances of U-SAT is $L_1 \cap L_2$. Take $L_1 = \{$formulas (or sets of clauses) such that there is at least one truth assignment satisfying them$\}$ and $L_2 = \{$formulas (or sets of clauses) such that there is at most one assignment satisfying them$\}$. This proof can be applied *mutatis mutandis* to the other two problems; alternatively, the equivalence between the three problems can be used. $\triangle$

Remark 15 *In [45], it is also stated that "U-SAT is not believed to be DP-complete".*

3 Uniqueness of Solutions: Colouring

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 on the values of a k-colouring f also is a k-colouring. Now, rather than saying that we look for a unique colouring up to permutations, it is more convenient to consider the problem stated in terms of IS-partition. Also, if the graph is not connected and not trivial, the answer will be NO. Thus, the problems that we shall study below (and still abusively name U-k-COL and U-COL) will be the following:

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?

Note that, like 2-COL, U-2-COL is, quite obviously, polynomial. In Section 3.3, we shall deal with U-OCOL.

3.1 Preliminary Results on 3-Colourings

The following lemma is somehow inspired by the proof of the *NP*-completeness of 3-COL, see Theorem 2.1 and Figure 1 in [22]. Note that the proof from [22] cannot convey uniqueness, even up to permutations of colours.

Lemma 16 *Consider the graph $G_0 = (V_0, E_0)$ described by Figure 4.*

(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.

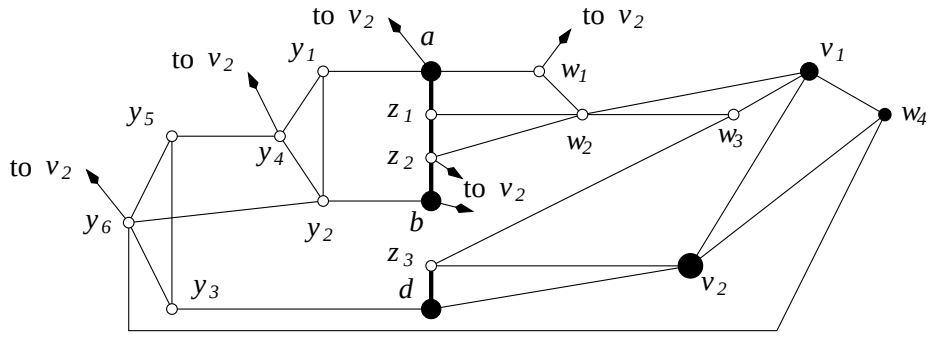

Figure 4: The graph G_0 of Lemma 16. Because of their particular roles, the vertices v_1, v_2, w_4, a, b and d are represented by black circles.

(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. Wlog, 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.

When $a \in S_1$, $b \in S_2$, $d \in S_1$, 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 when $a \in S_2$, $b \in S_1$, $d \in S_2$ is the same, with the roles of S_1 and S_2 permuted.

When $a \in S_2$, $b \in S_1$, $d \in S_1$, 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 when $a \in S_1$, $b \in S_2$, $d \in S_2$ is the same, with the roles of S_1 and S_2

permuted. △

Remark 17 *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).

3.2 Equivalence of Uniqueness of Colouring and of Satisfiability

We are going to prove the polynomial reductions
U-1-3-SAT $\rightarrow_p$ U-3-COL (Theorem 18),
for $k \geq 3$, U-3-COL $\rightarrow_p$ U-k-COL $\rightarrow_p$ U-COL (Proposition 19),
and U-COL $\rightarrow_p$ U-SAT (Theorem 20).

Theorem 18 *There exists a polynomial reduction from U-1-3-SAT to U-3-COL: U-1-3-SAT $\rightarrow_p$ U-3-COL.*

Proof. A possible polynomial reduction is the following. If the instance of U-1-3-SAT consists of a set $\mathcal{C}$ of m clauses over n variables x_i $(1 \leq i \leq n)$, then the vertex set W of the graph G we are constructing is

$$W = \{x_i, \overline{x}_i : 1 \leq i \leq n\} \cup \{v_1, v_2\} \cup \{z_{i,j} : 1 \leq i \leq 3, 1 \leq j \leq m\} \cup$$

$$\{w_{i,j} : 1 \leq i \leq 4, 1 \leq j \leq m\} \cup \{y_{i,j} : 1 \leq i \leq 6, 1 \leq j \leq m\}.$$

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

$$V_j^- = \{z_{i,j} : 1 \leq i \leq 3\} \cup \{w_{i,j} : 1 \leq i \leq 4\} \cup \{y_{i,j} : 1 \leq i \leq 6\},$$

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

For each clause $c_j = \{a_j, b_j, d_j\}$ where a_j, b_j and d_j belong to $\{x_i, \overline{x}_i : 1 \leq i \leq n\}$, 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 $v_{1,j}$, $1 \leq j \leq m$, 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

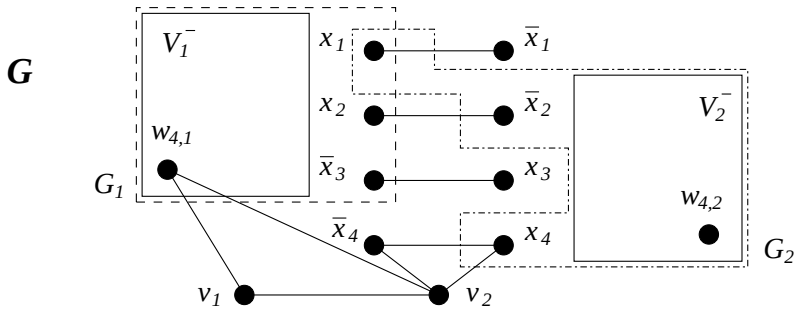

Figure 5: A small example for Theorem 18: $c_1 = \{x_1, x_2, \overline{x}_3\}$, $c_2 = \{x_1, \overline{x}_2, x_4\}$. Not all vertices and edges are represented.

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 $\overline{x}_iv_2$, $1 \leq i \leq n$, and we have our graph G; see Figure 5 for a small example.

(a) 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 16, 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, wlog, 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, $1 \leq i \leq n$, by setting $\mathcal{A}(x_i) = \mathrm{T}$ if $x_i \in S_2^*$ and $\mathcal{A}(x_i) = \mathrm{F}$ 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 16(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 17, $w_{4,j_0} \in S_1^*$ and $v_1 \in S_2^*$. This in turn implies that *all* the vertices $w_{4,j}$, $1 \leq j \leq m$, 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.

(b) 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. △

Proposition 19 *For every integer $k \geq 3$, there exists a polynomial reduction from U-3-COL to U-k-COL: U-3-COL $\rightarrow_p$ U-k-COL, and from U-k-COL to U-COL: U-k-COL $\rightarrow_p$ U-COL.*

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

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. △

Theorem 20 *There exists a polynomial reduction from U-COL to U-SAT: U-COL $\rightarrow_p$ U-SAT.*

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

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

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

(ii) for $1 \leq h \leq |V|$ and $1 \leq i < j \leq k$, 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 $2 \leq h \leq |V|$ and $1 \leq i \leq k$, clauses $c_i^h = \{\overline{x}_i^h, x_i^1, \ldots, x_i^{h-1}, x_{i-1}^1, \ldots, x_{i-1}^{h-1}\}$.

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.

Note that the number of variables and clauses is polynomial wrt the order of G, in particular because we may take $k \leq |V|$.

(1) We assume that there is a k-IS-partition of V into k sets S_1, S_2, ..., 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^*; ...; 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, ..., 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, ..., 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, $2 \leq h \leq |V|$, $1 \leq i \leq k$, and assume that it is *not* satisfied by $\mathcal{A}_1$. 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 $\overline{x}_i^h, \overline{x}_j^h$ 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 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 $x_i^h, x_i^{h'}$ 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 $\mathcal{A}_2(x_1^1) = \mathrm{T}$, 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

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

Now $\mathcal{A}_2(x_{p+1}^{q_1}) = \mathrm{T}$, 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. △

Theorem 21 *For every integer* $k \geq 3$*, the problems U-SAT, U-*1*-3-SAT, U-k-SAT, U-k-COL and U-COL have equivalent complexities, up to polynomials. All are NP-hard and belong to the class DP.* △

Note that, using the same argument as in the proof of Proposition 14, it could have been shown directly that U-k-COL and U-COL belong to *DP*.

3.3 Location of U-OCOL

Obvioulsy, U-OCOL is *NP*-hard. We provide an algorithm which answers YES or NO to U-OCOL, thus giving an upper bound on its complexity, and allowing to locate it, still in a fuzzy way, in the class hierarchy.

Proposition 22 *The problem U-OCOL belongs to the class* L^{NP}.

Proof. Using a standard dichotomous process, with a logarithmic number of calls to an algorithm solving k-COL, which is in *NP*, we can build an algorithm outputting the chromatic number of G, for any graph G.

Once we have computed the chromatic number, we question the instance G of U-k-COL, with k equal to the chromatic number, and we get the answer to U-OCOL.

Since U-k-COL belongs to *DP* and $DP \subseteq L^{NP}$, all in all we obtain an algorithm solving U-OCOL with a logarithmic number of calls to an algorithm solving a problem in *NP*. △

Note that using an algorithm for U-k-COL without knowing the chromatic number leads nowhere, because a NO answer cannot be interpreted: 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.

4 Conclusion

Theorem 21 states that the five problems, U-SAT, U-1-3-SAT, U-k-SAT, U-k-COL and U-COL, are equivalent and lie somewhere in the vertically hatched area of Figure 6, but probably not in *DP*-*C*, cf. Remark 15.

As for the problem U-OCOL, we have a poorer result (Proposition 22): it lies within the areas that are hatched horizontally or vertically.

Finally, U-2-SAT (Theorem 4) and U-2-COL (trivially) are polynomial.

In [8], the authors wonder whether
• (A) U-SAT is *NP*-hard, but here what they mean is: does there exist a *polynomial* reduction from an *NP*-complete problem to U-SAT ? i.e., they use the *second* definition of *NP*-hardness.

They show that (A) is true if and only if
• (B) U-SAT is *DP*-complete.

So, if one is careless and considers that U-SAT is *NP*-hard without checking according to which definition, one might easily jump too hastily to the conclusion that U-SAT is *DP*-complete, which, to our knowledge, is not known to be true or not.

As for U-3-SAT, we do not know where to locate it more precisely either; in [10] the problems U-k-SAT and, more particularly, U-3-SAT are studied, but it appears that they are versions where the given set of clauses has zero or one solution, which makes quite a difference with our problem.

Open problem(s). Give a better location for U-SAT, U-k-SAT, U-1-3-SAT, U-k-COL, U-COL and U-OCOL, in the classes of complexity.

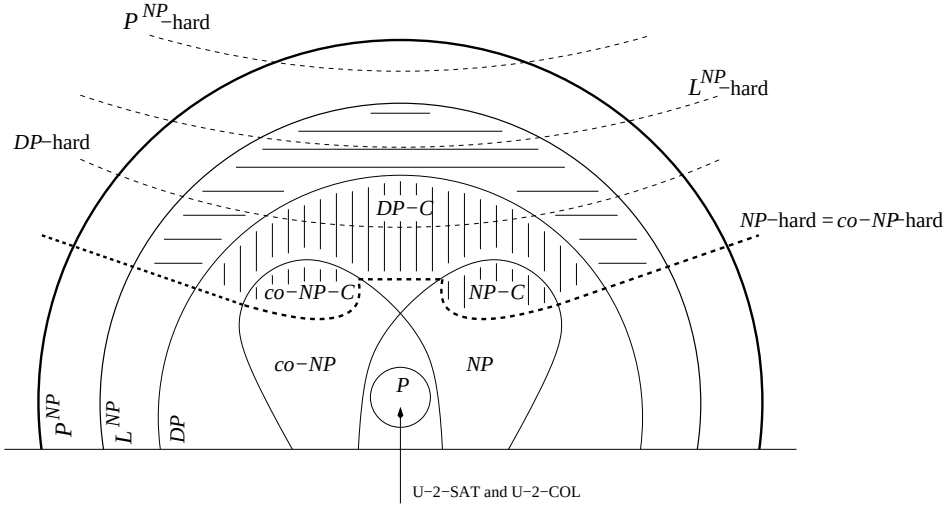

Figure 6: Some classes of complexity: Figure 1 revisited.

Complexity of Unique (Optimal) Solutions in Graphs: Vertex Cover and Domination

Olivier Hudry
LTCI, Télécom ParisTech, Université Paris-Saclay
46 rue Barrault, 75634 Paris Cedex 13 - France

& Antoine Lobstein
Centre National de la Recherche Scientifique
Laboratoire de Recherche en Informatique, UMR 8623,
Université Paris-Sud, Université Paris-Saclay
Bâtiment 650 Ada Lovelace, 91405 Orsay Cedex - France

olivier.hudry@telecom-paristech.fr
antoine.lobstein@lri.fr

Abstract

We study the complexity of four decision problems dealing with the *uniqueness* of a solution in a graph: "Uniqueness of a Vertex Cover with bounded size" (U-VC) and "Uniqueness of an Optimal Vertex Cover" (U-OVC), and for any fixed integer $r \geq 1$, "Uniqueness of an r-Dominating Code with bounded size" (U-DC$_r$) and "Uniqueness of an Optimal r-Dominating Code" (U-ODC$_r$). In particular, we give a polynomial reduction from "Unique Satisfiability of a Boolean formula" (U-SAT) to U-OVC, and from U-SAT to U-ODC$_r$. We prove that U-VC and U-DC$_r$ have complexity equivalent to that of U-SAT (up to polynomials); consequently, these problems are all *NP*-hard, and U-VC and U-DC$_r$ belong to the class *DP*.

Key Words: Graph Theory, Complexity Theory, *NP*-Hardness, Decision Problems, Polynomial Reduction, Uniqueness of (Optimal) Solution, Domination, Dominating Codes, Vertex Covers, Boolean Satisfiability Problems

5 Introduction

5.1 The Vertex Cover and Domination Problems

For the vast topic of 1-domination in graphs, see [25].

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, $|V|$. In a connected graph G, we can define the *distance* between any two vertices x and y, denoted by $d_G(x, y)$, as the length of any shortest path between x and y. This definition can be extended to disconnected graphs, using the convention that $d_G(x, y) = +\infty$ if no path exists between x and y. The subscript G can be dropped when there is no ambiguity.

Given a graph $G = (V, E)$, an *independent set*, or *stable set*, is a subset $V^* \subseteq V$ such that for all $u \in V^*$, $v \in V^*$, we have $uv \notin E$. A *clique*, or *complete graph*, is any subgraph $(V^- \subseteq V, E^- \subseteq E)$ such that for all $u \in V^-$, $v \in V^-$, $u \neq v$, we have $uv \in E^-$. For an integer $r \geq 1$, the *r-th power* of G is the graph $G^r = (V, E^r)$, with $E^r = \{uv : u \in V, v \in V, d_G(u, v) \leq r\}$.

A *vertex cover* of G (VC for short) is a subset of vertices $V^* \subseteq V$ such that for every edge $e = uv \in E$, $V^* \cap \{u, v\} \neq \emptyset$. We denote by $\phi(G)$ the smallest cardinality of a VC of G, and call it the *vertex cover number* of G; any VC V^* with $|V^*| = \phi(G)$ is said to be *optimal*.

For any vertex $v \in V$, the *open neighbourhood* $N(v)$ *of* v consists of the set of vertices adjacent to v, i.e., $N(v) = \{u \in V : uv \in E\}$; the *closed neighbourhood of* v is $B_1(v) = N[v] = N(v) \cup \{v\}$. This notation can be generalized to any integer $r \geq 0$ by setting

$$B_r(v) = \{x \in V : d(x, v) \leq r\}.$$

For $X \subseteq V$, we denote by $B_r(X)$ the set of vertices within distance r from X:

$$B_r(X) = \cup_{x \in X} B_r(x).$$

Whenever two vertices x and y are such that $x \in B_r(y)$ (which is equivalent to $y \in B_r(x)$), we say that x and y *r-dominate* each other; note that every vertex r-dominates itself. A set W is said to r-dominate a set Z if every vertex in Z is r-dominated by at least one vertex of W, or equivalently: $Z \subseteq B_r(W)$.

A *code* C is simply a subset of V, and its elements are called *codewords*.

We say that C is an *r-dominating code* in G if all the sets $B_r(v) \cap C$, $v \in V$, are nonempty; in other words, every vertex is r-dominated by C, or

$V = B_r(C)$. We denote by $\gamma_r(G)$ the smallest cardinality of an r-dominating code in G, and call it the *r-domination number* of G; any r-dominating code C with $|C| = \gamma_r(G)$ is said to be *optimal*. The following result needs no proof.

Lemma 23 *For all $r \geq 1$, the code C is r-dominating in G if and only if it is 1-dominating in the r-th power of G.* △

The following two problems are well known in graph theory as well as in complexity theory, specially when $r = 1$ for the second problem, stated here for any fixed integer $r \geq 1$.

Problem VC (Vertex Cover with bounded size):
Instance: A graph G and an integer k.
Question: Does G admit a vertex cover of size at most k?

Problem DC_r (r-Dominating Code with bounded size):
Instance: A graph G and an integer k.
Question: Does G admit an r-dominating code of size at most k?

As we shall see, these problems are *NP*-complete (Propositions 29 from [39], [21] and 39 from [21], [33]). In this paper, we wish to locate, in the hierarchy of complexity classes, the following four problems, dealing with the *uniqueness* of solutions, optimal or not.

Problem U-VC (Unique Vertex Cover with bounded size):
Instance: A graph G and an integer k.
Question: Does G admit a *unique* vertex cover of size at most k?

Problem U-OVC (Unique Optimal Vertex Cover):
Instance: A graph G.
Question: Does G admit a *unique optimal* vertex cover?

Problem U-DC_r (Unique r-Dominating Code with bounded size):
Instance: A graph G and an integer k.
Question: Does G admit a *unique* r-dominating code of size at most k?

Problem U-ODC_r (Unique Optimal r-Dominating Code):
Instance: A graph G.
Question: Does G admit a *unique optimal* r-dominating code?

In Sections 6 and 7, we establish our results on vertex covers and dominating codes, respectively; we prove in particular that there is a polynomial reduction from "Unique Satisfiability of a Boolean formula" (U-SAT) to U-OVC, and from U-SAT to U-ODC_r; and that U-VC and U-DC_r are equivalent to U-SAT (up to polynomials). This implies that:

– U-VC, U-OVC, UDC$_r$ and U-ODC$_r$ ($r \geq 1$) are *NP*-hard;
– U-VC and UDC$_r$ ($r \geq 1$) belong to the class *DP*.
We also show that U-OVC and U-ODC$_r$ ($r \geq 1$) belong to the class L^{NP}.

In forthcoming papers, we likewise investigate the issue of the uniqueness of solutions for (a) Boolean satisfiability and graph colouring [34] *(= Sections 1–4 in this Report)*, of which we shall use some of the results in the present paper; (b) r-Identifying Code and r-Locating-Dominating Code [36] *(Sections 9–13)*; (c) Hamiltonian Cycle [37] *(Sections 14–16)*.

In [33], we already investigated the complexity of the existence of, and of the search for, optimal r-dominating codes, as well as optimal r-dominating codes containing a given subset of vertices; some results will be re-used here, see, e.g., Lemma 38.

For other works in this area, see [26] for vertex covers, and [20], [31], [42] and [43] for some problems related to domination in the binary hypercube.

In the sequel, we shall also need the following tools, which constitute classical definitions and decision problems, related to Boolean satisfiability. We consider a set $\mathcal{X}$ of n *Boolean variables* x_i and a set $\mathcal{C}$ of m *clauses* ($\mathcal{C}$ is also called a *Boolean formula*); each clause c_j contains κ_j *literals*, a literal being a variable x_i or its complement $\overline{x}_i$. 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 $\mathcal{X}$, each clause containing at least two different literals.
Question: Is there a truth assignment for $\mathcal{X}$ that satisfies $\mathcal{C}$?

The following problem is stated for any fixed integer $k \geq 2$.

Problem k-SAT (k-Satisfiability):
Instance: A set $\mathcal{X}$ of variables, a collection $\mathcal{C}$ of clauses over $\mathcal{X}$, each clause containing exactly k different literals.
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. We shall also consider the following variants of the above problems:

U-SAT (Unique Satisfiability),

U-k-SAT (Unique k-Satisfiability),

U-1-3-SAT (Unique One-in-Three Satisfiability).

They have the same instances as SAT, k-SAT and 1-3-SAT respectively, but now the question is "Is there a *unique* truth assignment...?".

We shall give in Proposition 25 and Corollary 26 what we need to know about the complexities of these problems. We now provide the necessary definitions and notation for complexity.

5.2 Necessary Notions in Complexity

We expound here, not too formally, the notions of complexity that will be needed in the sequel. We refer the reader to, e.g., [6], [21], [38] or [45] for more on this topic.

A *decision problem* is of the type "Given an instance I and a property $\mathcal{PR}$ on I, is $\mathcal{PR}$ true for I?", and has only two solutions, "yes" or "no". The class P will denote the set of problems which can be solved by a *polynomial* (time) algorithm, and the class 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 transformation 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 shall write $\pi_1 \rightarrow_p \pi_2$. Cook [14] proved that there is one problem in NP, namely SAT, to which every other problem in NP can be polynomially reduced. Thus, in a sense, SAT is the "hardest" problem inside NP. Other problems share this property in NP and are called *NP-complete* problems; their class is denoted by *NP-C*. The way to show that a decision problem π is NP-complete is, once it is proved to be in NP, to choose some NP-complete problem π_1 and to polynomially reduce it to π. From a practical viewpoint, the NP-completeness of a problem π implies that we do not know any polynomial algorithm solving π, and that, under the assumption $P \neq NP$, which is widely believed to be true, no such algorithm exists: the time required can grow exponentially with the size of the instance (when

the instance is a graph, its size is polynomially linked to the order of the graph).

The *complement* of a decision problem, "Given I and $\mathcal{PR}$, is $\mathcal{PR}$ true for I?", is "Given I and $\mathcal{PR}$, is $\mathcal{PR}$ false for I?". The class *co-NP* (respectively, *co-NP-C*) is the class of the problems which are the complement of a problem in *NP* (respectively, *NP-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. Thus, in this sense, π_2 is "at least as hard" as π_1. A problem π is *NP-hard* (respectively, *co-NP-hard*) if there is a Turing reduction from some *NP*-complete (respectively, co-*NP*-complete) problem to π [21, p. 113].

Remark 24 *Note that with these definitions, NP-hard and co-NP-hard coincide [21, p. 114].*

The notions of completeness and hardness can of course be extended to classes other than *NP* or co-*NP*. *NP*-hardness is defined differently in [15] and [27]: there, a problem π is *NP-hard* if there is a *polynomial* reduction from some *NP*-complete problem to π; this may lead to confusion (see Section 8).

We also introduce the classes P^{NP} (also known as Δ_2 in the class hierarchy) and L^{NP} (also denoted by $P^{NP[O(\log n)]}$ or Θ_2), which contain the decision problems which can be solved by applying, with a number of calls which is polynomial (respectively, logarithmic) with respect to the size of the instance, a subprogram able to solve an appropriate problem in *NP* (usually, an *NP*-complete problem); and the class *DP* [46] (or DIF^P [8] or BH_2 [38], [52], ...) as the class of languages (or problems) L such that there are two languages $L_1 \in NP$ and $L_2 \in$ co-*NP* satisfying $L = L_1 \cap L_2$. This class is not to be confused with $NP \cap$ co-*NP* (see the warning in, e.g., [45, p. 412]); actually, *DP* contains $NP \cup$ co-*NP* and is contained in L^{NP}. See Figure 7.

Membership to *P*, *NP*, co-*NP*, *DP*, L^{NP} or P^{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 *P=NP*, they would lose their interest. But we do not know whether or where the classes of complexity collapse.

We now consider the satisfiability problems defined in Section 5.1.

The problems SAT and 3-SAT are two of the basic and most well-known *NP*-complete problems [14], [21, p. 39, p. 46 and p. 259]. More generally,

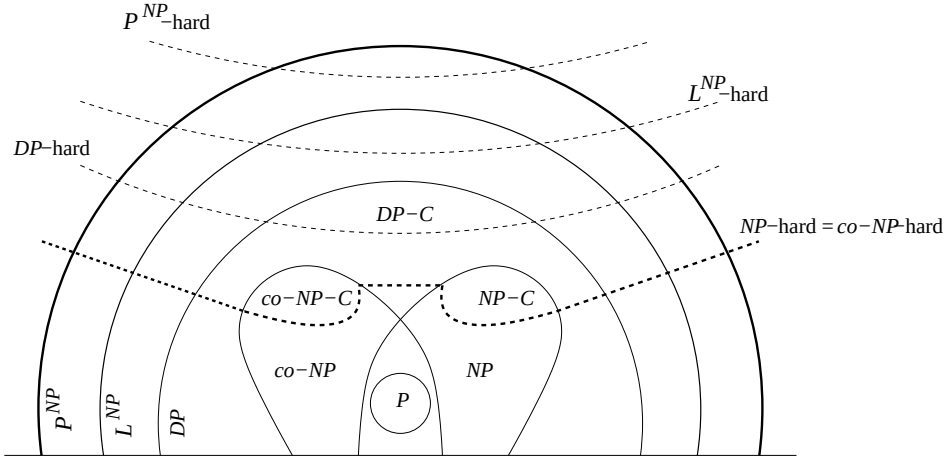

Figure 7: Some classes of complexity.

k-SAT is *NP*-complete for $k \geq 3$ and polynomial for $k = 2$. The problem 1-3-SAT, which is obvioulsy in *NP*, is also *NP*-complete [48, Lemma 3.5], [21, p. 259], *and Remark 3 in this Report.*

In [34] we proved the following *(= Theorem 10 in this Report).*

Proposition 25 *For every integer $k \geq 3$, the problems U-SAT, U-k-SAT and U-1-3-SAT are equivalent, up to polynomials.* △

Using results from [8] and [45, p. 415], it is then rather simple to obtain the following result.

Corollary 26 *For every integer $k \geq 3$,*

(a) the decision problems U-SAT, U-k-SAT and U-1-3-SAT are NP-hard (and co-NP-hard by Remark 24);

(b) the decision problems U-SAT, U-k-SAT and U-1-3-SAT belong to the class DP. △

Remark 27 *It is not known whether these problems are DP-complete. In [45, p. 415], it is said that "U-SAT is not believed to be DP-complete".*

We are now ready to investigate the problems of Vertex Cover and Domination.

6 Vertex Covers

In this section, we are going to describe three polynomial reductions:

U-1-3-SAT $\rightarrow_p$ U-VC and U-1-3-SAT $\rightarrow_p$ U-OVC (Theorem 30),

U-VC $\rightarrow_p$ U-SAT (Theorem 33).

The consequence of these reductions is that U-SAT and U-VC have equivalent complexity, that U-VC and U-OVC are *NP*-hard, and that U-VC belongs to *DP*. We shall also show that U-OVC belongs to the class L^{NP}.

6.1 Preliminary Results

The following result characterizes the vertices belonging to at least one optimal VC, through the comparison of two vertex cover numbers, and will be used in the (constructive) proof of Proposition 35.

Lemma 28 *Let $G = (V, E)$ be a graph. For a given vertex $\alpha \in V$, we consider the following graph: $G_\alpha = (V_\alpha, E_\alpha)$, with*

$$V_\alpha = V \cup \{\beta_1, \beta_2\},\ E_\alpha = E \cup \{\alpha\beta_1, \alpha\beta_2\},$$

where, for $1 \leq i \leq 2$, $\beta_i \notin V$. Then α belongs to at least one optimal vertex cover in G if and only if $\phi(G) = \phi(G_\alpha)$.

Proof. (a) Let α be a vertex belonging to at least one optimal vertex cover C in G; then C is a VC in G_α as well, and $\phi(G_\alpha) \leq |C| = \phi(G)$.

On the other hand, let C^* be an optimal VC in G_α. Then obviously $\alpha \in C^*$ and none of the β_i's belongs to C^*. Consequently, $C^* \subseteq V$, C^* is a VC in G, and $\phi(G) \leq |C^*| = \phi(G_\alpha)$.

Therefore, with this assumption on α, we have: $\phi(G) = \phi(G_\alpha)$.

(b) Conversely, assume that $\phi(G) = \phi(G_\alpha)$ for a vertex $\alpha \in V$. Let C^* be an optimal VC in G_α; again, $\alpha \in C^*$, and none of the β_i's belongs to C^*. Then C^* is a VC in G, it has size $\phi(G_\alpha) = \phi(G)$, and it contains α. △

Proposition 29 *[39], [21, p. 46 and p. 190] The decision problem VC is NP-complete.* △

Actually, we shall only use the fact that the problem VC belongs to *NP*, in the proofs of Propositions 35 and 36.

6.2 Uniqueness of Vertex Cover

6.2.1 From U-1-3-SAT to U-VC and U-OVC

Theorem 30 *There exists a polynomial reduction from U-1-3-SAT to U-VC and to U-OVC: U-1-3-SAT $\rightarrow_p$ U-VC and U-1-3-SAT $\rightarrow_p$ U-OVC.*

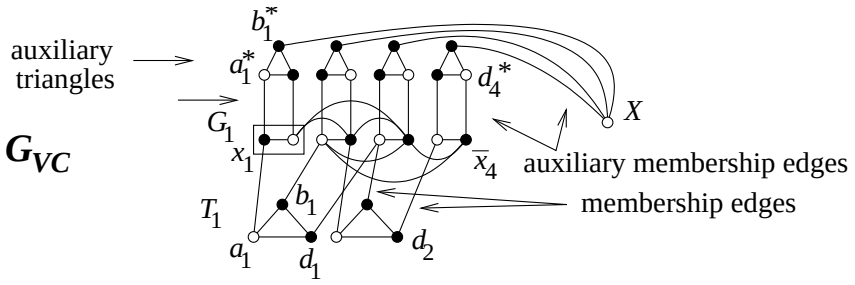

Figure 8: Illustration of the graph constructed for the reduction from U-1-3-SAT to U-OVC, with four variables and two clauses, $c_1 = \{x_1, x_2, x_3\}$, $c_2 = \{\overline{x}_2, x_3, x_4\}$. The sixteen black vertices form the (not unique) optimal vertex cover V^* corresponding to the (not unique) truth assignment $x_1 =$ T, $x_2 =$F, $x_3 =$F, $x_4 =$F 1-3-satisfying the clauses. As soon as we set $V^* \cap (V_1 \cup V_2 \cup V_3 \cup V_4) = \{x_1, \overline{x}_2, \overline{x}_3, \overline{x}_4\}$, the other vertices in V^* are forced.

Proof. Going deeper into the proof of the *NP*-completeness of the problem Vertex Cover (see [39], [21, pp. 54–56]), which uses a polynomial reduction from 3-SAT and obviously does not convey the uniqueness of the solution, we describe a polynomial reduction from the problem U-1-3-SAT to U-VC and U-OVC, see Figure 8. If the instance of U-1-3-SAT consists of a set $\mathcal{C}$ of m clauses over n variables, we construct

– one vertex denoted by X;

– for each clause c_j, a triangle $T_j = \{a_j, b_j, d_j\}$;

– for each variable x_i, a component $G_i = (V_i = \{x_i, \overline{x}_i\}, E_i = \{x_i\overline{x}_i\})$ and an auxiliary triangle $T_i^* = \{a_i^*, b_i^*, d_i^*\}$ whose vertices are linked to x_i, $\overline{x}_i$ and X by the three edges $x_i a_i^*$, $\overline{x}_i d_i^*$ and $b_i^* X$, called "auxiliary membership edges".

Then we link the components G_i on the one hand, and the triangles T_j on the other hand, according to which literals appear in which clauses ("membership edges"). For each clause $c_j = \{\ell_1, \ell_2, \ell_3\}$, we also add the triangular set of edges $E'_j = \{\overline{\ell}_1\overline{\ell}_2, \overline{\ell}_1\overline{\ell}_3, \overline{\ell}_2\overline{\ell}_3\}$.

The graph G thus constructed constitutes the instance of U-OVC, and, together with the integer $k = 2m + 3n$, the instance of U-VC. The order of G is $1 + 3m + 5n$.

Note that if V^* is a VC, then each triangle T_j and each auxiliary triangle T_i^* contain at least two vertices, each component G_i at least one vertex, and $|V^*| \geq 2(m + n) + n = 2m + 3n = k$; if $|V^*| = k$, then V^* is optimal, and each triangle contains exactly two vertices, each component G_i exactly

one vertex. We can also observe that, because of the edge sets E'_j, at least two vertices among $\overline{\ell}_1, \overline{\ell}_2, \overline{\ell}_3$ belong to any VC.

(a) 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 $\mathcal{C}$. Then, by taking, in each G_i, the vertex corresponding to the literal which is TRUE, in every triangle T_j, the two vertices which are linked to the two false literals of c_j, and in every auxiliary triangle the vertex linked to X and the one linked to the false literal, we obtain a VC V^* whose size is equal to $k = 2m + 3n$, which is optimal. Note that $X \notin V^*$; in fact, once we have put the n vertices corresponding to the true literals in the VC V^* in construction, we have *no choice* for the optimal (up to k) completion of V^*: when we take two vertices in T_j, we *must* take the two vertices which cover the membership edges linked to the two false literals (in Figure 8, the vertices b_1, d_1 and b_2, d_2); similarly, we have no choice either for the auxiliary triangles. So, if another optimal VC V^+ (of size k) exists, it must have a different distribution of its vertices over the components G_i, still with exactly one vertex in each G_i; this in turn defines a valid truth assignment, by setting $x_i = \mathrm{T}$ if $x_i \in V^+$, $x_i = \mathrm{F}$ if $\overline{x}_i \in V^+$. Now this assignment 1-3-satisfies $\mathcal{C}$, thanks in particular to our observation on the covering of the edges in E'_j. So we have two truth assignments 1-3-satisfying $\mathcal{C}$, contradicting the YES answer to U-1-3-SAT; therefore, V^* is the only optimal VC (with size k). So we have a YES answer to both U-VC and U-OVC.

(b) 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, this would lead, using the same argument as in the previous paragraph, to at least two optimal VC (of size $k = 2m + 3n$), and a NO answer to both U-VC and U-OVC. So we are left with the case when the set of clauses $\mathcal{C}$ cannot be 1-3-satisfied. As seen previously when discussing V^+, this implies that no VC of size k exists; this is sufficient to show that the answer is also NO for U-VC.

Assume then that V^* is an optimal VC, of unknown size $|V^*| > 2m+3n$. Then (i) at least one triangle contains three vertices of V^*, or (ii) at least one component G_i contains two vertices of V^*, or (iii) $X \in V^*$. Assume first that one triangle T_j or one auxiliary triangle T_i^* contains three vertices belonging to V^*; this can happen only if the three membership edges, auxiliary or not, starting from this triangle are not covered by their other ends (otherwise, we could save one vertex in the triangle). But then, exchanging in V^* one vertex of this triangle with the other end of its membership edge gives another optimal VC, and a NO answer to U-OVC. So we may assume from now on

that all the triangles have exactly two vertices in V^*. Assume next that a component G_i has two vertices belonging to V^*; then inside its auxiliary triangle T_i^*, we have at least two possibilities for choosing the two vertices belonging to V^*, and, once again, a NO answer to U-OVC. So we are left with the case when $X \in V^*$, but again, using the same type of argument, there is choice inside the auxiliary triangles. So in all cases, we have a NO answer to U-OVC. △

Corollary 31 *The decision problems UVC and U-OVC are NP-hard.*

Proof. Use Theorem 30 and Corollary 26(a). △

6.2.2 An Upper Bound for the Complexity of U-VC

Remark 32 *The method carried out in the proof of the following theorem is quite general and can be used with other types of problems, e.g., those involving the existence of a vertex set with bounded size in a graph: roughly speaking, the clauses constructed below in (a) "describe" the problem, those in (b) deal with the size of the set, and finally the clauses in (c) rule out multiple solutions obtained by permutations, symmetries, ..., and guarantee the uniqueness. See also the proof of Theorem 46 below,* as well as of Theorem 68.

Theorem 33 *There exists a polynomial reduction from U-VC to U-SAT: U-VC* $\rightarrow_p$ *U-SAT.*

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

(a) for each edge $x^h x^\ell \in E$, clauses of size $2k$: $\{x_1^h, x_2^h, \ldots, x_k^h, x_1^\ell, \ldots, x_k^\ell\}$;
(b1) for $1 \leq i \leq k$ and $1 \leq h < \ell \leq |V|$, clauses of size two: $\{\overline{x}_i^h, \overline{x}_i^\ell\}$;
(b2) for $1 \leq i < j \leq k$ and $1 \leq h \leq |V|$, clauses of size two: $\{\overline{x}_i^h, \overline{x}_j^h\}$;
(c) for $1 \leq i < k$ and $1 < \ell \leq |V|$, for $1 \leq h < \ell$ and $i < j \leq k$, clauses of size two: $\{\overline{x}_i^\ell, \overline{x}_j^h\}$.

Note that the number of variables and clauses is polynomial with respect to the order of G, since we may assume that $k \leq |V|$.

Assume that we have a unique VC of size k in G, $V^* = \{x^{p_1}, x^{p_2}, \ldots, x^{p_k}\}$, with $p_1 < p_2 < \ldots < p_k$. Observe that V^* is optimal (otherwise, any optimal VC destroys the uniqueness assumption). Define the

assignment $\mathcal{A}_1$ by $\mathcal{A}_1(x_q^{p_q}) = \mathrm{T}$ for $1 \leq q \leq k$, and all the other variables are set FALSE by $\mathcal{A}_1$. This assignment satisfies all the clauses; indeed:

(a) at least one of x^h, x^ℓ belongs to V^*; if, say, $x^h = x^{p_q} \in V^*$, then by definition x_q^h is set TRUE by $\mathcal{A}_1$, and satisfies the clause;

(b1) if $\{\overline{x}_i^h, \overline{x}_i^\ell\}$ is not satisfied for some h, i, ℓ, then $\mathcal{A}_1(x_i^h) = \mathcal{A}_1(x_i^\ell) = \mathrm{T}$, which would mean that two different vertices are the i-th element in V^*;

(b2) if $\{\overline{x}_i^h, \overline{x}_j^h\}$ is not satisfied, this means that x^h appears more than once in V^*;

(c) if $\{\overline{x}_i^\ell, \overline{x}_j^h\}$ is not satisfied for some i, ℓ, with $h < \ell$ and $i < j$, then $\mathcal{A}_1(x_i^\ell) = \mathcal{A}_1(x_j^h) = \mathrm{T}$. This means that $x^\ell = x^{p_i}$ and $x^h = x^{p_j}$; so $\ell = p_i$, $h = p_j$. Now $h < \ell$ implies that $p_j < p_i$, but $i < j$ implies that $p_i < p_j$, a contradiction.

Is $\mathcal{A}_1$ unique? Assume on the contrary that another assignment, $\mathcal{A}_2$, also satisfies the constructed instance of U-SAT. By (a), at least one variable x_i^h or x_j^ℓ is set TRUE by $\mathcal{A}_2$, for every h, ℓ corresponding to an edge $x^h x^\ell$, and for some i or j; so if V^+ is a vertex set which contains the vertex x^h as soon as some variable x_i^h is set TRUE by $\mathcal{A}_2$, then V^+ is a vertex cover. By (b1), for each $i \in \{1, \ldots, k\}$ there is at most one variable with subscript i set TRUE by $\mathcal{A}_2$; this tells us that we have constructed a VC with (at most) k elements. Since such a VC is unique by assumption, we can see that $\mathcal{A}_1$ and $\mathcal{A}_2$ have "selected" the same k vertices, i.e., for each $p_q \in \{p_1, \ldots, p_k\}$, there is exactly one variable, $x_q^{p_q}$, set TRUE by $\mathcal{A}_1$, and, thanks to (b2), exactly one variable, say $x_s^{p_q}$, set TRUE by $\mathcal{A}_2$. All the other variables with superscript p_q are FALSE by $\mathcal{A}_1$ or $\mathcal{A}_2$. Then, using (c), we can see that necessarily $q = s$ for every p_q, and that $\mathcal{A}_1$ and $\mathcal{A}_2$ must coincide: indeed, assume on the contrary that for some $q \in \{1, \ldots, k\}$, we have $q \neq s$; we treat the case $1 \leq q < s \leq k$, the case $1 \leq s < q \leq k$ being similar. Then if we consider the subscripts smaller than s, there must be one, say v, such that there is a superscript $p_u > p_q$ verifying $\mathcal{A}_2(x_v^{p_u}) = \mathrm{T}$. Now the clause from (c) $\{\overline{x}_v^{p_u}, \overline{x}_s^{p_q}\}$ is not satisfied by $\mathcal{A}_2$, a contradiction.

So a YES answer for U-VC leads to a YES answer for U-SAT. Assume now that the answer to U-VC is negative. If it is negative because there are at least two VC of size k, 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 VC, and different VC obviously lead to different assignments. If there is no VC of size k, then there is no assignment satisfying U-SAT, because such an assignment would give a VC of size k, as we have seen above with $\mathcal{A}_2$. So in both cases, a NO answer to U-VC implies a NO answer to U-SAT. $\triangle$

Theorem 34 *For every integer $k \geq 3$, the problems U-SAT, U-1-3-SAT, U-k-SAT and U-VC have equivalent complexity, up to polynomials.*

As a consequence, U-VC belongs to the class DP.

Proof. Simply gather Proposition 25, Theorems 30 and 33, and Corollary 26(b). △

Note that it could have been shown directly that U-VC belongs to *DP*.

6.2.3 Two Upper Bounds for the Complexity of U-OVC

We give a first upper bound on the complexity of U-OVC, because the proof is interesting in itself, because it uses a constructive argument (if there is a unique optimal vertex cover, the algorithm can output it), and because for some problems it is sometimes the only available method and result. In the case of Vertex Cover however, we can improve on Proposition 35, and instead of calling a polynomial number of times an algorithm solving a problem in *NP*, we need to call it only a logarithmic number of times, see Proposition 36.

Proposition 35 *The decision problem U-OVC belongs to the class P^{NP}. In case of a YES answer, one can give the only optimal vertex cover within the same complexity.*

Proof. Let $\mathcal{A}_1$ be an algorithm solving the problem VC, which is in *NP*, cf. Proposition 29; using a standard dichotomous process, we obtain an algorithm $\mathcal{A}_2$ outputting $\phi(G)$ for any graph G, with a logarithmic number of calls to $\mathcal{A}_1$.

Let $G = (V, E)$ be any instance of U-OVC, with n vertices. Run $\mathcal{A}_2$ for G, then, for each vertex $v \in V$, run $\mathcal{A}_2$ for G_v, the graph defined in Lemma 28. By the same lemma, we know, by comparing $\phi(G)$ and $\phi(G_v)$, whether v belongs to at least one optimal VC in G or not. Let $Y = \{v_1, \ldots, v_\ell\}$ be the set of vertices with a positive answer; then necessarily $\ell \in \{\phi(G), \phi(G) + 1, \ldots, n\}$. Now if $\ell > \phi(G)$, there is more than one optimal VC in G, whereas if $\ell = \phi(G)$, there is only one, namely, Y.

This shows that we can obtain the answer to U-OVC with $n + 1$ calls to $\mathcal{A}_2$, which leads to a polynomial number of calls to an algorithm solving the problem VC, together with negligible operations such as constructing G_v. Moreover, we can construct the optimal VC when there is one. △

Proposition 36 *The decision problem U-OVC belongs to the class L^{NP}.*

Proof. We use the same algorithm $\mathcal{A}_2$ as in the proof of the previous proposition, for any graph G. Then, once we have $\phi(G)$, we run an algorithm solving U-VC, for the instance consisting of G and $\phi(G)$. The answer is YES if and only if there is a unique optimal VC in G. Since the problem VC is in *NP* and U-VC is in *DP* (but actually, membership to L^{NP} would suffice), this amounts to a logarithmic number of calls to an algorithm solving a problem in *NP*, plus one call for a problem in *DP*. Because $DP \subseteq L^{NP}$, we are done. △

Note that using an algorithm for U-VC without knowing $\phi(G)$ would lead nowhere, because a NO answer cannot be interpreted: either $k < \phi(G)$ and there is no VC, or $k \geq \phi(G)$, but there is more than one VC.

6.3 Related Results: Cliques and Independent Sets

Let $G = (V, E)$ be a graph, and $G^c = (V, E^c)$ be its *complement*: $E^c = \{uv : u \in V, v \in V, u \neq v, uv \notin E\}$. Then the following three statements are equivalent: (a) V^* is a vertex cover in G; (b) $V \setminus V^*$ is an independent set in G; (c) $V \setminus V^*$ induces a clique in G^c. These relationships are simple enough to make it trivial to polynomially transform the problem VC to any one of the following two problems, and *vice-versa*:

Instance: A graph G and an integer k.

Question: Does G admit

(1) an independent set of size at least k? **(2)** a clique of size at least k?

It follows that

– these two problems have the same complexity as the problem VC;

– the problems of a unique clique (with bounded size or optimal) and of a unique independent set (with bounded size or optimal) have the same complexity as U-VC and U-OVC, respectively.

7 Dominating Codes

The approach for dominating codes is quite similar to the one for vertex covers, but slightly more complicated because we shall have to study successively 1-domination, then r-domination for general r.

After some necessary preliminary results, we are going to prove, for $r \geq 1$, the polynomial reductions

U-3-SAT $\rightarrow_p$ U-DC$_1$ and U-3-SAT $\rightarrow_p$ U-ODC$_1$ (Theorem 40),

U-DC$_1$ $\rightarrow_p$ U-DC$_r$ and U-ODC$_1$ $\rightarrow_p$ U-ODC$_r$ (Theorem 42),

U-DC$_r$ $\rightarrow_p$ U-DC$_1$ and U-ODC$_r$ $\rightarrow_p$ U-ODC$_1$ (Proposition 44),

U-DC$_1$ $\rightarrow_p$ U-SAT (Theorem 46).

The consequence of these reductions is that U-DC$_r$ and U-ODC$_r$ are *NP*-hard. Also, U-SAT and U-DC$_r$ have equivalent complexity; as a result, U-DC$_r$ belongs to *DP*. We shall also show that U-ODC$_r$ belongs to the class L^{NP}.

7.1 Preliminary Results

The following lemma will be used in the proof of Theorem 42.

Lemma 37 *Let $r \geq 1$ be any integer and let $G^*_{uv} = (V^*_{uv}, E^*_{uv})$ be the graph defined as follows:*

$$V^*_{uv} = \{u, v\} \cup \{\alpha_i : 1 \leq i \leq r-1\} \cup \{\beta_{i,j} : 1 \leq i \leq r-1, 1 \leq j \leq 3r\},$$

$$E^*_{uv} = \{u\alpha_1, \alpha_1\alpha_2, \ldots, \alpha_{r-1}v\} \cup$$
$$\cup \{\alpha_i\beta_{i,1}, \beta_{i,1}\beta_{i,2}, \ldots, \beta_{i,r}\beta_{i,r+1}, \ldots, \beta_{i,2r-1}\beta_{i,2r} : 1 \leq i \leq r-1\} \cup$$
$$\cup \{\beta_{i,r}\beta_{i,2r+1}, \beta_{i,2r+1}\beta_{i,2r+2}, \ldots, \beta_{i,3r-1}\beta_{i,3r} : 1 \leq i \leq r-1\},$$

see Figure 10 further down. Then $\gamma_r(G^*_{uv}) = r$, $C_1 = \{u\} \cup \{\beta_{i,r} : 1 \leq i \leq r-1\}$ and $C_2 = \{v\} \cup \{\beta_{i,r} : 1 \leq i \leq r-1\}$ are two optimal r-dominating codes in G^*_{uv}, and any optimal r-dominating codes in G^*_{uv} contains $W = \{\beta_{i,r} : 1 \leq i \leq r-1\}$.

Proof. Because the $r-1$ vertices $\beta_{i,2r}$ must be r-dominated by some codeword, at least $r-1$ codewords are necessary. But no vertex can simultaneously r-dominate $\beta_{i,2r}$ and u or v, so at least one more codeword is required, and $\gamma_r(G^*_{uv}) \geq r$. On the other hand, it is quite straightforward to check that C_1 and C_2 are r-dominating codes, of size r, so that they are optimal, and $\gamma_r(G^*_{uv}) = r$. Finally, for a given $i \in \{1, \ldots, r-1\}$, only $\beta_{i,r}$ can r-dominate both $\beta_{i,2r}$ and $\beta_{i,3r}$, and so $\beta_{i,r}$ belongs to any optimal r-dominating code. △

Also note that, for any given i with $1 \leq i \leq r-1$, the vertex $\beta_{i,r}$ r-dominates *exactly* α_i and $\beta_{i,j}$, for $1 \leq j \leq 3r$.

The following lemma, which characterizes the vertices belonging to at least one optimal r-dominating code, through the comparison of two r-domination numbers, is very similar to Lemma 28 for vertex covers; it will be used for Proposition 48. It is a simplified version of [33, Cor. 2].

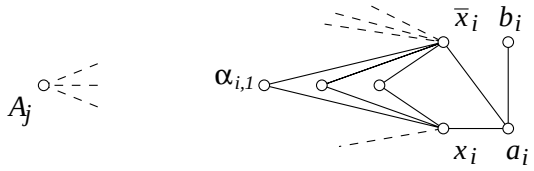

Figure 9: The graph G constructed in the proof of Theorem 40.

Lemma 38 *Let $G = (V, E)$ be a graph, and let $r \geq 1$ be any integer. For a given vertex $\alpha \in V$, we consider the following graph: $G_\alpha = (V_\alpha, E_\alpha)$, with*

$$V_\alpha = V \cup \{\beta_i : 1 \leq i \leq r\},\ E_\alpha = E \cup \{\alpha\beta_1\} \cup \{\beta_i\beta_{i+1} : 1 \leq i \leq r-1\},$$

where for $i \in \{1, \ldots, r\}$, $\beta_i \notin V$. Then α belongs to at least one optimal r-dominating code in G if and only if $\gamma_r(G) = \gamma_r(G_\alpha)$. △

Proposition 39 *[21, p. 75 and p. 190, for $r = 1$], [33, Prop. 9] Let $r \geq 1$ be any integer. The decision problem DC$_r$ is NP-complete.* △

Actually, we shall only use the fact that DC$_r$ belongs to *NP*, for Propositions 48 and 49. Note that the proofs of Proposition 39 do not deal with the problem of the uniqueness of a solution.

7.2 Uniqueness of Dominating Code

7.2.1 From U-3-SAT to U-DC$_1$ and U-ODC$_1$

Theorem 40 *There exists a polynomial reduction from U-3-SAT to U-DC$_1$ and to U-ODC$_1$: U-3-SAT $\rightarrow_p$ U-DC$_1$ and U-3-SAT $\rightarrow_p$ U-ODC$_1$.*

Proof. The construction of a graph is common to the two reductions, then we add an integer k for U-DC$_1$. We start from an instance of U-3-SAT, a collection $\mathcal{C}$ of m clauses over a set $\mathcal{X}$ of n variables. For each variable $x_i \in \mathcal{X}$, $1 \leq i \leq n$, we construct the graph $G_i = (V_i, E_i)$ as follows (see Figure 9):

$$V_i = \{x_i, \overline{x}_i, a_i, b_i\} \cup \{\alpha_{i,\ell} : 1 \leq \ell \leq 3\},$$

$$E_i = \{x_i a_i, \overline{x}_i a_i, a_i b_i\} \cup \{x_i\alpha_{i,\ell}, \overline{x}_i\alpha_{i,\ell} : 1 \leq \ell \leq 3\}.$$

Then for each clause c_j, $1 \leq j \leq m$, containing three literals, we create one vertex, A_j, and link it to the three vertices corresponding, in the graphs G_i,

to the literals of c_j. This is our graph G; its order is $7n + m$. Additionally, we set $k = 2n$ for U-DC$_1$.

Note already that, because of the vertices b_i and $\alpha_{i,\ell}$, any optimal 1-dominating code in G contains x_i or $\overline{x}_i$, and a_i or b_i, for all $i \in \{1, \ldots, n\}$. Consequently, $\gamma_1(G) \geq 2n = k$.

We claim that there is a unique solution to 3-SAT if and only if there is a unique optimal 1-dominating code, and if and only if there is a unique 1-dominating code of size (at most) k, in G.

(1) Assume first that there is a unique truth assignment satisfying all the clauses. We construct the following code C: for $i \in \{1, \ldots, n\}$, among the vertices $x_i \in V_i$, $\overline{x}_i \in V_i$, we put in C the vertex x_i if the literal x_i has been set TRUE, the vertex $\overline{x}_i$ if the literal x_i is FALSE, and we add a_i. Then C is a 1-dominating code; in particular, every vertex A_j is 1-dominated by at least one codeword since every clause contains at least one true literal. The code C has size $k = 2n$ and is optimal. Moreover, once x_i or $\overline{x}_i$ is a codeword, the *only way* to complete the code with not more than n additional codewords is to take a_i, so that $\overline{x}_i$ or x_i is 1-dominated by C, since no A_j is a codeword. The code C is unique: suppose on the contrary that C^* is another 1-dominating code of size $2n$ in G. Then $|C^* \cap V_i| = 2$ for all $i \in \{1, \ldots, n\}$, no A_j is a codeword, and exactly one of x_i and $\overline{x}_i$ is in C^*. This defines a valid truth assignment for $\mathcal{X}$, by setting $x_i = \mathrm{T}$ if $x_i \in C^*$, $x_i = \mathrm{F}$ if $\overline{x}_i \in C^*$. Since $C \neq C^*$, this assignment is different from the assignment used to build C. But the fact that C^* 1-dominates A_j for all j shows that there is a codeword x_i or $\overline{x}_i$ 1-dominating A_j, which means that the clause c_j is satisfied. Therefore, we have a second assignment satisfying the instance of 3-SAT, a contradiction. We can conclude that both problems, U-DC$_1$ and U-ODC$_1$, also have a YES answer.

(2) Assume next that the answer to U-3-SAT is NO: this may be either because no truth assignment satisfies the instance, or because at least two assignments do; in the latter case, this would lead, using the same argument as previously, to at least two optimal 1-dominating codes of size $k = 2n$, and a NO answer to U-DC$_1$ and to U-ODC$_1$. So we are left with the case when the set of clauses $\mathcal{C}$ cannot be satisfied. This implies that no 1-dominating code of size $k = 2n$ exists, as seen with C^*; this is sufficient to end the proof for U-DC$_1$, but we have to go on for U-ODC$_1$: assume then that C is an optimal 1-dominating code of unknown size $|C| > 2n$. We know that each V_i contains at least two codewords. Where can the extra codeword(s) be?

Suppose that a vertex A_{j_0} is a codeword. If the three vertices in $\{x_i, \overline{x}_i : 1 \leq i \leq n\}$ to which A_{j_0} is linked are codewords, then A_{j_0} could have been saved. So there is at least one neighbour of A_{j_0}, say x_{i_0}, which does

not belong to C. Now $\overline{x}_{i_0} \in C$, x_{i_0} is 1-dominated by $A_{j_0} \in C$, and either $C \cap V_{i_0} = \{\overline{x}_{i_0}, a_{i_0}\}$ or $C \cap V_{i_0} = \{\overline{x}_{i_0}, b_{i_0}\}$ may be part of an optimal solution, i.e., we have a NO answer for U-ODC$_1$.

So we can assume from now on that no A_j is a codeword, and that there is at least one set V_{i_0} containing at least three codewords. If it is more than three, then codewords can be spared. If it is exactly three, then x_{i_0} and $\overline{x}_{i_0}$ are codewords, so that some vertices A_j linked to them are 1-dominated by C thanks to x_{i_0} and $\overline{x}_{i_0}$: otherwise, two codewords inside V_{i_0} would have been sufficient. Now we have a choice for the third codeword in V_{i_0}: it can be either a_{i_0} or b_{i_0}.

In all cases, we have proved that there can be several optimal 1-dominating codes, i.e., we have a NO answer for G, the constructed instance of U-ODC$_1$. △

Remark 41 *The fact that all the clauses have degree three has no importance whatsoever, and the proof could also work using any problem U-k-SAT, $k \geq 3$, or U-SAT. Only the degrees of the vertices A_j would be affected.*

7.2.2 Extension to $r \geq 2$

Theorem 42 *Let $r \geq 2$ be any integer. There is a polynomial reduction from U-ODC$_1$ to U-ODC$_r$ and from U-DC$_1$ to U-DC$_r$: U-ODC$_1$ $\rightarrow_p$ U-ODC$_r$ and U-DC$_1$ $\rightarrow_p$ U-DC$_r$.*

Proof. This proof is inspired by the proof of Proposition 9 in [33]. We start from a graph $G = (V, E)$ and an integer k for U-DC$_1$, and the same graph G for U-ODC$_1$.

The graph $G^* = (V^*, E^*)$ is common to the reduction to U-DC$_r$ and to U-ODC$_r$ (see Figure 10), and is defined by setting, for each edge $e = uv \in E$,

$$V_e^* = \{\alpha_{e,i} : 1 \leq i \leq r-1\} \cup \{\beta_{e,i,j} : 1 \leq i \leq r-1, 1 \leq j \leq 3r\},$$

$$E_e^* = \{u\alpha_{e,1}, \alpha_{e,1}\alpha_{e,2}, \ldots, \alpha_{e,r-2}\alpha_{e,r-1}, \alpha_{e,r-1}v\} \cup$$
$$\cup \{\alpha_{e,i}\beta_{e,i,1}, \beta_{e,i,1}\beta_{e,i,2}, \ldots, \beta_{e,i,r}\beta_{e,i,r+1}, \ldots, \beta_{e,i,2r-1}\beta_{e,i,2r} : 1 \leq i \leq r-1\} \cup$$
$$\cup \{\beta_{e,i,r}\beta_{e,i,2r+1}, \beta_{e,i,2r+1}\beta_{e,i,2r+2}, \ldots, \beta_{e,i,3r-1}\beta_{e,i,3r} : 1 \leq i \leq r-1\}.$$

Then we set

$$V^* = V \cup (\cup_{e \in E} V_e^*),\ E^* = \cup_{e \in E} E_e^*.$$

Finally, for U-DC$_r$, we put $k^* = k + (r-1)|E|$. The order of G^* is $|E|(r-1)(3r+1)$.

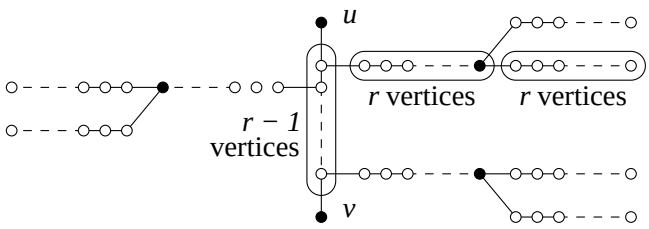

Figure 10: How the edge $e = uv \in E$ gives V_e^* and E_e^*. The black vertices on the branches are the vertices $\beta_{e,i,r}$.

(1) We claim that an instance of U-ODC$_1$ is positive if and only if the corresponding instance of U-ODC$_r$ is.

(a) First, we assume that there is a YES answer in G for U-ODC$_1$: there is a unique optimal 1-dominating code C in G. Let W be the set consisting of the $(r-1)|E|$ vertices $\beta_{e,i,r}$, $e \in E$, $1 \leq i \leq r-1$. Note that W r-dominates exactly $V^* \setminus V$, and that, by Lemma 37, any optimal r-dominating code in G^* contains W. Because C is 1-dominating in G, clearly $C^* = C \cup W$ is r-dominating in G^*. Note in particular that two vertices u and v at distance 1 in G are at distance r in G^*.

Because moreover C is optimal, the code C^* is also optimal: assume on the contrary that C^+ is an optimal r-dominating code in G^*, with $|C^+| < |C^*|$. We proceed as in the proof of Proposition 9 in [33]: the subcode $C^+ \setminus W$ must r-dominate all the vertices in V, and if a codeword in $V^* \setminus V$ performs part of this task, it can be replaced by a vertex in V. After such replacements have possibly been made, we have a code $C^\times$ such that $C^\times \cap V$ r-dominates, in G^*, all the vertices in V; this implies that in G, $C^\times \cap V$ 1-dominates all the vertices. But $|C^\times \cap V| \leq |C^+ \setminus W| < |C^* \setminus W| = |C| = \gamma_1(G)$, i.e., $|C^\times \cap V| < \gamma_1(G)$, which is impossible.

Finally, because we assumed that C was the only optimal 1-dominating code in G, the code $C^* = C \cup W$ is the only optimal r-dominating code in G^*. Suppose on the contrary that C^+ is another optimal code in G^*: $|C^+| = |C^*| = |C| + |W|$.

(i) $C^+ \cap V = C^+ \setminus W$, or, equivalently, $(V^* \setminus V) \cap C^+ = W$. Then, as we have already seen, $C^+ \cap V$ is 1-dominating in G, and either $C^+ \cap V = C^* \cap V$, which leads to $C^+ = C^*$, or $C^+ \cap V \neq C^* \cap V$, in which case we have two optimal 1-dominating codes in G, $C^+ \setminus W$ and $C^* \setminus W = C$. In both cases, we have a contradiction.

(ii) $C^+ \cap V \neq C^+ \setminus W$. Then there is at least one codeword $z \in C^+$

belonging to some $V_e^* \setminus \{\beta_{e,i,r} : 1 \leq i \leq r-1\}$, with $e = uv \in E$. Then z must r-dominate u or v or both; otherwise, z is useless and can be saved. For the same reason, $u \notin C^+$ and $v \notin C^+$; but then $(C^+ \setminus \{z\}) \cup \{u\}$ and $(C^+ \setminus \{z\}) \cup \{v\}$ are also both r-dominating in G^*. By similarly replacing all the t codewords in $(V^* \setminus W) \setminus V$, $t \geq 1$, by codewords in V, we have 2^t codes included in $V \cup W$ which are all r-dominating in G^*, which implies that their intersections with V all are 1-dominating in G; moreover, these intersections have size (at most) $|C^+| - |W| = |C|$, i.e., are optimal. Finally, there are at least two of them, so at least one is different from C, contradicting its uniqueness.

This shows that a YES answer to U-ODC$_1$ leads to a YES answer to U-ODC$_r$.

(b) And a NO answer to U-ODC$_1$ leads to a NO answer to U-ODC$_r$, since two optimal 1-dominating codes in G, C_1 and C_2, give two optimal r-dominating codes in G^*, $C_1 \cup W$ and $C_2 \cup W$.

This ends the part for U-ODC$_r$.

(2) We claim that an instance of U-DC$_1$ is positive if and only if the corresponding instance of U-DC$_r$ is.

(a) First, we assume that there is a YES answer for U-DC$_1$: there is a unique 1-dominating code C with size at most k in G. Obviously, C is optimal, otherwise any optimal 1-dominating code in G would contradict the uniqueness of C. Consider the code $C^* = C \cup W$ in G^*, of size $k+(r-1)|E| = k^*$. Exactly as in Step (1) of this proof, C^* is r-dominating, is optimal, and is the only r-dominating code of size at most k^* in G^*. So the answer to U-DC$_r$ is also positive.

(b) Next, we assume that the answer to U-DC$_1$ is NO: either there is no 1-dominating code with size at most k in G, or there is more than one. In the latter case, we have more than one r-dominating code with size at most k^* in G^*: simply add the set W to the codes in G. So we assume that we are in the first case. This implies in particular that $\gamma_1(G) > k$; using the same argument as in Step (1), we can see that $\gamma_r(G^*) > k + (r-1)|E| = k^*$, and there is no r-dominating code with size at most k^* in G^*. In all cases, the answer to U-DC$_r$ is NO. △

Corollary 43 *Let $r \geq 1$ be any integer. The decision problems U-DC$_r$ and U-ODC$_r$ are NP-hard.* △

Proposition 44 *Let $r \geq 2$ be any integer. There is a polynomial reduction from U-DC$_r$ to U-DC$_1$ and from U-ODC$_r$ to U-ODC$_1$: U-DC$_r$ $\rightarrow_p$ U-DC$_1$ and U-ODC$_r$ $\rightarrow_p$ U-ODC$_1$.*

Proof. Let (G, k) be an instance of U-DC$_r$ and G be an instance of U-ODC$_r$, for $r \geq 2$. The instance for U-DC$_1$ is simply (G^r, k), and G^r for U-ODC$_1$, where G^r is the r-th power of G: obviously, by Lemma 23, there is a unique 1-dominating code of size k in G^r if and only if there is a unique r-dominating code of size k in G, and there is a unique optimal 1-dominating code in G^r if and only if there is a unique optimal r-dominating code in G. △

Corollary 45 *Let $r_1 \geq 1$ and $r_2 \geq 1$ be any integers.*

(a) The decision problems U-DC$_{r_1}$ and U-DC$_{r_2}$ are equivalent, up to polynomials.

(b) The decision problems U-ODC$_{r_1}$ and U-ODC$_{r_2}$ are equivalent, up to polynomials. △

7.2.3 An Upper Bound for the Complexity of U-DC$_r$

Theorem 46 *There exists a polynomial reduction from U-DC$_1$ to U-SAT: U-DC$_1$ $\rightarrow_p$ U-SAT.*

Proof. We fully use Remark 32: we start from an instance of U-DC$_1$, a graph $G = (V, E)$ and an integer k, with $V = \{x^1, \ldots, x^{|V|}\}$; we assume that $|V| \geq 3$. We create the set of variables $\mathcal{X} = \{x_i^h : 1 \leq h \leq |V|, 1 \leq i \leq k\}$ and the following clauses:

(a) for each vertex $x^h \in V$ with neighbours $x^{h_1}, \ldots, x^{h_s}$, clauses of size $(s+1)k$: $\{x_1^h, x_2^h, \ldots, x_k^h, x_1^{h_1}, \ldots, x_k^{h_1}, \ldots, x_1^{h_s}, \ldots, x_k^{h_s}\}$;

(b1) for $1 \leq i \leq k$ and $1 \leq h < \ell \leq |V|$, clauses of size two: $\{\overline{x}_i^h, \overline{x}_i^\ell\}$;

(b2) for $1 \leq i < j \leq k$ and $1 \leq h \leq |V|$, clauses of size two: $\{\overline{x}_i^h, \overline{x}_j^h\}$;

(c) for $1 \leq i < k$ and $1 < \ell \leq |V|$, for $1 \leq h < \ell$ and $i < j \leq k$, clauses of size two: $\{\overline{x}_i^\ell, \overline{x}_j^h\}$.

Compared to the proof of Theorem 33, only the clauses in (a) are different: they convey the fact that among the vertices $x^h, x^{h_1}, \ldots, x^{h_s}$, at least one must be put in a 1-dominating code. The proof then goes exactly as for Theorem 33. △

By Proposition 44 or its corollary, this immediately implies that there is a polynomial reduction from U-DC$_r$ to U-SAT.

Theorem 47 *Let $r \geq 1$ and $k \geq 3$ be any integers. The problem U-DC$_r$ has complexity equivalent to that of U-SAT, U-k-SAT and U-1-3-SAT.*

As a consequence, U-DC$_r$ belongs to the class DP. △

Note that it could have been shown directly that U-DC$_r$ belongs to *DP*.

7.2.4 Two Upper Bounds for the Complexity of U-ODC$_r$

The (constructive) membership of U-ODC$_r$ to the class P^{NP} and the membership to L^{NP} can be proved in exactly the same way as in Section 6.2.3 for U-OVC. This time, it is the characterizing Lemma 38 which must be used, together with Proposition 39.

Proposition 48 *For $r \geq 1$, the decision problem U-ODC$_r$ belongs to the class P^{NP}. In case of a YES answer, one can give the only optimal r-dominating code within the same complexity.* △

Proposition 49 *For $r \geq 1$, the decision problem U-ODC$_r$ belongs to L^{NP}.* △

8 Conclusion

Corollary 26 states that the three problems, U-SAT, U-1-3-SAT and U-k-SAT ($k \geq 3$), are equivalent and lie somewhere in the vertically hatched area of Figure 11, but probably not in *DP-C*, cf. Remark 27.

Theorems 34 and 47 state that U-VC and U-DC$_r$, $r \geq 1$, have the same complexity as the above three problems, and consequently are located in the same hatched area.

We have also established that the decision problems U-OVC and U-ODC$_r$, $r \geq 1$, belong to the class L^{NP}, see Propositions 36 and 49, and that they are *NP*-hard, see Corollaries 31 and 43. This means that they lie within the areas that are hatched horizontally or vertically. Moreover, all the problems U-ODC$_r$ are equivalent between each other, by Corollary 45(b).

We have the same conclusions for Cliques and Independent Sets (see Section 6.3).

In [8], the authors wonder whether

(A) U-SAT is *NP*-hard, but here we believe that what they mean is: does there exist a *polynomial* reduction from an *NP*-complete problem to U-SAT ? i.e., they use the *second* definition of *NP*-hardness;

finally, they show that (A) is true if and only if

(B) U-SAT is *DP*-complete.

So, if one is careless and considers that U-SAT is *NP*-hard without checking according to which definition, one might easily jump too hastily to the conclusion that U-SAT is *DP*-complete, which, to our knowledge, is not known to be true or not. As for U-3-SAT, we do not know where to locate it more

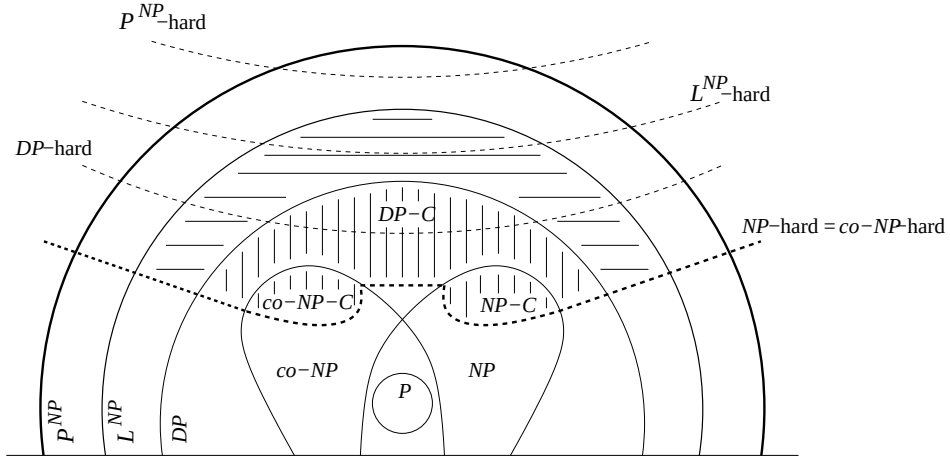

Figure 11: Some classes of complexity: Figure 7 re-visited.

precisely either; in [10] the problems U-k-SAT and more particularly U-3-SAT are studied, but it appears that they are versions where the given set of clauses has zero or one solution, which makes quite a difference with our problem.

Open problem 1 (general). For $k \geq 3$ and $r \geq 1$, improve the location of U-SAT, U-k-SAT, U-1-3-SAT, U-VC, U-OVC, U-DC$_r$ and U-ODC$_r$, within the classes of complexity.

Open problem 2. It is easy to establish that U-OVC $\rightarrow_p$ U-ODC$_1$ (and U-OVC $\rightarrow_p$ U-ODC$_r$, $r \geq 2$). What more can be said about the relationship between U-OVC and U-ODC$_1$ (and U-ODC$_r$)?

Open problem 3 (conjecture). Under the assumption $P \neq NP$, U-OVC is not equivalent to U-VC, and U-ODC$_r$ is not equivalent to U-DC$_r$, $r \geq 1$.

Finally, in [17] (respectively, [24]), characterizations of the trees (respectively, the block graphs, which are a class of graphs including the trees) admitting a unique optimal 1-dominating code are given. This result, together with a linear algorithm determining optimal 1-dominating codes in block graphs, then allows to show that the sub-problem of U-ODC$_1$ where the instance is any block graph is linear.

Open problem 4. Is it possible to extend this kind of result to any integer $r \geq 1$? to other classes of graphs?

Unique (Optimal) Solutions: Complexity Results for Identifying and Locating-Dominating Codes

Olivier Hudry
LTCI, Télécom ParisTech, Université Paris-Saclay
46 rue Barrault, 75634 Paris Cedex 13 - France

& Antoine Lobstein
Centre National de la Recherche Scientifique
Laboratoire de Recherche en Informatique, UMR 8623,
Université Paris-Sud, Université Paris-Saclay
Bâtiment 650 Ada Lovelace, 91405 Orsay Cedex - France

olivier.hudry@telecom-paristech.fr
antoine.lobstein@lri.fr

Abstract

We investigate the complexity of four decision problems dealing with the *uniqueness* of a solution in a graph: "Uniqueness of an r-Locating-Dominating Code with bounded size" (U-LDC$_r$), "Uniqueness of an Optimal r-Locating-Dominating Code" (U-OLDC$_r$), "Uniqueness of an r-Identifying Code with bounded size (U-IdC$_r$), "Uniqueness of an Optimal r-Identifying Code" (U-OIdC$_r$), for any fixed integer $r \geq 1$.

In particular, we describe a polynomial reduction from "Unique Satisfiability of a Boolean formula" (U-SAT) to U-OLDC$_r$, and from U-SAT to U-OIdC$_r$; for U-LDC$_r$ and U-IdC$_r$, we can do even better and prove that their complexity is the same as that of U-SAT, up to polynomials. Consequently, all these problems are *NP*-hard, and U-LDC$_r$ and U-IdC$_r$ belong to the class *DP*.

Key Words: Graph Theory, Complexity, Complexity Classes, Polynomial Hierarchy, *NP*-Completeness, Hardness, P^{NP}, Uniqueness of (Optimal) Solution, Locating-Dominating Codes, Identifying Codes, Twin-Free Graphs, Boolean Satisfiability Problems

9 Introduction

We intend to locate in the classes of complexity some problems dealing with the existence of a unique identifying or locating-dominating code in a given graph.

9.1 Identifying and Locating-Dominating Codes

For graph theory, we refer to, e.g., [7] or [16].

For identification in graphs, see the seminal paper [40]; for locating-dominating codes, see the first papers [13] and [49]. For both, see also the large bibliography at [41].

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, $|V|$.

A *path* $P_k = x_1 x_2 \dots x_k$ is a sequence of k distinct vertices x_i, $1 \leq i \leq k$, such that $x_i x_{i+1}$ is an edge for $i \in \{1, 2, \dots, k-1\}$. The *length* of P_k is its number of edges, $k-1$. A *cycle* $\mathcal{C}_k = x_1 x_2 \dots x_k$ is a sequence of k distinct vertices x_i, $1 \leq i \leq k$, where $x_i x_{i+1}$ is an edge for $i \in \{1, 2, \dots, k-1\}$, and $x_k x_1$ is also an edge; its length is k.

In a connected graph G, we can define the *distance* between any two vertices x and y, denoted by $d_G(x, y)$, as the length of any shortest path between x and y. This definition can be extended to disconnected graphs, using the convention that $d_G(x, y) = +\infty$ if no path exists between x and y. The subscript G can be dropped when there is no ambiguity.

For an integer $k \geq 2$, the *k-th transitive closure*, or *k-th power* of $G = (V, E)$ is the graph $G^k = (V, E^k)$ defined by $E^k = \{uv : u \in V, v \in V, d_G(u, v) \leq k\}$.

For any vertex $v \in V$, the *open neighbourhood $N(v)$ of v* consists of the set of vertices adjacent to v, i.e., $N(v) = \{u \in V : uv \in E\}$; the *closed neighbourhood of v* is $B_1(v) = N(v) \cup \{v\}$. This notation can be generalized to any integer $r \geq 0$ by setting

$$B_r(v) = \{x \in V : d(x, v) \leq r\}.$$

For $X \subseteq V$,we denote by $B_r(X)$ the set of vertices within distance r from X:

$$B_r(X) = \cup_{x \in X} B_r(x).$$

Two vertices x and y such that $B_r(x) = B_r(y)$, $x \neq y$, are called *r-twins*. If G has no r-twins, we say that G is *r-twin-free*. Whenever two vertices

x and y are such that $x \in B_r(y)$ (which is equivalent to $y \in B_r(x)$), we say that x and y *r-cover* or *r-dominate* each other; note that every vertex r-dominates itself. A set W is said to r-dominate a set Z if every vertex in Z is r-dominated by at least one vertex of W, or equivalently: $Z \subseteq B_r(W)$. When three vertices x, y, z are such that $x \in B_r(z)$ and $y \notin B_r(z)$, we say that z *r-separates* x and y in G (note that $z = x$ is possible). A set of vertices is said to r-separate x and y if it contains at least one vertex which does.

A *code* C is simply a subset of V, and its elements are called *codewords*.

We say that C is an *r-identifying code* [40] if all the sets $B_r(v) \cap C$, $v \in V$, are nonempty and distinct: in other words, every vertex is r-covered by C, *and* every pair of vertices is r-separated by C. It is quite easy to observe that a graph G admits an r-identifying code if and only if G is r-twin-free; this is why r-twin-free graphs are also called *r-identifiable*. When G is r-twin-free, we denote by $i_r(G)$ the smallest cardinality of an r-identifying code in G, and call it the *r-identification number* of G; any r-identifying code C such that $|C| = i_r(G)$ is said to be *optimal*.

We say that C is an *r-locating-dominating code* (r-LD code for short) [49], [13] if all the sets $B_r(v) \cap C$, $v \in V \setminus C$, are nonempty and distinct: in other words, every vertex is r-dominated by C (since a codeword dominates itself), and every pair of non-codewords is r-separated by C. We denote by $LD_r(G)$ the smallest cardinality of an r-locating-dominating code in G, and call it the *r-location-domination number* of G; any r-LD code C such that $|C| = LD_r(G)$ is said to be *optimal*.

For the needs of Theorems 68 and 83, we give the following obvious characterization: a code C is r-identifying (respectively, r-LD) if and only if (a) for every vertex $x \in V$, $B_r(x) \cap C \neq \emptyset$, and (b) for every pair of distinct vertices $x^i \in V$, $x^j \in V$ (respectively, $x^i \in V \setminus C$, $x^j \in V \setminus C$), we have

$$\big(B_r(x^i) \Delta B_r(x^j)\big) \cap C \neq \emptyset, \tag{2}$$

where Δ stands for the symmetric difference.

Note that, when dealing with locating-dominating codes, we shall rather use the word "dominate", whereas for identifying codes, we shall prefer "cover".

It is known that the following two decision problems, stated for any integer $r \geq 1$, are *NP*-complete (see below Propositions 62 from [13], [11], and 78 from [12], [11]):

Problem LDC$_r$ (r-Locating-Dominating Code with bounded size):
Instance: A graph G and an integer k.

Question: Does G admit an r-locating-dominating code of size at most k?

Problem IdC_r (r-Identifying Code with bounded size):
Instance: An r-twin-free graph G and an integer k.
Question: Does G admit an r-identifying code of size at most k?

In this paper, we are interested in the following four problems, which deal with the *uniqueness* of a solution, and we are going to locate them in the classes of complexity.

Problem U-LDC_r (Unique r-Locating-Dominating Code with bounded size):
Instance: A graph G and an integer k.
Question: Does G admit a *unique* r-locating-dominating code of size at most k?

Problem U-OLDC_r (Unique Optimal r-Locating-Dominating Code):
Instance: A graph G.
Question: Does G admit a *unique optimal* r-locating-dominating code?

Problem U-IdC_r (Unique r-Identifying Code with bounded size):
Instance: An r-twin-free graph G and an integer k.
Question: Does G admit a *unique* r-identifying code of size at most k?

Problem U-OIdC_r (Unique Optimal r-Identifying Code):
Instance: An r-twin-free graph G.
Question: Does G admit a *unique optimal* r-identifying code?

Our results are the following: we give polynomial reductions from "Unique Satisfiability of a Boolean formula" (U-SAT) to U-OLDC_r, as well as from U-SAT to U-OIdC_r; we prove that U-LDC_r and U-IdC_r have the same complexity as U-SAT, up to polynomials. As a consequence, all these problems are *NP*-hard, and U-LDC_r and U-IdC_r belong to the class *DP*. The problems U-OLDC_r and U-OIdC_r belong "only" to the class L^{NP}, which contains *DP*.

In a previous work [32], we have investigated the complexity of the existence of, and of the search for, optimal r-identifying codes, as well as optimal r-identifying codes containing a given subset of vertices; see also [11], [12, Sec. 5]. In a forthcoming work, we extend the present study on uniqueness issues to Boolean satisfiability and graph colouring [34] *(Sections 1–4 in this Report)*, Vertex Cover and Dominating Set (as well as its generalization to domination within distance r) [35] *(Sections 5–8)*, and Hamiltonian Cycle [37] *(Sections 14–16)*. At the other end, there has been research on *how many* optimal r-identifying codes can exist in a graph [28], and on the structure of the ensemble of optimal r-locating-dominating codes [29] and of optimal r-identifying codes [30].

For other works in the area of complexity, see, e.g., [2], [3], [4], [18] and [47], which establish, in particular, polynomiality or *NP*-completeness results for the identification problem when restricted to some subclasses of graphs, such as trees, planar graphs, bipartite graphs, interval graphs or line graphs. See also [19], [23] and [50] for approximation issues for both identifying and locating-dominating codes.

In the sequel, we shall also need the following tools, which constitute classical definitions related to Boolean satisfiability.

We consider a set $\mathcal{X}$ of n *Boolean variables* x_i and a set $\mathcal{C}$ of m *clauses*, each clause c_j containing κ_j *literals*, a literal being a variable x_i or its complement $\overline{x}_i$. 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 problem, for which the size of the instance is polynomially linked to $n+m$, is a classical problem in complexity.

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

We shall also need the variant U-SAT of SAT, which has the same instance as SAT but the question now is: "Is there a *unique* truth assignment...?".

We shall give in Proposition 51 what we need to know about the complexity of this problem. We now provide the necessary definitions and notation for complexity.

9.2 Necessary Notions in Complexity

We expound here, not too formally, the notions of complexity that will be needed in the sequel. We refer the reader to, e.g., [6], [21], [38] or [45] for more on this topic.

A *decision problem* is of the type "Given an instance I and a property $\mathcal{PR}$ on I, is $\mathcal{PR}$ true for I?", and has only two solutions, "yes" or "no". The class P will denote the set of problems which can be solved by a *polynomial* (time) algorithm, and the class *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 transformation 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 shall write $\pi_1 \rightarrow_p \pi_2$. Cook [14] proved that there is one problem in *NP*, namely SAT, to which every other problem in *NP* can be polynomially reduced. Thus, in a sense, SAT is the "hardest" problem inside *NP*. Other problems share this property in *NP* and are called *NP-complete* problems; their class is denoted by *NP-C*. The way to show that a decision problem π is *NP*-complete is, once it is proved to be in *NP*, to choose some *NP*-complete problem π_1 and to polynomially reduce it to π. From a practical viewpoint, the *NP*-completeness of a problem π implies that we do not know any polynomial algorithm solving π, and that, under the assumption $P \neq NP$, which is widely believed to be true, no such algorithm exists: the time required can grow exponentially with the size of the instance (when the instance is a graph, the size is polynomially linked to its order).

The *complement* of a decision problem, "Given I and $\mathcal{PR}$, is $\mathcal{PR}$ true for I?", is "Given I and $\mathcal{PR}$, is $\mathcal{PR}$ false for I?". The class *co-NP* (respectively, *co-NP-C*) is the class of the problems which are the complement of a problem in *NP* (respectively, in *NP-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. Thus, in this sense, π_2 is "at least as hard" as π_1. A problem π is *NP-hard* (respectively, *co-NP-hard*) if there is a Turing reduction from some *NP*-complete (respectively, co-*NP*-complete) problem to π [21, p. 113].

Remark 50 *Note that with these definitions, NP-hard and co-NP-hard coincide [21, p. 114].*

The notions of completeness and hardness can of course be extended to classes other than *NP* or co-*NP*. *NP*-hardness is defined differently in [15] and [27]: there, a problem π is *NP-hard* if there is a *polynomial* reduction from some *NP*-complete problem to π; this may lead to confusion (see Section 13).

Finally we introduce the classes P^{NP} (also known as Δ_2 in the hierarchy of classes) and L^{NP} (also denoted by $P^{NP[O(\log n)]}$ or Θ_2), which contain the decision problems which can be solved by applying, with a number of calls which is polynomial (respectively, logarithmic) with respect to the size of the instance, a subprogram able to solve an appropriate problem in *NP* (usually, an *NP*-complete problem); and the class *DP* [46] (or DIF^P [8] or BH_2 [38], [52], ...) as the class of languages (or problems) L such that there are two languages $L_1 \in NP$ and $L_2 \in$ co-*NP* satisfying $L = L_1 \cap L_2$. This class is not to be confused with $NP \cap$ co-*NP* (see the warning in, e.g., [45,

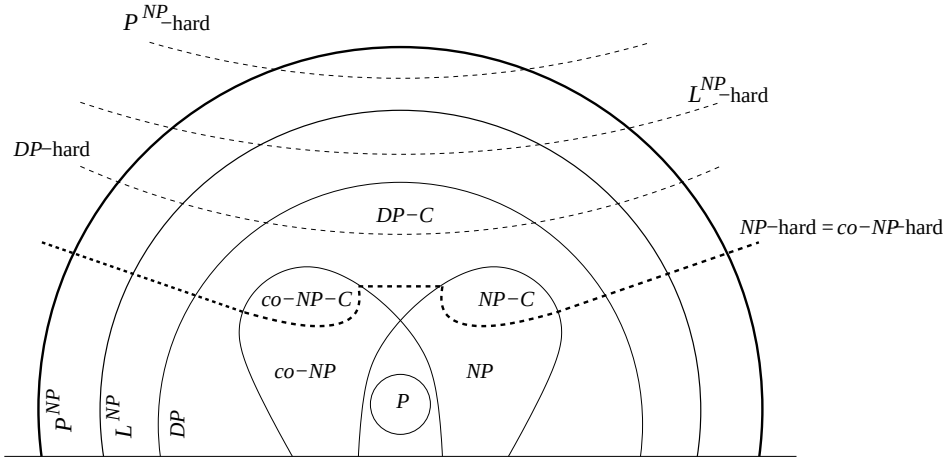

Figure 12: Some classes of complexity.

p. 412]); actually, DP contains $NP \cup$ co-NP and is contained in L^{NP}. See Figure 12.

Membership to P, NP, co-NP, DP, L^{NP} or P^{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 $P{=}NP$, they would lose their interest. But it is not known whether or where the classes of complexity collapse.

The problem SAT is one of the most well-known NP-complete problems [14], [21, p. 39, p. 46 and p. 259]. The following result is easy.

Proposition 51 *The problem U-SAT is NP-hard (and co-NP-hard by Remark 50), and belongs to the class DP.* △

Remark 52 *It is not known whether U-SAT is DP-complete: in [45, p. 415], it is said that "U-SAT is not believed to be DP-complete".*

We are now ready to investigate the problems of the uniqueness of identifying and LD codes.

10 Some Easy Preliminary Results

These results are as old as the definitions of identifying and LD codes.

Lemma 53 *(a) For any graph $G = (V, E)$ of order n and any integer $r \geq 1$, we have*

$$LD_r(G) \geq \lceil \log_2(n - LD_r(G) + 1) \rceil. \tag{3}$$

(b) For any integer $r \geq 1$ and any r-twin-free graph $G = (V, E)$ of order n, we have

$$i_r(G) \geq \lceil \log_2(n + 1) \rceil. \tag{4}$$

Proof. (a) Let C be any r-LD code in G. All the $n - |C|$ non-codewords $v \in V \setminus C$ must be given nonempty and distinct sets $B_r(v) \cap C$ constructed with the $|C|$ codewords, so $2^{|C|} - 1 \geq n - |C|$, from which (3) follows when C is optimal; (b) the argument is the same, but we have to consider all the n vertices $v \in V$, so $2^{|C|} - 1 \geq n$. △

Lemma 54 *Let $r \geq 2$ be any integer and $G = (V, E)$ be a graph.*

(a) A code C is 1-locating-dominating in G^r, the r-th power of G, if and only if it is r-locating-dominating in G.

(b) A code C is 1-identifying in G^r if and only if it is r-identifying in G.

Proof. (a) For every vertex $v \in V$, we have:

$$\{c \in C : d_G(v, c) \leq r\} = \{c \in C : d_{G^r}(v, c) \leq 1\},$$

so if for all $v \in V \setminus C$, the sets on the left-hand side of the equality are nonempty and distinct, then the sets on the right-side also are, and *vice-versa*; (b) same proof, for all $v \in V$. △

The following obvious lemma is often used implicitly; we give it without proof.

Lemma 55 *Let $r \geq 1$ be any integer and $G = (V, E)$ be a graph.*

(a) If C is r-locating-dominating in G, so is any set $S \supset C$.

(b) If C is r-identifying in G, so is any set $S \supset C$. △

11 Locating-Dominating Codes

After some necessary preliminary results, we are going to prove, for $r \geq 2$ and $q \geq 1$, the following polynomial reductions:

U-SAT $\rightarrow_p$ U-LDC$_1$ and U-SAT $\rightarrow_p$ U-OLDC$_1$ (Theorem 63),

U-SAT $\rightarrow_p$ U-LDC$_r$ and U-SAT $\rightarrow_p$ U-OLDC$_r$ (Theorem 64),

U-LDC$_{qr}$ $\rightarrow_p$ U-LDC$_q$ and U-OLDC$_{qr}$ $\rightarrow_p$ U-OLDC$_q$ (Proposition 67),

U-LDC$_1$ $\rightarrow_p$ U-SAT (Theorem 68).

The consequence of these reductions is that, for $r \geq 1$, U-LDC$_r$ and U-OLDC$_r$ are *NP*-hard, and that U-SAT and U-LDC$_r$ have equivalent complexities; as a result, U-LDC$_r$ belongs to *DP*. We shall also show that U-OLDC$_r$ belongs to the class L^{NP} (Proposition 71).

We do not have that U-OLDC$_r$ belongs to *DP* for lack of a polynomial reduction from U-OLDC$_1$ to U-SAT; we conjecture that such a reduction does not exist and that U-OLDC$_r \notin$ *DP* (see also Conclusion).

Also note that the polynomial reduction U-SAT $\rightarrow_p$ U-LDC$_1$ is a consequence of the chain of reductions U-SAT $\rightarrow_p$ U-LDC$_r$ $\rightarrow_p$ U-LDC$_1$; we still give Theorem 63 and its proof, because it constitutes a preliminary step for the proof of Theorem 64.

11.1 Preliminary Results

Lemma 56 *Let $h \geq 1$ and $r \geq 1$ be integers; let G be a graph of order $2^h - 1 + h$ with $LD_r(G) = h$. Then:*

(a) no vertex r-dominating 2^{h-1}, or fewer, vertices can belong to an optimal r-locating-dominating code in G;

(b) no vertex r-dominating $2^{h-1} + h + 1$, or more, vertices can belong to an optimal r-locating-dominating code in G.

Proof. Let C be any optimal r-LD code in G: $|C| = LD_r(G) = h$. Because there are $2^h - 1$ non-codewords, all the nonempty subsets of C coincide with all the nonempty, distinct sets $B_r(v) \cap C$, $v \in V \setminus C$. Then every codeword c appears exactly 2^{h-1} times in these subsets, which means that c r-dominates exactly 2^{h-1} non-codewords; since it r-dominates between one (itself) and h codewords, all in all it r-dominates between $2^{h-1} + 1$ and $2^{h-1} + h$ vertices, and (a) and (b) follow. △

The following lemma is easy, and we prove only its last assertion.

Lemma 57 *(a) The path $P_5 = x_1x_2x_3x_4x_5$ admits only one optimal 1-locating-dominating code, $C = \{x_2, x_4\}$.*

(b) If we construct the graph $G_A = (V_A, E_A)$ by adding to P_5 a vertex denoted by A together with the edge x_2A, then A is 1-dominated by x_2, but x_1 and A are not 1-separated by C.

(c) If G_A is plunged in a larger graph G^+ with only A linked to the outside, then every optimal 1-locating-dominating code C^+ in G^+ contains x_2 and x_4. At most one additional codeword, x_1 or A, may be necessary in $V_A \cap C^+$.

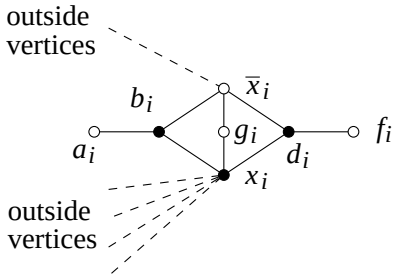

Figure 13: The graph G_i defined in Lemma 58. The black vertices form one of the two optimal 1-LD codes in G_i.

Proof. (c) Let C^+ be any optimal 1-LD code in G^+. (i) If A is a codeword, obviously C^+ contains x_2 and x_4, and no other codeword in P_5. (ii) The same is true if A is not a codeword, but is 1-dominated by at least one outside codeword. (iii) Otherwise, C^+ contains x_2, to 1-dominate A, it contains x_1, otherwise x_1 and A are not 1-separated by C^+, and it contains x_4, which is the only vertex which 1-dominates x_5 and 1-separates A and x_3 at the same time. △

The statements of the following lemma have been given, although in a different way, in [11, proof of Lemma 3.1]; for completeness, we give here the (easy) proof.

Lemma 58 *Let $G_i = (V_i, E_i)$ be the following graph: $V_i = \{x_i, \overline{x}_i, a_i, b_i, d_i, f_i, g_i\}$ and $E_i = \{a_ib_i, b_ix_i, b_i\overline{x}_i, x_id_i, \overline{x}_id_i, d_if_i, x_ig_i, \overline{x}_ig_i\}$, and C_i be a 1-locating-dominating code in G_i, see Figure 13. Then:*

(a) at least one of x_i and $\overline{x}_i$ belong to C_i;

(b) at least two more codewords necessarily belong to C_i, so that we have $LD_1(G_i) \geq 3$;

(c) we have $LD_1(G_i) = 3$, and $\{x_i, b_i, d_i\}$ and $\{\overline{x}_i, b_i, d_i\}$ are the only optimal 1-locating-dominating codes in G_i;

(d) if G_i is plunged in a larger graph G^+, with only x_i and $\overline{x}_i$ linked to the outside, then every 1-locating-dominating code in G^+ contains at least three codewords inside V_i.

Proof. (a) Because x_i and $\overline{x}_i$ have the same neighbours in G_i. (b) Because a_i and f_i must be 1-dominated by C_i, we have $|C_i \cap \{a_i, b_i\}| \geq 1$ and $|C_i \cap \{d_i, f_i\}| \geq 1$. Alternatively, use (3) in Lemma 53. (c) Assume that it is x_i that belongs to C_i. Then taking a_i and f_i would not 1-dominate $\overline{x}_i$, and

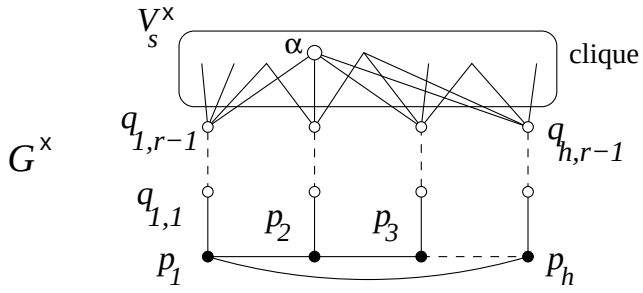

Figure 14: The graph $G^\times$ has a unique optimal r-LD code, $V_p^\times$.

taking a_i and d_i, or b_i and f_i, would not 1-separate $\overline{x}_i$ and f_i, or $\overline{x}_i$ and a_i, respectively; on the other hand, $\{x_i, b_i, d_i\}$ is 1-LD. (d) Only x_i and $\overline{x}_i$ can be 1-dominated by the outside, and a_i, f_i and g_i have to be 1-dominated by the code, so anyway at least three codewords are necessary inside V_i. △

The previous two lemmas will be used in the proof of Theorem 63. In particular, Lemma 57(a) gives the example of a graph, P_5, with a unique optimal 1-LD code. We want to have the same for any $r > 1$ (see Proposition 59(a)), in view of Theorem 64. We shall proceed as follows (see Figure 14):

We set $h = 2r + 1$, even if everything that follows also holds for any $h \geq 2r + 1$. Let $G_p^\times = (V_p^\times, E_p^\times)$ be the cycle $\mathcal{C}_h$ of length h, with $V_p^\times = \{p_i : 1 \leq i \leq h\}$. Then we construct $G_q^\times = (V_q^\times, E_q^\times)$, with $V_q^\times = \{q_{i,j} : 1 \leq i \leq h, 1 \leq j \leq r-1\}$ and $E_q^\times = \cup_{1 \leq i \leq h}\{q_{i,j}q_{i,j+1} : 1 \leq j \leq r-2\}$. The set of edges between $G_p^\times$ and $G_q^\times$ is $E_{p,q}^\times = \{p_i q_{i,1} : 1 \leq i \leq h\}$. Next, we construct $G_s^\times = (V_s^\times, E_s^\times)$ with $V_s^\times = \{s_i : 1 \leq i \leq 2^h - 1 - (r-1)h\}$ and $E_s^\times = \{s_{i_1}s_{i_2} : 1 \leq i_1 < i_2 \leq |V_s^\times|\}$, i.e., $G_s^\times$ is a clique.

We set $V^\times = V_p^\times \cup V_q^\times \cup V_s^\times$.

In order to define the set $E_{q,s}^\times$ of edges between $\{q_{i,r-1} : 1 \leq i \leq h\}$ and $V_s^\times$, we introduce, for every vertex $v \in V_q^\times \cup V_s^\times$, the *signature* of v as the set $B_r(v) \cap V_p^\times$ of the elements of the cycle that r-dominate v, and we wish to have nonempty and distinct signatures. Since

(a) the h vertices in $V_p^\times$ can provide $2^h - 1$ such signatures,
(b) $|V_q^\times \cup V_s^\times| = |V_q^\times| + |V_s^\times| = 2^h - 1$,
(c) the vertices in $V_q^\times$ have nonempty and different signatures (in particular, thanks to the fact that $h \geq 2r$, all the vertices $q_{i,1}$ have signatures of size $2r-1$),
(d) a vertex in $V_s^\times$ which is linked (respectively, not linked) to $q_{i,r-1}$ is at

distance equal to (respectively, greater than) r from p_i,

we can see that it is possible to construct $E^{\times}_{q,s}$ in such a way that the vertices in $V^{\times}_s$ have nonempty signatures which are different inside $V^{\times}_s$, and different from those for $V^{\times}_q$. In particular, in $V^{\times}_s$ there is a vertex which has signature equal to $V^{\times}_p$; we denote this vertex by α. Note also that we could not have more vertices with this signature property.

We set $E^{\times} = E^{\times}_p \cup E^{\times}_{p,q} \cup E^{\times}_q \cup E^{\times}_{q,s} \cup E^{\times}_s$ and $G^{\times} = (V^{\times}, E^{\times})$. The order of $G^{\times}$ is $n^{\times} = 2^h - 1 + h$.

We claim that, for a fixed $r \geq 2$ and $h = 2r + 1$, $C = V^{\times}_p$ is the *unique* optimal r-LD code in $G^{\times}$; we shall prove it by going through the following three easy facts.

Fact 1 *For any $r \geq 2$ and $h = 2r + 1$, the code $C = V^{\times}_p$ is an optimal r-locating-dominating code in $G^{\times}$.*

Proof. When $C = V^{\times}_p$, the signatures are the sets $B_r(v) \cap C$, for $v \in V^{\times} \setminus C$. By construction, they are all nonempty and distinct, hence C is r-LD. The optimality comes from (3) in Lemma 53. △

Fact 2 *For any $r \geq 2$ and $h = 2r + 1$, the graph $G^{\times}$ meets the conditions of Lemma 56.*

Proof. Because $LD_r(G^{\times}) = |V^{\times}_p| = h$ and $G^{\times}$ has order $2^h - 1 + h$. △

Fact 3 *For any $r \geq 2$ and $h = 2r + 1$, no vertex in $V^{\times}_q \cup V^{\times}_s$ can belong to any optimal r-locating-dominating code C in $G^{\times}$.*

Proof. Because $V^{\times}_s$ is a clique, every vertex $q_{i,j}$, $1 \leq i \leq h$, $1 \leq j \leq r-1$, is within distance r from every vertex in $V^{\times}_s$; so every $q_{i,j}$ r-dominates at least $|V^{\times}_s| = 2^h - 1 - (r-1)h$ vertices. This number is greater than $2^{h-1} + h + 1$ for $r \geq 2$ and $h = 2r + 1$. The same is true for the vertices in $V^{\times}_s$. We can conclude using Lemma 56(b). △

We are now ready to prove the following proposition.

Proposition 59 *Let $r \geq 2$ and $h = 2r + 1$. Then:*

(a) the only optimal r-locating-dominating code in $G^{\times}$ is $C = V^{\times}_p$;

(b) if $G^{\times}$ is plunged in a larger graph $G^{+} = (V^{+}, E^{+})$, with only α linked to the outside, then every optimal r-locating-dominating code in G^{+} contains $V^{\times}_p$.

Proof. (a) Now that Fact 3 has ruled out the vertices in $V_q^\times \cup V_s^\times$, the only possibility left is to take all the h codewords in $V_p^\times$.

(b) Let C be an optimal r-LD code in G^+, and let $|C \cap (V_s^\times \setminus \{\alpha\})| = X$ and $|C \cap V_p^\times| = Y$. If $Y = |V_p^\times|$, we are done, so we assume that $Y \leq h-1$. How does C r-separate the $2^h-(r-1)h-2-X$ vertices in $V_s^\times \setminus \{\alpha\}$ that need to be r-separated? Depending on its distance to α, a vertex outside $V^\times$ r-dominates either α alone, or all the vertices in $V_s^\times$ plus all the vertices $q_{i,j}$, $1 \leq i \leq h$, for some $j \geq 1$. This means that no outside codeword can r-separate the vertices in $V_s^\times \setminus \{\alpha\}$: this must be an inside-$V^\times$ job. But the vertices in $V_q^\times \cup V_s^\times$ cannot do it either, because every such vertex r-dominates all the vertices in $V_s^\times \setminus \{\alpha\}$; so the Y codewords in $V_p^\times$ must do it, and, according to whether the vertices in $V_s^\times \setminus \{\alpha\}$ are r-dominated by other codewords or not, we must have $2^Y - \varepsilon \geq 2^h - (r-1)h - 2 - X$, with $\varepsilon = 0$ or 1; since $Y \leq h-1$, this implies

$$2^{h-1} \leq (r-1)h + 2 + X - \varepsilon. \tag{5}$$

For $r \geq 2$ and $h = 2r+1$, the study of (5) shows that necessarily $X \geq h+2$. What is the role of these (at least) $h+2$ codewords belonging to $V_s^\times \setminus \{\alpha\}$?

(a) They contribute to r-dominate and r-separate some vertices in $V^+ \setminus V^\times$. From this perspective, all the vertices in $V_s^\times \setminus \{\alpha\}$ have an equivalent role towards $V^+ \setminus V^\times$. So one codeword in $V_s^\times \setminus \{\alpha\}$ is sufficient for this task.

(b) They contribute to r-dominate and r-separate some vertices in $V^\times$, and they themselves need not be r-separated from other vertices by the code; but we have already seen (Fact 1) that if we take the h vertices in $V_p^\times$ as codewords, then we can take care of all the vertices in $V^\times$.

(c) They contribute to r-separate some vertices in $V^+ \setminus V^\times$ from some vertices in $V^\times$. But the h vertices in $V_p^\times$ r-dominate all the vertices inside $V^\times$ and no vertex outside $V^\times$.

Therefore, if we take h codewords in $V_p^\times$ and one codeword in $V_s^\times \setminus \{\alpha\}$, we can do, with respect to the whole graph G^+, at least as well as with $X + Y \geq X \geq h+2$ codewords, contradicting the optimality of C. $\triangle$

The following lemma and its obvious corollary will be used for Proposition 70. They characterize the vertices belonging to at least one optimal r-LD code, through the comparison of two 1-location-domination numbers.

Lemma 60 *Let $G = (V, E)$ be a graph. For a given vertex $\alpha \in V$, we consider the following graph: $G_\alpha = (V_\alpha, E_\alpha)$, with*

$$V_\alpha = V \cup \{\beta_i : 1 \leq i \leq 6\},$$

$$E_\alpha = E \cup \{\alpha\beta_i : i \in \{1,4\}\} \cup \{\beta_i\beta_{i+1} : i \in \{1,2,4,5\}\},$$

where for $i \in \{1,\ldots,6\}$, $\beta_i \notin V$. *Then* α *belongs to at least one optimal* 1*-locating-dominating code in* G *if and only if* $LD_1(G) = LD_1(G_\alpha) - 2$.

Proof. Let $B_\alpha = \{\beta_i : 1 \leq i \leq 6\}$, and $B^*_\alpha = \{\beta_i : i \in \{2,3,5,6\}\}$.

First, we prove that α belongs to every optimal 1-LD code C_α in G_α: assume on the contrary that $\alpha \notin C_\alpha$; then obviously $|C_\alpha \cap B_\alpha| \geq 2+2$, and $(C_\alpha \setminus (C_\alpha \cap B_\alpha)) \cup \{\alpha, \beta_2, \beta_5\}$ is a 1-LD code in G_α, with fewer elements than C_α, a contradiction. So $\alpha \in (C_\alpha \cap V)$.

(a) Assume that α belongs to at least one optimal 1-LD code C in G. Then $C_\alpha = C \cup \{\beta_2, \beta_5\}$ is obviously 1-LD in G_α, and $LD_1(G_\alpha) \leq |C_\alpha| = LD_1(G) + 2$.

Consider now an optimal 1-LD code C_α in G_α. Obviously, we have $|C_\alpha \cap B^*_\alpha| \geq 1+1$, and so, if we set $C = C_\alpha \cap V$, we have: $|C| \leq LD_1(G_\alpha) - 2$. We have already established that $\alpha \in C_\alpha$, and thus $\alpha \in C$; now, we can see that it is sufficient, for any optimal 1-LD code in G_α, to have two codewords in B_α, namely β_2 and β_5, and therefore $|C| \geq LD_1(G_\alpha) - 2$. Now, no codeword in $C_\alpha \setminus C$ 1-dominates any vertex in V, and necessarily C is a 1-LD code in G, which proves that $LD_1(G) \leq |C| = LD_1(G_\alpha) - 2$.

So we can conclude that if α belongs to at least one optimal 1-LD code in G, then $LD_1(G) = LD_1(G_\alpha) - 2$.

(b) Assume now that $LD_1(G) = LD_1(G_\alpha) - 2$. Consider an optimal 1-LD code C_α in G_α, and let $C = C_\alpha \cap V$. Then $\alpha \in C_\alpha$ and, exactly as before in (a), $\alpha \in C$, $C_\alpha = C \cup \{\beta_2, \beta_5\}$, C is a 1-LD code in G and its size is $LD_1(G_\alpha) - 2 = LD_1(G)$, i.e., it is optimal (and contains α). △

Corollary 61 *Let* $r \geq 1$ *be any integer,* G *be a graph containing a vertex* α*, and* G^r *be the* r*-th power of* G*. We construct the graph* $(G^r)_\alpha$ *in the same way as in the previous lemma for* G*. Then* α *belongs to at least one optimal* r*-locating-dominating code in* G *if and only if* $LD_1(G^r) = LD_1((G^r)_\alpha) - 2$.

Proof. Use Lemmas 54(a) and 60. △

In the following proposition, we shall only use the fact that LDC_1 belongs to *NP*, for Propositions 70 and 71.

Proposition 62 *[13, for* $r = 1$*], [11] Let* $r \geq 1$ *be any integer. The decision problem* LDC_r *is NP-complete.* △

The proofs of Proposition 62 do not treat however the problem of the uniqueness of a solution.

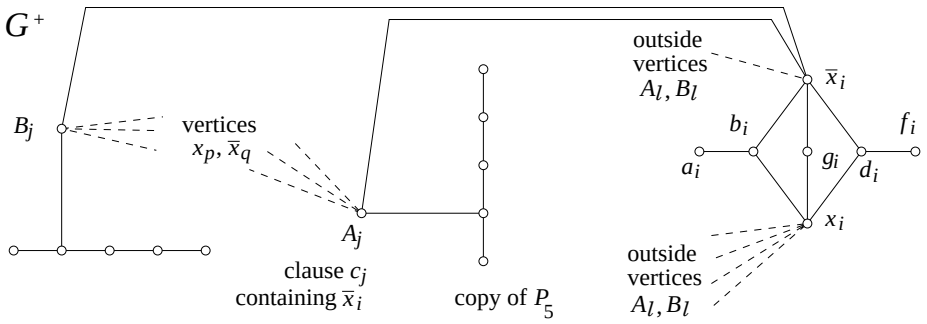

Figure 15: For $r = 1$, the graph G^+, constructed from a set of clauses.

11.2 Uniqueness of Locating-Dominating Code

11.2.1 From U-SAT to U-LDC$_1$ and U-OLDC$_1$

Theorem 63 *There exists a polynomial reduction from U-SAT to U-LDC$_1$ and to U-OLDC$_1$: U-SAT $\rightarrow_p$ U-LDC$_1$ and U-SAT $\rightarrow_p$ U-OLDC$_1$.*

Proof. We give a polynomial reduction starting from an instance of U-SAT, that is, a collection $\mathcal{C}$ of m clauses over a set $\mathcal{X}$ of n variables.

For each variable $x_i \in \mathcal{X}$, $1 \leq i \leq n$, we take the graph $G_i = (V_i, E_i)$ defined in Lemma 58, identifying the literals $x_i, \overline{x}_i$ to the vertices $x_i, \overline{x}_i$. For each clause c_j, containing ε_j literals, $\varepsilon_j \geq 2$, we create two vertices, A_j and B_j, and we link them to the ε_j vertices corresponding, in the graphs G_i, to the literals of c_j. We also take a copy of P_5, $P_5(A_j) = A_{j,1}A_{j,2}A_{j,3}A_{j,4}A_{j,5}$, and link A_j to $A_{j,2}$. We do the same for B_j and a second copy of P_5, $P_5(B_j) = B_{j,1}B_{j,2}B_{j,3}B_{j,4}B_{j,5}$.

We call this graph G^+, see Figure 15. The order of G^+ is $7n + 12m$. Because the extremities of the copies of P_5 must be 1-dominated by a codeword, and thanks to Lemma 58(d), we have: $LD_1(G^+) \geq 4m + 3n$. We set $k = 4m + 3n$.

We claim that there is a unique solution to SAT if and only if there is a unique optimal 1-LD code in G^+, and if and only if there is a unique 1-LD code of size at most k in G^+.

(1) Assume first that there is a unique truth assignment satisfying all the clauses. We construct the following code C: for $i \in \{1, \ldots, n\}$, among the vertices $x_i \in V_i$, $\overline{x}_i \in V_i$, we put in C the vertex x_i if the literal x_i has been set TRUE, the vertex $\overline{x}_i$ if the literal x_i is FALSE, and we add b_i and d_i, as well as the second and fourth vertices in each of the copies of P_5.

Then C is a 1-LD code in G^+: thanks to our preliminary observations (Lemmas 57 and 58), the only thing that remains to be checked is that for all $j \in \{1, \ldots, m\}$, the code C 1-separates A_j and $A_{j,1}$, B_j and $B_{j,1}$, and this is so because there is at least one true literal in the clause c_j, which means that A_j and B_j are 1-dominated by at least one codeword of type x_i or $\overline{x}_i$.

Moreover, $|C| = 3n + 4m = k$, which proves that it is optimal, and no vertex A_j nor B_j is a codeword. This implies that, once we have decided between x_i and $\overline{x}_i$, we have *no choice left* inside G_i if we want a code of size k (optimal): we *must* take b_i and d_i, because neither x_i nor $\overline{x}_i$ is 1-dominated by any outside codeword; the same is true for the copies of P_5, which *must* each contain their second and fourth vertices as only codewords.

Why is C unique? Suppose on the contrary that C^* is another 1-LD code of size $k = 3n + 4m$ in G^+. Then $|C^* \cap V_i| = 3$ for all $i \in \{1, \ldots, n\}$, and at most one of x_i and $\overline{x}_i$ is in C^*. Also, no vertex A_j or B_j is a codeword, and each copy of P_5 contains exactly two codewords, which are necessarily the second and the fourth ones. As another consequence, at least one of x_i and $\overline{x}_i$ is a codeword, because no codeword 1-separates them, and so exactly one of them belongs to C^*. This defines a valid truth assignment for $\mathcal{X}$, by setting $x_i = \mathrm{T}$ if $x_i \in C^*$, $x_i = \mathrm{F}$ if $\overline{x}_i \in C^*$. Since $C \neq C^*$, this assignment is different from the assignment used to build C. But the fact that, for all j, C^* 1-separates A_j and $A_{j,1}$, B_j and $B_{j,1}$, shows that there is a codeword x_i or $\overline{x}_i$ 1-dominating A_j and B_j, which means that the clause c_j is satisfied. Therefore, we have a second assignment satisfying the instance of SAT, a contradiction. We can conclude that both problems, U-LDC$_1$ and U-OLDC$_1$, also receive the answer YES.

(2) Assume next that the answer to U-SAT is NO: this may be either because no truth assignment satisfies the instance, or because at least two assignments do; in the latter case, this would lead, using the same argument as previously, to at least two (optimal) 1-LD codes (of size k), and a NO answer to both U-LDC$_1$ and U-OLDC$_1$. So we are left with the case when the set of clauses $\mathcal{C}$ cannot be satisfied. This implies that no 1-LD code of size k exists, for the same reason as in the previous paragraph with C^*; this suffices to prove that we have also a NO answer to U-LD$_1$, but we have to go further for U-OLD$_1$. Assume then that C is an optimal 1-LD code of unknown size $|C| > 4m + 3n$. For $1 \leq j \leq m$, let $\mathcal{A}_j = C \cap \{A_j, A_{j,i} : 1 \leq i \leq 5\}$ and $\mathcal{B}_j = C \cap \{B_j, B_{j,i} : 1 \leq i \leq 5\}$.

Suppose first that there is a j_0 such that A_{j_0} and B_{j_0} are not 1-dominated by any codeword x_i nor any codeword $\overline{x}_i$. Then $|\mathcal{A}_{j_0}| > 2$, and actually this set has size exactly three; the same is true for $\mathcal{B}_{j_0}$. Now we have several

optimal codes, because $C \cap (\mathcal{A}_{j_0} \cup \mathcal{B}_{j_0})$ can be equal to

$\{A_{j_0}, A_{j_0,2}, A_{j_0,4}, B_{j_0}, B_{j_0,2}, B_{j_0,4}\}$, or

$\{A_{j_0,1}, A_{j_0,2}, A_{j_0,4}, B_{j_0}, B_{j_0,2}, B_{j_0,4}\}$, or

$\{A_{j_0}, A_{j_0,2}, A_{j_0,4}, B_{j_0,1}, B_{j_0,2}, B_{j_0,4}\}$,

—but in general, not $\{A_{j_0,1}, A_{j_0,2}, A_{j_0,4}, B_{j_0,1}, B_{j_0,2}, B_{j_0,4}\}$, for this might affect the vertices x_i, $\overline{x}_i$ to which A_{j_0} and B_{j_0} are linked.

So from now on, we assume that all vertices A_j, B_j are 1-dominated by at least one codeword x_i or $\overline{x}_i$. Because the set of clauses cannot be satisfied, it is impossible that for all i, exactly one of x_i and $\overline{x}_i$ is a codeword, for this would lead to a valid truth assignment which would satisfy all the clauses. So there is a subscript i_0 such that either both x_{i_0} and $\overline{x}_{i_0}$ are codewords, or none is a codeword. If both are codewords, then $C \cap V_{i_0}$ contains x_{i_0}, $\overline{x}_{i_0}$ and can contain any combination with exactly one codeword among a_{i_0}, b_{i_0} and exactly one among d_{i_0} and f_{i_0}, yielding at least four optimal solutions. If none of x_{i_0}, $\overline{x}_{i_0}$ is a codeword, then they are 1-separated by some codeword(s) A_j, B_j; moreover, g_{i_0}, which must belong to C, 1-dominates both x_{i_0}, $\overline{x}_{i_0}$. Then again, we can have any of the four combinations with one codeword among a_{i_0}, b_{i_0} and one among d_{i_0} and f_{i_0}.

So in all cases, we do not have a unique optimal 1-LD code. △

11.2.2 Extension to $r \geq 2$

It seems difficult to go directly from $r = 1$ to the general case $r \geq 2$, and we start again from U-SAT, which does not change the final result; see [11, Rem. 5] about this possible difficulty.

Theorem 64 *Let $r \geq 2$ be any integer. There exists a polynomial reduction from U-SAT to U-LDC$_r$ and to U-OLDC$_r$: U-SAT $\rightarrow_p$ U-LDC$_r$ and U-SAT $\rightarrow_p$ U-OLDC$_r$.*

Proof. We give a polynomial reduction starting from an instance of U-SAT, i.e., a collection $\mathcal{C}$ of m clauses over a set $\mathcal{X}$ of n variables.

We take the graph G^+ constructed in the proof of Theorem 63 (cf. Figure 15), and rename it $G_I = (V_I, E_I)$, for Intermediate graph. Then, for each edge $e = uv \in E_I$, we "paste" $r-1$ copies of the graph $G^\times$ constructed for Proposition 59 (cf. Figure 14), by deleting the edge $e = uv$ and creating the edges $u\alpha_1$, $\alpha_1\alpha_2$, …, $\alpha_{r-1}v$, where the α_i's are copies of the vertex α in $G^\times$, see Figure 16: we shall say that the edge e is *dilated*. We denote by G^+ the graph thus constructed. Since r, hence $h = 2r + 1$, is fixed, the fact that $G^\times$ has order $2^h + h - 1$ does not affect the polynomiality of our construction with respect to $n + m$. We set $k = 3n + 4m + (r-1)h|E_I|$.

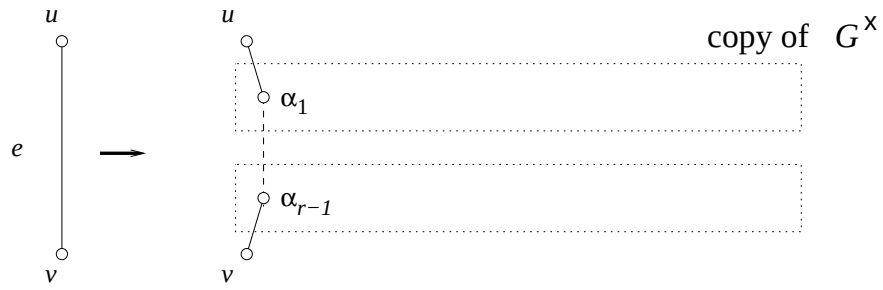

Figure 16: How the edge $e = uv \in E_I$ is dilated in the proof of Theorem 64.

The use of copies of $G^{\times}$ can be seen as a way of putting at distance r, in the graph G^+, the vertices which are at distance one in G_I, so that the vertices in V_I will behave with respect to each other in a way very similar to the case $r = 1$. It is still true that, in addition to at least h codewords taken in each copy of $G^{\times}$, at least three codewords are necessary in order to deal with the vertices in each V_i, and that at least two are necessary to cope with every copy of $\{x_1, \ldots, x_5\}$, the set of vertices in P_5. Consequently, any optimal r-LD code in G^+ has size at least $k = 3n + 4m + (r-1)h|E_I|$, the three terms corresponding respectively to (a) the sets V_i, $1 \leq i \leq n$, (b) the $2m$ copies of P_5 (which are now dilated copies), and (c) the $(r-1)$ copies of the graph $G^{\times}$ on each edge of the intermediate graph G_I.

One role of the codewords is to deal with the vertices in V_I, that is, if these are not codewords themselves, to r-dominate them, and to r-separate between them —the domination and separation inside the copies of $G^{\times}$ and the separation between vertices in V_I and vertices in the copies of $G^{\times}$ are already performed by the copies of the cycle $\mathcal{C}_h$, which are necessarily included in any optimal r-LD code in G^+, as already observed.

We can also make the following useful remark.

Remark 65 *Let C be any optimal r-locating-dominating code in G^+, $e = uv$ be any edge in E_I,and $G^{\times}$ be one of the copies pasted on e. If z is a codeword belonging to $G^{\times}$ and not to its cycle $\mathcal{C}_h$, then $(C \setminus \{z\}) \cup \{u\}$ or $(C \setminus \{z\}) \cup \{v\}$ is also an optimal r-locating-dominating code in G^+.*

Indeed, if (a) z r-dominates neither u nor v, then it can be spared and C is not optimal; if (b) z r-dominates u, not v, i.e., it r-separates u and v (and z cannot r-dominate any other vertex in V_I), then, when u or v becomes a codeword, u and v need not be r-separated anymore; if (c) z r-dominates both u and v, these two vertices will remain r-dominated by a common code-

word, u or v. In all cases, the fact that other vertices in V_I can now be r-dominated by the substitute codeword (u or v) does not change anything (cf. Lemma 55(a)).

We claim that there is a unique solution to SAT if and only if there is a unique optimal r-LD code in G^+, and if and only if there is a unique r-LD code of size at most k in G^+.

(1) Assume first that there is a unique truth assignment satisfying all the clauses. We construct the following code C: for $i \in \{1, \ldots, n\}$, among the vertices $x_i \in V_i$, $\overline{x}_i \in V_i$, we put in C the vertex x_i if the literal x_i has been set TRUE, the vertex $\overline{x}_i$ if the literal x_i is FALSE, and we add b_i and d_i, as well as the second and fourth vertices in each of the (dilated) copies of P_5. We add the cycle $\mathcal{C}_h$ in each copy of $G^\times$. Then C is an r-LD code in G^+: thanks to our preliminary observations, the only thing that remains to be checked is that for all $j \in \{1, \ldots, m\}$, the code C r-separates A_j and $A_{j,1}$, B_j and $B_{j,1}$, and this is so because there is at least one true literal in the clause c_j, which means that A_j and B_j are r-dominated by at least one codeword of type x_i or $\overline{x}_i$: everything develops exactly as in the case $r = 1$.

Moreover, $|C| = k$, which proves that it is optimal, and no vertex A_j nor B_j is a codeword. This implies that, once we have decided between x_i and $\overline{x}_i$, we have no choice left inside G_i: we *must* take b_i and d_i, because neither x_i nor $\overline{x}_i$ is r-dominated by any outside codeword. We have no choice in the copies of P_5 either: no pair of vertices in copies of $G^\times$ can r-dominate the first and last vertices, and r-separate the first, third and last, at the same time: only the second and fourth vertices can perform this.

Why is C unique? Suppose on the contrary that C^* is another (optimal) r-LD code (of size k) in G^+. Then each copy of $G^\times$ intersects C^* on exactly h vertices which are the vertices of the cycle $\mathcal{C}_h$; also, $|C^* \cap V_i| = 3$ for all $i \in \{1, \ldots, n\}$, and at most one of x_i and $\overline{x}_i$ is in C^*. Moreover, no vertex A_j nor B_j is a codeword. As a consequence, at least one of x_i and $\overline{x}_i$ is a codeword, because no codeword r-separates them, and so exactly one of them belongs to C^*. This defines a valid truth assignment for $\mathcal{X}$, by setting $x_i = \mathrm{T}$ if $x_i \in C^*$, $x_i = \mathrm{F}$ if $\overline{x}_i \in C^*$. Since $C \neq C^*$, this assignment is different from the assignment used to build C. But the fact that, for all j, C^* r-separates A_j and $A_{j,1}$, B_j and $B_{j,1}$, shows that there is a codeword x_i or $\overline{x}_i$ r-dominating A_j and B_j, which means that the clause is satisfied. Therefore, we have a second assignment satisfying the instance of SAT, a contradiction.

(2) Assume next that the answer to U-SAT is NO: this may be either

because no truth assignment satisfies the instance, or because at least two assignments do; in the latter case, this would lead, using the same argument as previously, to at least two optimal r-LD codes (of size k), and a NO answer to U-LDC$_r$ and U-OLDC$_r$. So we are left with the case when the set of clauses $\mathcal{C}$ cannot be satisfied. This implies that no r-LD code of size k exists; this ends the case of U-LDC$_r$ but we have to go on with U-OLDC$_r$: assume that C is an optimal r-LD code of unknown size $|C| > k$, with at least three codewords to deal with each V_i, at least two codewords to deal with each copy of P_5, at least h codewords in each copy of $G^\times$, and possibly vertices of type A_j, B_j. By Remark 65, if there is a codeword belonging to a copy of $G^\times$ and not to its cycle $\mathcal{C}_h$, then we have at least two optimal r-LD codes, and we are done. So from now on we assume that no copy of $G^\times$ contains codewords outside the cycle $\mathcal{C}_h$. Then C contains at least three codewords in each V_i, at least two codewords in each copy of P_5, exactly h codewords in each copy of $G^\times$, and possibly vertices of type A_j and B_j. Now the argument of the case $r = 1$ (Theorem 64) can be repeated almost word for word: first, we can exclude that there is a vertex A_{j_0} not r-dominated by any codeword x_i or $\overline{x}_i$; then we deal with the cases when both x_i and $\overline{x}_i$ are codewords, and when none of them is. In all cases, we have more than one optimal r-LD code in G^+. △

Corollary 66 *Let $r \geq 1$ be any integer. The decision problems U-LDC$_r$ and U-OLDC$_r$ are NP-hard.*

Proof. Because U-SAT is *NP*-hard (Proposition 51). △

Proposition 67 *Let $r \geq 2$ and $q \geq 1$ be any integers. There is a polynomial reduction from U-LDC$_{qr}$ to U-LDC$_q$ and from U-OLDC$_{qr}$ to U-OLDC$_q$: U-LDC$_{qr} \rightarrow_p$ U-LDC$_q$ and U-OLDC$_{qr} \rightarrow_p$ U-OLDC$_q$.*

As a particular case, we have U-LDC$_r \rightarrow_p$ U-LDC$_1$ and U-OLDC$_r \rightarrow_p$ U-OLDC$_1$.

Proof. Let (G, k) be an instance of U-LDC$_{qr}$ and G be an instance of U-OLDC$_{qr}$, for $r \geq 2$ and $q \geq 1$. The instance for U-LDC$_q$ is simply (G^r, k), and G^r for U-OLDC$_q$, where G^r is the r-th power of G: obviously, by Lemma 54(a), there is a unique q-LD code of size k in G^r if and only if there is a unique qr-LD code of size k in G, and there is a unique optimal q-LD code in G^r if and only if there is a unique optimal qr-LD code in G. △

11.2.3 An Upper Bound for the Complexity of U-LDC$_r$

Theorem 68 *There exists a polynomial reduction from U-LDC$_1$ to U-SAT: U-LDC$_1 \rightarrow_p$ U-SAT.*

Proof. In *Remark 32 of this Report*, we developed a general argument for this kind of reduction, with three types of clauses, one type for the description of the specific problem, here the fact that we want the code to be 1-LD, one type for the fact that we want a code of size at most k, and one type to break the multiple solutions. The same method will be applied for Theorem 83, with 1-identifying codes.

We start from an instance of U-LDC$_1$: a graph $G = (V, E)$ and an integer k, with $V = \{x^1, \ldots, x^{|V|}\}$; we assume that $|V| \geq 3$. We create the set of $k|V|$ variables $\mathcal{X} = \{x^i_m : 1 \leq i \leq |V|, 1 \leq m \leq k\}$ and the following clauses:

(a1) for each vertex $x^i \in V$ with neighbours $x^{n_1}, \ldots, x^{n_s}$ (where $s = s(x^i)$ is the degree of x^i), we take the clause of size $k(s+1)$:

$$c_{x^i} = \{x^i_1, x^i_2, \ldots, x^i_k, x^{n_1}_1, x^{n_1}_2, \ldots, x^{n_1}_k, x^{n_2}_1, \ldots, x^{n_2}_k, \ldots, x^{n_s}_1, \ldots, x^{n_s}_k\};$$

(a2) for each pair of vertices $x^i \in V$, $x^j \in V$, we consider the set $B_1(x^i)\Delta B_1(x^j) = \{x^{h_1}, x^{h_2}, \ldots, x^{h_\ell}\}$ (where ℓ depends on x^i and x^j) and we construct the clause $c_{x^i x^j}$:

$$\{x^i_1, x^i_2, \ldots, x^i_k, x^j_1, \ldots, x^j_k, x^{h_1}_1, x^{h_1}_2, \ldots, x^{h_1}_k, x^{h_2}_1, \ldots, x^{h_2}_k, \ldots, x^{h_\ell}_1, \ldots, x^{h_\ell}_k\};$$

we shall say that $\{x^i_1, x^i_2, \ldots, x^i_k, x^j_1, \ldots, x^j_k\}$ is the first part of $c_{x^i x^j}$ and $\{x^{h_1}_1, x^{h_1}_2, \ldots, x^{h_1}_k, x^{h_2}_1, \ldots, x^{h_2}_k, \ldots, x^{h_\ell}_1, \ldots, x^{h_\ell}_k\}$ its second part, which exists only when $\ell > 0$ and may contain variables also appearing in the first part, which is unimportant;

(b1) for $1 \leq m \leq k$ and $1 \leq h < \ell \leq |V|$, we construct clauses of size two: $\{\overline{x}^h_m, \overline{x}^\ell_m\}$;

(b2) for $1 \leq m < s \leq k$ and $1 \leq h \leq |V|$, we construct clauses of size two: $\{\overline{x}^h_m, \overline{x}^h_s\}$;

(c) for $1 \leq m < k$ and $1 < \ell \leq |V|$, for $1 \leq h < \ell$ and $m < s \leq k$, we construct clauses of size two: $\{\overline{x}^\ell_m, \overline{x}^h_s\}$.

All these clauses constitute the instance of U-SAT. Note that the number of variables and clauses is polynomial with respect to the order of G, since we may assume that $k \leq |V|$.

Assume that we have a unique 1-LD code of size k in G, $C = \{x^{p_1}, x^{p_2}, \ldots, x^{p_k}\}$, with $p_1 < p_2 < \ldots < p_k$. We can see that C is optimal (otherwise, any optimal 1-LD code contradicts our uniqueness assumption). Define the assignment $\mathcal{A}_1$ by $\mathcal{A}_1(x^{p_q}_q) = \mathrm{T}$ for $1 \leq q \leq k$, and all the other variables are set FALSE by $\mathcal{A}_1$. We claim that this assignment satisfies all the clauses; indeed:

(a1) at least one among x^i and its neighbours is a codeword, so the clause c_{x^i} is satisfied by $\mathcal{A}_1$.

(a2) (i) If at least one of x^i or x^j belongs to C, say $x^i = x^{p_q} \in C$, then the variable $x_q^{p_q} = x_q^i$ has been set True by $\mathcal{A}_1$ and the first part of the clause $c_{x^i x^j}$, hence the whole clause, is satisfied. (ii) If neither x^i nor x^j belongs to C, then, using the characterization given by (2), we can see that at least one x^{h_m} belongs to C, which guarantees that the second part of $c_{x^i x^j}$ is satisfied.

(b1) If a clause $\{\overline{x}_m^h, \overline{x}_m^\ell\}$ is not satisfied for some m, h, ℓ, this means that $\mathcal{A}_1(x_m^h) = \mathcal{A}_1(x_m^\ell) = \mathrm{T}$, i.e., two different vertices are the m-th element in C.

(b2) If $\{\overline{x}_m^h, \overline{x}_s^h\}$ is not satisfied, then x^h appears at least twice in C.

(c) If $\{\overline{x}_m^\ell, \overline{x}_s^h\}$ is not satisfied for some m, ℓ, with $h < \ell$ and $m < s$, then $\mathcal{A}_1(x_m^\ell) = \mathcal{A}_1(x_s^h) = \mathrm{T}$. This means that $x^\ell = x^{p_m}$ and $x^h = x^{p_s}$; so $\ell = p_m$, $h = p_s$. Now $h < \ell$ implies that $p_s < p_m$, but $m < s$ implies that $p_m < p_s$, a contradiction.

Is $\mathcal{A}_1$ unique? Assume on the contrary that another assignment, $\mathcal{A}_2$, also satisfies the constructed instance of U-SAT. We construct a new code C^+ by putting in C^+ the vertex x^h as soon as some variable x_m^h is set TRUE by $\mathcal{A}_2$.

Now the satisfaction, by $\mathcal{A}_2$, of the clause c_{x^i} in (a1) proves that every vertex in V is 1-dominated by C^+; the satisfaction of $c_{x^i x^j}$ from (a2) means that at least one vertex among $x^i, x^j, x^{h_1}, \ldots, x^{h_\ell}$ belongs to C^+. So for every pair of vertices x^i, x^j, either one of them is in the code, or an element in $B_1(x^i) \Delta B_1(x^j)$ is in the code. So every pair of non-codewords is 1-separated by C^+.

Therefore, we have just proved that C^+ is a 1-LD-code.

Using (b1) for $\mathcal{A}_2$, we can see that for each $m \in \{1, \ldots, k\}$ there is at most one variable with subscript m that is set TRUE by $\mathcal{A}_2$; this means that we have constructed a 1-LD code with (at most) k elements. Since such a code is unique by assumption, we can see that $\mathcal{A}_1$ and $\mathcal{A}_2$ "selected" the same k codewords: for each $p_q \in \{p_1, \ldots, p_k\}$, we already know that there is exactly one variable, $x_q^{p_q}$, set TRUE by $\mathcal{A}_1$, and, using (b2) after (b1) for $\mathcal{A}_2$, exactly one variable, say $x_t^{p_q}$, set TRUE by $\mathcal{A}_2$. It is now time to use (c) in order to prove that $q = t$ for every p_q, so that $\mathcal{A}_1$ and $\mathcal{A}_2$ actually coincide: indeed, assume on the contrary that for some $q \in \{1, \ldots, k\}$, we have $q \neq t$; we treat the case $1 \leq q < t \leq k$, the case $1 \leq t < q \leq k$ being analogous. If we consider the subscripts smaller than t, there must be one, say v, such that there is a superscript $p_u > p_q$ verifying $\mathcal{A}_2(x_v^{p_u}) = \mathrm{T}$. Now

the clause $\{\overline{x}_v^{p_u}, \overline{x}_t^{p_q}\}$ from (c) is not satisfied by $\mathcal{A}_2$, a contradiction.

So a YES answer to U-LDC$_1$ leads to a YES answer to U-SAT. Assume now that the answer to U-LDC$_1$ is negative. If it is negative because there are at least two 1-LD codes of size k, 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 1-LD code, and different 1-LD codes obviously lead to different assignments. On the other hand, if there is no 1-LD code of size k, then there can be no assignment satisfying U-SAT, because such an assignment would give a 1-LD code of size k, as we have seen above when dealing with $\mathcal{A}_2$. So in both cases, a NO answer to U-LDC$_1$ implies a NO answer to U-SAT. △

By Proposition 67 or its corollary, this immediately implies that there is a polynomial reduction from U-LDC$_r$ to U-SAT.

Theorem 69 *Let $r \geq 1$ be any integer. The problem U-LDC$_r$ has complexity equivalent to that of U-SAT.*

As a consequence, U-LDC$_r$ belongs to the class DP. △

Note that it could have been shown directly that U-LDC$_r$ belongs to *DP*.

11.2.4 Two Upper Bounds for the Complexity of U-OLDC$_r$

In [35] *(see Sections 6.2.3 and 7.2.4 in this Report)*, we give, for two problems structurally similar to U-OLDC$_r$, two upper bounds, the first one being weaker but constructive. We do not give the proofs of the following two results but refer to [35] instead. These proofs use Lemma 60, Corollary 61 and Proposition 62.

Proposition 70 *For $r \geq 1$, the decision problem U-OLDC$_r$ belongs to the class P^{NP}. In case of a YES answer, one can give the only optimal r-locating-dominating code within the same complexity.* △

Proposition 71 *For $r \geq 1$, the decision problem U-OLDC$_r$ belongs to L^{NP}.* △

12 Identifying Codes

The structure of this Section and its results are the same as Section 11 for LD-codes, although the preliminary graphs and arguments are quite different technically.

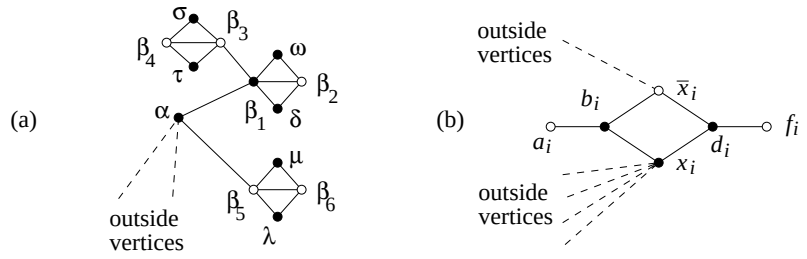

Figure 17: (a) The graph $G^\times$ of Lemma 72. Black vertices belong to any 1-identifying code in $G^\times$. (b) The graph G_i of Lemma 73, with an optimal 1-identifying code (black vertices).

12.1 Preliminary Results

Lemma 72 will be used in the proof of Theorem 79, and Lemma 73 in the proofs of Theorems 79 and 80.

Lemma 72 *Let $G^\times = (V^\times, E^\times)$ be the following graph:*

$$V^\times = \{\alpha, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \omega, \delta, \sigma, \tau, \lambda, \mu\},$$

$$E^\times = \{\alpha\beta_1, \beta_1\beta_2, \beta_1\delta, \beta_1\omega, \beta_2\delta, \beta_2\omega, \beta_1\beta_3, \beta_3\beta_4, \beta_3\sigma, \beta_3\tau, \beta_4\sigma, \beta_4\tau\} \cup$$
$$\cup \{\alpha\beta_5, \beta_5\beta_6, \beta_5\lambda, \beta_5\mu, \beta_6\lambda, \beta_6\mu\},$$

see Figure 17(a). Then $i_1(G^\times) = 8$, any 1-identifying code in $G^\times$ contains the set of vertices $C = \{\alpha, \beta_1, \omega, \delta, \sigma, \tau, \lambda, \mu\}$, and C is the only optimal 1-identifying code in $G^\times$.

If $G^\times$ is plunged in a larger graph G^+, with only α linked to the outside, then every 1-identifying code in G^+ contains C; the outside neighbours of α are 1-covered, not 1-separated, by α, and they are 1-separated from $V^\times$ by C.

Proof. Straightforward: α is the only vertex 1-separating β_5 and β_6; the same is true about β_1, for β_3, β_4; about ω, for β_2, δ; about δ, for β_2, ω; about σ, for β_4, τ; about τ, for β_4, σ; about λ, for β_6, μ; and about μ, for β_6, λ. So these eight vertices belong to any 1-identifying code, and since they obvioulsy constitute a 1-identifying code, this is the only optimal 1-identifying code.

When we consider G^+, all the above arguments still work. △

The statements of the following lemma have been given, although in a different way, in [12, proof of Th. 5.1]; for completeness, we give here the (easy) proof.

Lemma 73 *Let $G_i = (V_i, E_i)$ be the following graph: $V_i = \{x_i, \overline{x}_i, a_i, b_i, d_i, f_i\}$ and $E_i = \{a_ib_i, b_ix_i, b_i\overline{x}_i, x_id_i, \overline{x}_id_i, d_if_i\}$, and C_i be a 1-identifying code in G_i, see Figure 17(b). Then*

(a) At least one of x_i and $\overline{x}_i$ belong to C_i.

(b) At least two more codewords are necessary in C_i, so that $i_1(G_i) \geq 3$.

(c) We have $i_1(G_i) = 3$, and $\{x_i, b_i, d_i\}$ and $\{\overline{x}_i, b_i, d_i\}$ are the only optimal 1-identifying codes in G_i.

(d) If G_i is plunged in a larger graph G^+, with only x_i and $\overline{x}_i$ linked to the outside, then every 1-identifying code in G^+ contains at least three codewords in V_i, and one of them is x_i or $\overline{x}_i$.

Proof. (a) Because a_i and b_i, or d_i and f_i, must be 1-separated by C_i. (b) Because a_i and f_i must be 1-covered by C_i. Alternatively, use (4) in Lemma 53. (c) Assume that it is x_i that belongs to C_i. Then taking a_i and f_i would not 1-cover $\overline{x}_i$, and taking a_i and d_i, or b_i and f_i, would not 1-separate x_i and d_i, or x_i and b_i, respectively; on the other hand, $\{x_i, b_i, d_i\}$ is 1-identifying. (d) Only x_i and $\overline{x}_i$ can be 1-covered by the outside, and a_i and f_i still have to be 1-covered, a_i and b_i, and d_i and f_i still must be pairwise 1-separated by a codeword, requiring at least three inside codewords, one of them being x_i or $\overline{x}_i$. △

The *diapason* and *shortened diapason*, introduced in the following lemma and its corollary, will be used in the proof of Theorem 80.

Lemma 74 *[11, Lemma 2.1, Cor. 2.1] Let $r \geq 2$ be any integer. Let $T = \{t_1, t_2, \ldots, t_r\}$, $Y = \{y_1, y_2, \ldots, y_{2r+1}\}$, and $Z = \{z_1, z_2, \ldots, z_{2r+1}\}$. Let Δ be the graph in Figure 18, with vertex set $T \cup Y \cup Z$ and edge set*

$$\{t_it_{i+1} : i = 1, 2, \ldots, r-1\} \cup \{t_ry_1, t_rz_1\} \cup \{y_iy_{i+1}, z_iz_{i+1} : i = 1, 2, \ldots, 2r\}.$$

(a) The smallest r-identifying code in Δ, C_0, has size $2r + 2$ and is unique: it consists of the vertices $y_1, y_2, \ldots, y_r, y_{2r+1}, z_1, z_2, \ldots, z_r$, and z_{2r+1}.

(b) Any r-identifying code in Δ contains at least $2r+2$ elements in $Y \cup Z$; among them, the $2r$ vertices $y_1, y_2, \ldots, y_r$ and $z_1, z_2, \ldots, z_r$ must belong to any r-identifying code.

(c) Consider $r-1$ copies, $\Delta_1, \Delta_2, \ldots, \Delta_{r-1}$, of the graph Δ, and in each copy rename the "first" vertex t_1 by $t_{1,1}, t_{2,1}, \ldots, t_{r-1,1}$, and the other vertices accordingly. Build the graph Ω (cf. Figure 19) by taking these $r-1$ copies and adding the edges $t_{1,1}t_{2,1}, t_{2,1}t_{3,1}, \ldots, t_{r-2,1}t_{r-1,1}$. Then the smallest r-identifying code in Ω, C_1, has size $(r-1)(2r+2)$, is unique and consists of $r-1$ copies of the code C_0, one copy of C_0 in each copy of Δ.

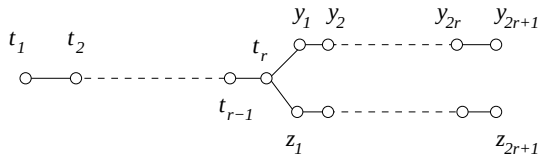

Figure 18: The diapason-graph Δ.

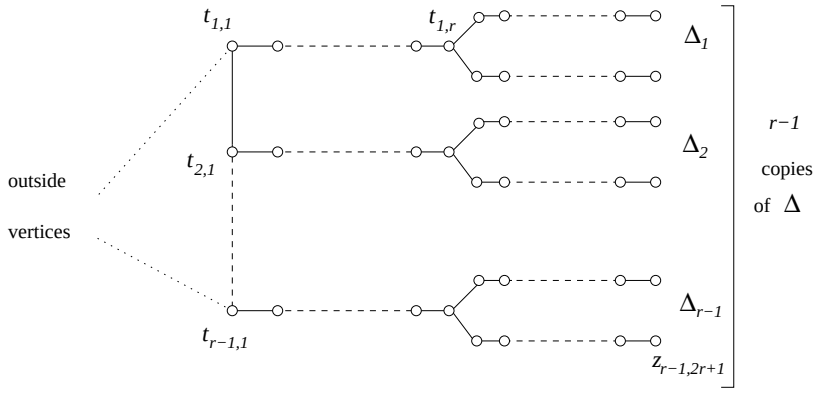

Figure 19: The graph Ω.

(d) If Ω is plunged in a larger graph G^+, with only $t_{1,1}$ and $t_{r-1,1}$ linked to the outside, then every r-identifying code in G^+ contains C_1; no outside vertex is r-covered by C_1. $\triangle$

We call the graph Δ a *diapason*. The sets Y and Z are the *branches*, the set T the *stem*, the vertex t_1 the *foot* of the diapason.

Corollary 75 *If we modify the previous lemma by considering a set T^- with one vertex less: $T^- = \{t_1, t_2, \ldots, t_{r-1}\}$ and t_{r-1} linked to y_1 and z_1, and if we denote by Δ^- the graph thus obtained, then the statements (a) and (b) of Lemma 74 remain true when we replace Δ by Δ^-.*

If Δ^- is plunged in a larger graph G^+, with only t_1 linked to the outside, then every r-identifying code in G^+ contains C_0; the outside neighbours of t_1 are the only outside vertices r-covered by C_0, they are not r-separated from one another by C_0, and they are r-separated by C_0 from all the vertices in Δ^-. $\triangle$

We call the graph Δ^- a *shortened diapason*.

Lemma 76 below, and its corollary, are similar to Lemma 60 and Corollary 61 on r-LD codes: they characterize the vertices belonging to at least one optimal r-identifying code, through the comparison of two 1-identification numbers. They will be used for Proposition 85. They are simplified versions of [32, Lemma 3] and [32, Cor. 4], respectively.

Lemma 76 *Let $G = (V, E)$ be a 1-twin-free graph. For a given vertex $\alpha \in V$, we construct the following graph $G_\alpha = (V_\alpha, E_\alpha)$:*

$$V_\alpha = V \cup \{\beta_1, \beta_2, \delta, \lambda\},\ E_\alpha = E \cup \{\alpha\beta_1, \beta_1\beta_2, \beta_1\delta, \beta_1\lambda, \beta_2\delta, \beta_2\lambda\},$$

where none of the vertices $\beta_1, \beta_2, \delta, \lambda$ belongs to V. Then α belongs to at least one optimal 1-identifying code in G if and only if $i_1(G) = i_1(G_\alpha) - 2$. △

Corollary 77 *Let $r \geq 1$ be any integer, G be an r-twin-free graph containing a vertex α, and G^r be the r-th power of G. We construct the graph $(G^r)_\alpha$ in the same way as in the previous lemma for G. Then α belongs to at least one optimal r-identifying code in G if and only if $i_1(G^r) = i_1((G^r)_\alpha) - 2$.* △

In the following proposition, we shall only use the fact that IdC_1 belongs to *NP* (see Propositions 85 and 86).

Proposition 78 *[12, for $r = 1$], [11] Let $r \geq 1$ be any integer. The decision problem IdC_r is NP-complete.* △

But the proofs for Proposition 78 do not deal with the problem of the uniqueness of a solution.

12.2 Uniqueness of Identifying Code

12.2.1 From U-SAT to U-IdC$_1$ and U-OIdC$_1$

Theorem 79 *There exists a polynomial reduction from U-SAT to U-IdC$_1$ and to U-OIdC$_1$: U-SAT $\rightarrow_p$ U-IdC$_1$ and U-SAT $\rightarrow_p$ U-OIdC$_1$.*

Proof. This proof is inspired by that of the *NP*-completeness of IdC_1 in [12].

We give a polynomial reduction starting from an instance of U-SAT, that is, a collection $\mathcal{C}$ of m clauses over a set $\mathcal{X}$ of n variables.

For each variable $x_i \in \mathcal{X}$, $1 \leq i \leq n$, we take the graph $G_i = (V_i, E_i)$ defined in Lemma 73. For each clause c_j, containing ε_j literals, $\varepsilon_j \geq 2$, we create two vertices, A_j and B_j, and we link A_j to the ε_j vertices corresponding, in the graphs G_i, to the literals of c_j. Finally we link the vertices A_j

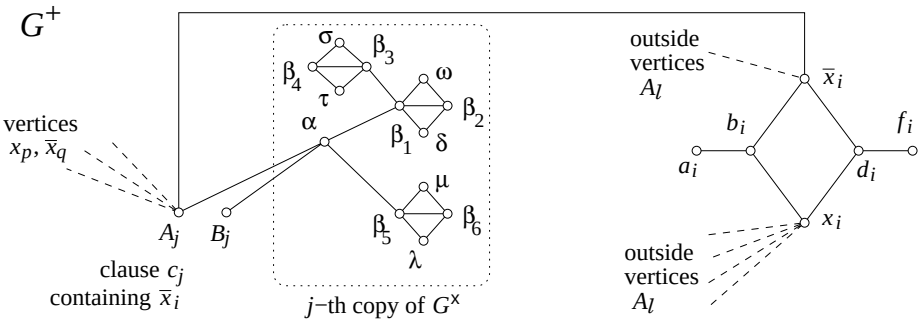

Figure 20: For $r = 1$, the graph G^+, constructed from a set of clauses.

and B_j to one copy of the graph $G^\times$ defined in Lemma 72, one different copy for each couple (A_j, B_j), by creating the edges $A_j\alpha$ and $B_j\alpha$ (or rather: we use the j-th copy of α); we call this graph G^+, see Figure 20. The order of G^+ is $6n + 15m$. Note that each pair of vertices A_j, B_j, $1 \leq j \leq m$, is 1-covered by a copy of α (which belongs necessarily to any 1-identifying code in G^+, see Lemma 72). By Lemmas 72 and 73, we have $i_1(G^+) \geq 8m + 3n$. We set $k = 8m + 3n$.

We claim that there is a unique solution to SAT if and only if there is a unique optimal 1-identifiying code in G^+, and if and only if there is a unique 1-identifying code of size at most k in G^+.

(1) Assume first that there is a unique truth assignment satisfying all the clauses. We construct the following code C: for $i \in \{1, \ldots, n\}$, among the vertices $x_i \in V_i$, $\overline{x}_i \in V_i$, we put in C the vertex x_i if the literal x_i has been set TRUE, the vertex $\overline{x}_i$ if the literal x_i is FALSE, and we add b_i and d_i. We add all the copies of the vertices α, β_1, ω, δ, σ, τ, λ and μ. Then C is a 1-identifying code in G^+: thanks to all our preliminary observations (Lemmas 72 and 73), the only thing that remains to be checked is that for all $j \in \{1, \ldots, m\}$, the vertices A_j and B_j are 1-separated by C. And this is so because there is at least one true literal in the clause c_j. Moreover, $|C| = k$, which proves that it is optimal. We can also see that, once we have decided between x_i and $\overline{x}_i$, we have no choice left inside G_i: we *must* take b_i and d_i, because neither x_i nor $\overline{x}_i$ is 1-covered by outside codewords, due to the fact that no vertex A_j can be a codeword, by an argument of cardinality.

Why is C unique? Suppose on the contrary that C^* is another 1-identifying code of size k in G^+. Then $|C^* \cap V_i| = 3$ for all $i \in \{1, \ldots, n\}$, and exactly one of x_i and $\overline{x}_i$ is in C^*. This defines a valid truth assignment

for $\mathcal{X}$, by setting $x_i = \mathrm{T}$ if $x_i \in C^*$, $x_i = \mathrm{F}$ if $\overline{x}_i \in C^*$. Since $C \neq C^*$, this assignment is different from the assignment used to build C. But the fact that C^* 1-separates A_j and B_j for all j shows that there is a codeword x_i or $\overline{x}_i$ 1-covering A_j, which means that the clause is satisfied. Therefore, we have a second assignment satisfying the instance of SAT, a contradiction.

(2) Assume next that the answer to U-SAT is NO: this may be either because no truth assignment satisfies the instance, or because at least two assignments do; in the latter case, this would lead, using the same argument as previously, to at least two optimal 1-identifying codes (of size k), and a NO answer to U-IdC$_1$ and U-OIdC$_1$. So we are left with the case when the set of clauses $\mathcal{C}$ cannot be satisfied. This implies that no 1-identifying code of size k exists; the case U-IdC$_1$ is closed, and, to go on with the problem U-OIdC$_1$, we assume that C is an optimal 1-identifying code of unknown size $|C| > 8m + 3n$. We know that each copy of $G^\times$ contains at least eight codewords, and each G_i at least three codewords. Where can the extra codeword(s) be? Any additional codeword in a copy of $G^\times$ is useless with respect to 1-identification and can be saved. If there are five or six codewords in a G_i, at least one can be saved; assume next that there are four of them: (a) if both x_i and $\overline{x}_i$ are codewords, then, e.g., $C \cap V_i = \{x_i, \overline{x}_i, b_i, d_i\}$ or $C \cap V_i = \{x_i, \overline{x}_i, b_i, f_i\}$ can be part of an optimal solution; (b) if only one of x_i and $\overline{x}_i$, say x_i, is a codeword, then there are also several possibilities for $C \cap V_i$, such as $\{x_i, b_i, d_i, a_i\}$ and $\{x_i, b_i, d_i, f_i\}$. So we can conclude that there are eight codewords in each copy of $G^\times$, three codewords in each G_i and the extra codewords are among the vertices A_j, B_j. If for some j, $B_j \in C$ and $A_j \notin C$, then B_j serves as a codeword only to 1-separate itself from A_j, but this can be done by A_j, so $(C \setminus \{B_j\}) \cup \{A_j\}$ would be another optimal 1-identifying code. So we are left with the case $A_j \in C$. Then A$_j$ 1-covers one vertex x_i or $\overline{x}_i$, say x_i, and then both $C \cap V_i = \{\overline{x}_i, b_i, d_i\}$ and $C \cap V_i = \{\overline{x}_i, a_i, f_i\}$ are possible. In all cases, we have proved that there are several optimal 1-identifying codes in G^+, i.e., we have a NO answer to the constructed instance of U-OIdC$_1$. △

12.2.2 Extension to $r \geq 2$

As for LD codes, we do not go directly from $r = 1$ to $r \geq 2$, but start again from U-SAT, which does not change the final result; see [11, Rem. 2] about this difficulty.

Theorem 80 *Let $r \geq 2$ be any integer. There exists a polynomial reduction from U-SAT to U-IdC$_r$ and to U-OIdC$_r$: U-SAT $\to_p$ U-IdC$_r$ and U-SAT $\to_p$ U-OIdC$_r$.*

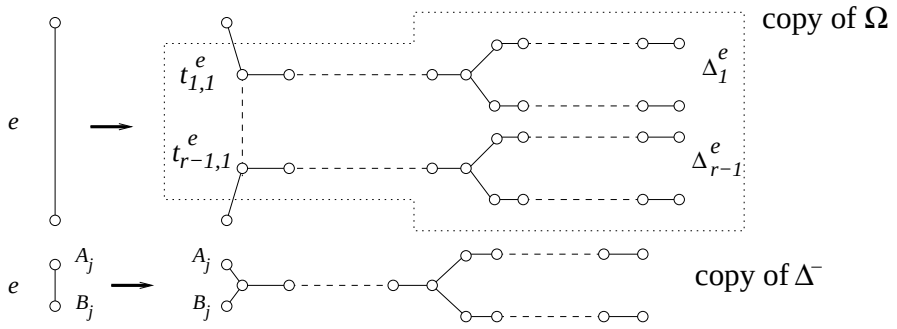

Figure 21: How the edge $e \in E_I$ is transformed in the proof of Theorem 80. If e is a membership edge or an edge in G_i, we use a copy of Ω; if $e = A_jB_j$, we use a copy of Δ^-.

Proof. This proof is inspired by that of the *NP*-completeness of IdC_r in [11].

We give a polynomial reduction starting from an instance of U-SAT, i.e., a collection $\mathcal{C}$ of m clauses over a set $\mathcal{X}$ of n variables.

In a first step, for each variable $x_i \in \mathcal{X}$, $1 \leq i \leq n$, we take the graph $G_i = (V_i, E_i)$ defined in Lemma 73. For each clause c_j, containing ε_j literals, $\varepsilon_j \geq 2$, we create two vertices, A_j and B_j, and the edge A_jB_j, and we link A_j to the ε_j vertices corresponding, in the graphs G_i, to the literals of c_j; we call these edges "membership edges". So far, we have constructed an intermediate graph, $G_I = (V_I, E_I)$.

In a second step (see Figure 21), for each membership edge and for each edge in the graphs G_i, we "paste" one copy of the graph Ω defined in Lemma 74, which is equivalent to pasting $r-1$ copies of the diapason Δ; and for each edge A_jB_j, $1 \leq j \leq m$, we paste *one* copy of the *shortened* diapason Δ^- defined in Corollary 75. We denote by G^+ the graph thus constructed, and set $k = 3n + (r-1)(2r+2)(|E_I| - m) + m(2r+2)$. The order of G^+ is $(6n + 2m) + (|E_I| - m)(r-1)(5r+2) + m(5r+1)$: the transformation is polynomial indeed.

The diapasons can be seen as a way of putting at distance r, in the graph G^+, the vertices in V_i, $1 \leq i \leq n$, and $\{A_j : 1 \leq j \leq m\}$ which are at distance one from one another in G_I. And so these vertices will behave with respect to each other in a way very similar to the case $r = 1$. In particular, it is still true that, in addition to codewords taken in the branches of the diapasons, at least three codewords are necessary to deal with the vertices in each V_i. Consequently, by Lemma 74(b), any optimal r-identifying code

in G^+ has size at least $3n + (r-1)(2r+2)(|E_I| - m) + m(2r+2) = k$, the three terms corresponding respectively to (a) the sets V_i, $1 \leq i \leq n$, (b) the $r-1$ copies of the diapason on each edge which is not A_jB_j, and (c) the copy of the shortened diapason on each edge A_jB_j, $1 \leq j \leq m$.

The role of the copies of the shortened diapason is to r-cover A_j and B_j without r-separating them, and to r-separate A_j and B_j from the other vertices belonging to V_I.

After these introductory observations, we can conclude that, in any r-identifying code in G^+, the role of the codewords which do not belong to the branches of the diapasons, is (a) to r-separate A_j from B_j, for all $j \in \{1, \ldots, m\}$; (b) to r-cover all the vertices in V_i, and to r-separate them, for all $i \in \{1, \ldots, n\}$.

We claim that there is a unique solution to SAT if and only if there is a unique optimal r-identifiying code in G^+, and if and only if there is a unique r-identifying code of size at most k in G^+.

(1) Assume first that there is a unique truth assignment satisfying all the clauses. We construct the following code C: for $i \in \{1, \ldots, n\}$, among the vertices $x_i \in V_i$, $\overline{x}_i \in V_i$, we put in C the vertex x_i if the literal x_i has been set TRUE, the vertex $\overline{x}_i$ if the literal x_i is FALSE, we add b_i and d_i, and we also take the unique optimal r-identifying codes in all the copies of Ω and Δ^-. Then, as in the case $r = 1$, we can check that C is an r-identifying code in G^+; in particular, for all $j \in \{1, \ldots, m\}$, the vertices A_j and B_j are r-separated by C, because there is at least one true literal in the clause c_j. Also, the code C has the right size and is optimal. We can also see that, once we know that, say, $x_i \in C$, we have *no choice left* for the completion of the code, because a_i, f_i and $\overline{x}_i$ must be r-covered (the latter because no A_j is a codeword), and x_i, b_i and d_i must be r-separated by the code. So b_i and d_i necessarily are the remaining two codewords in V_i, for all $i \in \{1, \ldots, n\}$.

Why is C unique? Suppose on the contrary that C^* is another r-identifying code, with $|C^*| = |C|$. Then for all $i \in \{1, \ldots, n\}$, exactly three codewords take care of V_i, and C^* contains b_i, d_i and exactly one of x_i and $\overline{x}_i$. This defines a valid truth assignment for $\mathcal{X}$, by setting $x_i = \mathrm{T}$ if $x_i \in C^*$, $x_i = \mathrm{F}$ if $\overline{x}_i \in C^*$. Since $C \neq C^*$, this assignment is different from the assignment used to build C. But the fact that C^* r-separates A_j and B_j for all j shows that there is one codeword x_i or $\overline{x}_i$ r-covering A_j, which means that the clause is satisfied. Therefore, we have a second assignment satisfying the instance of SAT, a contradiction.

(2) Assume next that the answer to U-SAT is NO: this may be either because no truth assignment satisfies the instance, or because at least two assignments do; in the latter case, this would lead, using the same argument

as previously, to at least two optimal r-identifying codes (of size k), and a NO answer to U-IdC_r and U-OIdC_r. So we are left with the case when the set of clauses $\mathcal{C}$ cannot be satisfied. This implies that no r-identifying code of size k exists: we have ended the case U-IdC_r; next, we assume that C is an optimal r-identifying code of unknown size $|C| > k$. We know that each copy of Ω or of Δ^- contains at least $(r-1)(2r+2)$ or $2r+2$ codewords, respectively, and that each V_i requires at least three codewords. Now, where can the extra codeword(s) be?

Note that, unfortunately, Remark 65 cannot be adapted to the present construction with pasted diapasons for r-identifying codes, because in the case (b) of the Remark, when the codeword z belonging to a diapason r-separates u and v and is replaced by u or v, then u and v are not r-separated anymore. This did not matter with LD-codes, but it does for identifying codes.

Any vertex B_j is a useless codeword, because, even in the case $r = 2$, it r-covers both A_j and itself. This is also true for all the codewords that would be on the branches of any diapason (apart from the codewords $y_1, y_2, \dots y_r$, y_{2r+1} and $z_1, z_2, \dots, z_r,\ z_{2r+1}$), as well as for codewords on the stem of a shortened diapason (for they all r-cover both A_j and B_j).

Let us now consider the case of a codeword on the stem of a diapason pasted on an edge $e = uv$: this edge is either a membership edge or an edge in some G_i; the (only) role of this codeword is either to r-cover exactly one of u an v, and consequently to r-separate u from v, or to r-cover both u and v, and consequently to r-separate them from other vertices in V_I. We distinguish between three cases. In each case, our goal is to show that one codeword can be spared (contradiction with the optimality of C) or that several optimal codes are possible.

(i) $r = 2$. The two vertices on the stem of the unique diapason pasted on e both 2-cover u and v, so at most one of them is necessary in the code, and they are interchangeable.

(ii) $r \geq 4$. (a) All the feet of the $r-1$ diapasons r-cover u and v, so at most one of them is necessary in the code, and they are interchangeable. (b) If, say, u is linked to $t_{1,1}$ and v to $t_{r-1,1}$, then $d_{G^+}(t_{1,r}, u) = r$, $d_{G^+}(t_{1,r-1}, u) = r-1$, $d_{G^+}(t_{1,r}, v) = 2r-2$, and $d_{G^+}(t_{1,r-1}, v) = 2r-3 > r$, so there are at least two vertices on the first stem which r-cover u, not v, at most one of them is necessary in the code, and they are interchangeable. The same is true by symmetry for $t_{r-1,r}$ and $t_{r-1,r-1}$.

(iii) $r = 3$. (a) The feet of the two diapasons 3-cover u and v, and the conclusion is the same as previously. (b) Without loss of generality, we assume that we are in the following case: the codeword is $t_{1,3}$; it is the *only*

vertex in Ω which 3-covers u, not v, so it 3-separates these two vertices. If $u = A_j$ for some $j \in \{1, 2, \ldots, m\}$, then it is 3-covered and 3-separated from v by the shortened diapason, and $t_{1,3}$ can be spared. So u is one of the six vertices in V_i for some $i \in \{1, 2, \ldots, n\}$. If $u \in \{x_i, \overline{x}_i, b_i, d_i\}$, that is, if u has degree at least two in G_I, then we can replace $t_{1,3}$ by another vertex suitably chosen in a diapason pasted on another edge incident to u in G_I. So we are left with the case when, say, $u = a_i$, and so $v = b_i$. But b_i is 3-covered by at least one codeword. (b1) This codeword also 3-covers a_i. Then $t_{1,3}$ can be replaced in C by a vertex 3-covering b_i, not a_i: this choice also allows to have a_i and b_i 3-covered and 3-separated by C. (b2) The codeword 3-covering b_i does not 3-cover a_i. Then $t_{1,3}$ can be replaced in C by, e.g., $t_{1,2}$, because a_i and b_i are already 3-separated by C.

So in all cases, we have at least two possible optimal codes, and we can assume from now on that each copy of Ω contains exactly $(r-1)(2r+2)$ codewords, and each copy of Δ^- exactly $2r+2$ codewords.

The case when there are four (or more) codewords in a component G_i can be treated exactly like the case $r = 1$, as if the copies of Ω did not exist. This is also true if each G_i has exactly three codewords, and one extra codeword is on some A_j, because then A_j r-covers some x_i or $\overline{x}_i$. In all cases, there is more than one possibility for $V_i \cap C$.

In conclusion, whenever there are more than k codewords in an optimal code, there are several possible optimal codes, and the answer to U-OIdC$_r$ is NO. △

Corollary 81 *Let $r \geq 1$ be any integer. The decision problems U-IdC$_r$ and U-OIdC$_r$ are NP-hard.* △

Proposition 82 *Let $r \geq 2$ and $q \geq 1$ be any integers. There is a polynomial reduction from U-IdC$_{qr}$ to U-IdC$_q$ and from U-OIdC$_{qr}$ to U-OIdC$_q$: U-IdC$_{qr}$ $\rightarrow_p$ U-IdC$_q$ and U-OIdC$_{qr}$ $\rightarrow_p$ U-OIdC$_q$.*

As a particular case, we have U-IdC$_r$ $\rightarrow_p$ U-IdC$_1$ and U-OIdC$_r$ $\rightarrow_p$ U-OIdC$_1$.

Proof. See the proof of Proposition 67 and Lemma 54(b). △

12.2.3 An Upper Bound for the Complexity of U-IdC$_r$

Theorem 83 *There exists a polynomial reduction from U-IdC$_1$ to U-SAT: U-IdC$_1$ $\rightarrow_p$ U-SAT.*

Proof. We refer to the proof of Theorem 68, and we give here only the clauses that describe the identification problem. These clauses are constructed in the following way:

(a1) for each vertex $x^i \in V$ with neighbours $x^{n_1}, \ldots, x^{n_s}$, we take the clause of size $k(s+1)$:

$$\{x_1^i, x_2^i, \ldots, x_k^i, x_1^{n_1}, x_2^{n_1}, \ldots, x_k^{n_1}, x_1^{n_2}, \ldots, x_k^{n_2}, \ldots, x_1^{n_s}, \ldots, x_k^{n_s}\};$$

(a2) for each pair of vertices $x^i \in V$, $x^j \in V$, we consider the set $B_1(x^i) \Delta B_1(x^j) = \{x^{h_1}, x^{h_2}, \ldots, x^{h_\ell}\}$; by the assumption that G is 1-twin-free, we have $\ell > 0$. Then we simply take the clause

$$\{x_1^{h_1}, x_2^{h_1}, \ldots, x_k^{h_1}, x_1^{h_2}, \ldots, x_k^{h_2}, \ldots, x_1^{h_\ell}, \ldots, x_k^{h_\ell}\},$$

which is the second part of the clause $c_{x^i x^j}$ in the aforementioned proof. Then the argument goes exactly like for Theorem 68, in particular thanks to the characterization given by (2). △

By Proposition 82 or its corollary, this immediately implies that there is a polynomial reduction from U-IdC$_r$ to U-SAT.

Theorem 84 *Let $r \geq 1$ be any integer. The problem U-IdC$_r$ has complexity equivalent to that of U-SAT.*

As a consequence, U-IdC$_r$ belongs to the class DP. △

12.2.4 Two Upper Bounds for the Complexity of U-OIdC$_r$

Exactly as in Section 11.2.4, we give, without proof, two results on U-OIdC$_r$; this time, Lemma 76, Corollary 77 and Proposition 78 are used.

Proposition 85 *For $r \geq 1$, the decision problem U-OIdC$_r$ belongs to the class P^{NP}. In case of a YES answer, one can give the only optimal r-identifying code within the same complexity.* △

Proposition 86 *For $r \geq 1$, the decision problem U-OIdC$_r$ belongs to L^{NP}.* △

13 Conclusion

We have established that the four decision problems U-LDC$_r$, U-IdC$_r$, U-OLDC$_r$ and U-OIdC$_r$ are, for any fixed $r \geq 1$, *NP*-hard, and that the two problems U-OLDC$_r$ and U-OIdC$_r$ belong to the class L^{NP}. For U-LDC$_r$ and

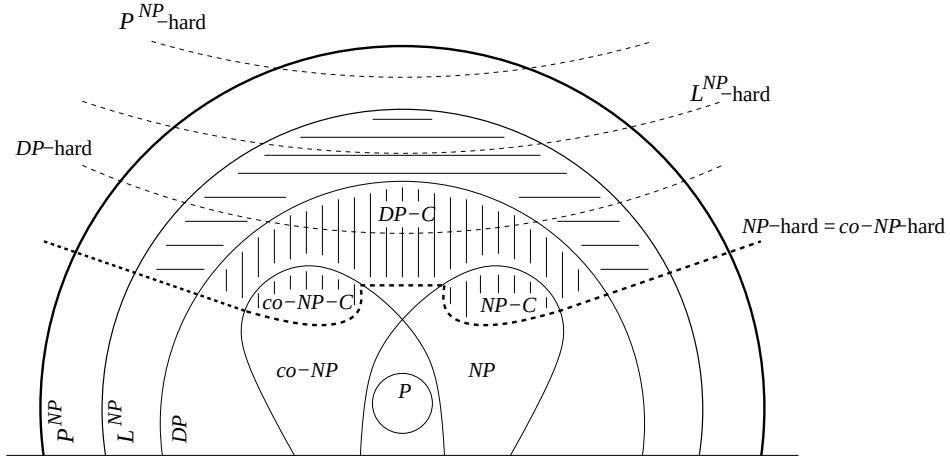

Figure 22: Some classes of complexity: Figure 12 re-visited.

U-IdC$_r$, we could go further and prove that they are *equivalent* to U-SAT and therefore belong to the class *DP*.

Conjecture Neither U-OLDC$_r$ nor U-OIdC$_r$ belong to *DP*.

Open Problem Give a better location, in the classes of complexity, for the problems U-LDC$_r$, U-IdC$_r$, U-OLDC$_r$ and U-OIdC$_r$.

We can see in Figure 22 that U-SAT, U-LDC$_r$ and U-IdC$_r$ are in the vertically hatched region, but probably not in *DP-C*, whereas U-OLDC$_r$ and U-OIdC$_r$ are somewhere in the region that is hatched horizontally or vertically.

In [9], a characterization of the trees which admit a unique optimal 1-LD code is given.

Open Problems Extend this study to other classes of graphs; to any integer $r \geq 1$; to identifying codes. What is the complexity of the sub-problem of U-OLDC$_1$ when the instance is any tree.

In [8], the authors wonder whether

(A) U-SAT is *NP*-hard, but here we believe that what they mean is: does there exist a *polynomial* reduction from an *NP*-complete problem to U-SAT ? i.e., they use the *second* definition of *NP*-hardness;

finally, they show that (A) is true if and only if

(B) U-SAT is *DP*-complete.

So, if one is careless and considers that U-SAT is *NP*-hard without checking according to which definition, one might easily jump too hastily to the conclusion that U-SAT is *DP*-complete.

On the Complexity of Determining Whether there is a Unique Hamiltonian Cycle or Path

Olivier Hudry
LTCI, Télécom ParisTech, Université Paris-Saclay
46 rue Barrault, 75634 Paris Cedex 13 - France

& Antoine Lobstein
Centre National de la Recherche Scientifique
Laboratoire de Recherche en Informatique, UMR 8623,
Université Paris-Sud, Université Paris-Saclay
Bâtiment 650 Ada Lovelace, 91405 Orsay Cedex - France

olivier.hudry@telecom-paristech.fr
antoine.lobstein@lri.fr

Abstract

The decision problems of the existence of a Hamiltonian cycle or of a Hamiltonian path in a given graph, and of the existence of a truth assignment satisfying a given Boolean formula $\mathcal{C}$, are well-known *NP*-complete problems. Here we study the problems of the *uniqueness* of a Hamiltonian cycle or path in an undirected, directed or oriented graph, and show that they have the same complexity, up to polynomials, as the problem U-SAT of the uniqueness of an assignment satisfying $\mathcal{C}$. As a consequence, these Hamiltonian problems are *NP*-hard and belong to the class *DP*, like U-SAT.

Key Words: Graph Theory, Hamiltonian Cycle, Hamiltonian Path, Travelling Salesman, Complexity Theory, *NP*-Hardness, Decision Problems, Polynomial Reduction, Uniqueness of Solution, Boolean Satisfiability Problems

14 Introduction

14.1 The Hamiltonian Cycle and Path Problems

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, $|V|$.

If $V = \{v_1, v_2, \ldots, v_n\}$, a *Hamiltonian path* $\mathcal{HP} =< v_{i_1} v_{i_2} \ldots v_{i_n} >$ is an ordering of all the vertices in V, such that $v_{i_j} v_{i_{j+1}} \in E$ for all j, $1 \leq j \leq n-1$. The vertices v_{i_1} and v_{i_n} are called the *ends* of $\mathcal{HP}$. A *Hamiltonian cycle* is an ordering $\mathcal{HC} =< v_{i_1} v_{i_2} \ldots v_{i_n} (v_{i_1}) >$ of all the vertices in V, such that $v_{i_n} v_{i_1} \in E$ and $v_{i_j} v_{i_{j+1}} \in E$ for all j, $1 \leq j \leq n-1$. Note that the same Hamiltonian cycle admits $2n$ representations, e.g., $< v_{i_2} v_{i_3} \ldots v_{i_n} v_{i_1} (v_{i_2}) >$ or $< v_{i_n} v_{i_{n-1}} \ldots v_{i_2} v_{i_1} (v_{i_n}) >$.

A *directed graph* $H = (X, A)$ is defined by its set X of vertices and its set A of directed edges, also called *arcs*, an arc being an ordered pair (x, y) of vertices; with this respect, (x, y) and (y, x) are two different arcs and may coexist. A directed graph is said to be *oriented* if it is antisymmetric, i.e., if we have, for any pair $\{x, y\}$ of vertices, at most one of the two arcs (x, y) or (y, x); if $(x, y) \in A$, we say that y is the *out-neighbour* of x, and x is the *in-neighbour* of y, and we define the *in-degree* and *out-degree* of a vertex accordingly. The notions of *directed Hamiltonian cycle* and of *directed Hamiltonian path* are extended to a directed graph by considering the arcs $(v_{i_n}, v_{i_1}) \in A$ and $(v_{i_j}, v_{i_{j+1}}) \in A$ in the above definitions. When there is no ambiguity, we shall often drop the words "directed" and "Hamiltonian".

The following six problems (stated as one) are well known, in graph theory as well as in complexity theory:

Problem HAMC / HAMP (Hamiltonian Cycle / Hamiltonian Path):
Instance: An undirected, directed or oriented graph.
Question: Does the graph admit a Hamiltonian cycle / Hamiltonian path?

As we shall see (Proposition 88), they have been known to be *NP*-complete for a long time. In this paper, we shall be interested in the following problems, and shall locate them in the complexity classes:

Problem U-HAMC[U] (Unique Hamiltonian Cycle in an Undirected graph):
Instance: An undirected graph $G = (V, E)$.
Question: Does G admit a *unique* Hamiltonian cycle?

Problem: U-HAMP[U] (Unique Hamiltonian Path in an Undirected graph):
Instance: An undirected graph $G = (V, E)$.
Question: Does G admit a *unique* Hamiltonian path?

Problem: U-HAMC[D] (Unique directed Hamiltonian Cycle in a Directed graph):
Instance: A directed graph $H = (X, A)$.
Question: Does H admit a *unique* directed Hamiltonian cycle?

Problem: U-HAMP[D] (Unique directed Hamiltonian Path in a Directed graph):
Instance: A directed graph $H = (X, A)$.
Question: Does H admit a *unique* directed Hamiltonian path?

Problem: U-HAMC[O] (Unique directed Hamiltonian Cycle in an Oriented graph):
Instance: An oriented graph $H = (X, A)$.
Question: Does H admit a *unique* directed Hamiltonian cycle?

Problem: U-HAMP[O] (Unique directed Hamiltonian Path in an Oriented graph):
Instance: An oriented graph $H = (X, A)$.
Question: Does H admit a *unique* directed Hamiltonian path?

We shall prove in Section 15 that these problems have the same complexity, up to polynomials, as the problem of the uniqueness of a truth assignment satisfying a Boolean formula (U-SAT). As a consequence, all are *NP*-hard and belong to the class *DP*. The closely related problem Unique Optimal Travelling Salesman has been investigated in [44], see Remark 94.

In a forthcoming work, we similarly reexamine some famous problems, from the viewpoint of uniqueness of solution: Boolean Satisfiability and Graph Colouring [34] *(Sections 1–4 in this Report)*, Vertex Cover and Dominating Set (as well as its generalization to domination within distance r) [35] *(Sections 5–8)*, and r-Identifying Code together with r-Locating-Dominating Code [36] *(Sections 9–13)*. We shall re-use here results from [34], and modify a construction from [35].

In the sequel, we shall need the following tools, which constitute classical definitions related to graph theory or to Boolean satisfiability. A *vertex cover* in an undirected graph G is a subset of vertices $V^* \subseteq V$ such that for every edge $e = uv \in E$, $V^* \cap \{u, v\} \neq \emptyset$. We denote by $\phi(G)$ the smallest cardinality of a vertex cover of G; any vertex cover V^* with $|V^*| = \phi(G)$ is said to be *optimal*.

Next we consider a set $\mathcal{X}$ of n *Boolean variables* x_i and a set $\mathcal{C}$ of m *clauses* ($\mathcal{C}$ is also called a *Boolean formula*); each clause c_j contains κ_j *literals*, a literal being a variable x_i or its complement $\overline{x}_i$. 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 are classical problems in complexity.

Problem VC (Vertex Cover with bounded size):
Instance: An undirected graph G and an integer k.
Question: Does G admit a vertex cover of size at most k?

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

The following problem is stated for any fixed integer $k \geq 2$.

Problem k-SAT (k-Satisfiability):
Instance: A set $\mathcal{X}$ of variables, a collection $\mathcal{C}$ of clauses over $\mathcal{X}$, each clause containing exactly k different literals.
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. We shall also consider the following variants of the above problems:

U-VC (Unique Vertex Cover with bounded size),
U-SAT (Unique Satisfiability),
U-k-SAT (Unique k-Satisfiability),
U-1-3-SAT (Unique One-in-Three Satisfiability).

They have the same instances as VC, SAT, k-SAT and 1-3-SAT respectively, but now the question is "Is there a *unique* vertex cover / truth assignment...?".

We shall give in Propositions 89–93 what we need to know about the complexities of these problems.

14.2 Some Classes of Complexity

We refer the reader to, e.g., [6], [21], [38] or [45] for more on this topic. A *decision problem* is of the type "Given an instance I and a property $\mathcal{PR}$ on I, is $\mathcal{PR}$ true for I?", and has only two solutions, "yes" or "no". The class P will denote the set of problems which can be solved by a *polynomial* (time) algorithm, and the class 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 transformation 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 shall write $\pi_1 \rightarrow_p \pi_2$. Cook [14] proved that there is one problem in NP, namely SAT, to which every other problem in NP can be polynomially reduced. Thus, in a sense, SAT is the "hardest" problem inside NP. Other problems share this property in NP and are called *NP-complete* problems; their class is denoted by *NP-C*. The way to show that a decision problem π is NP-complete is, once it is proved to be in NP, to choose some NP-complete problem π_1 and to polynomially reduce it to π. From a practical viewpoint, the NP-completeness of a problem π implies that we do not know any polynomial algorithm solving π, and that, under the assumption $P \neq NP$, which is widely believed to be true, no such algorithm exists: the time required can grow exponentially with the size of the instance (when the instance is a graph, its size is polynomially linked to its order; for a Boolean formula, the size is polynomially linked to, e.g., the number of variables plus the number of clauses).

The *complement* of a decision problem, "Given I and $\mathcal{PR}$, is $\mathcal{PR}$ true for I?", is "Given I and $\mathcal{PR}$, is $\mathcal{PR}$ false for I?". The class *co-NP* (respectively, *co-NP-C*) is the class of the problems which are the complement of a problem in NP (respectively, *NP-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. Thus, in this sense, π_2 is "at least as hard" as π_1. A problem π is *NP-hard* (respectively, *co-NP-hard*) if there is a Turing reduction from some NP-complete (respectively, co-NP-complete) problem to π [21, p. 113].

Remark 87 *Note that with these definitions, NP-hard and co-NP-hard coincide [21, p. 114].*

The notions of completeness and hardness can of course be extended to

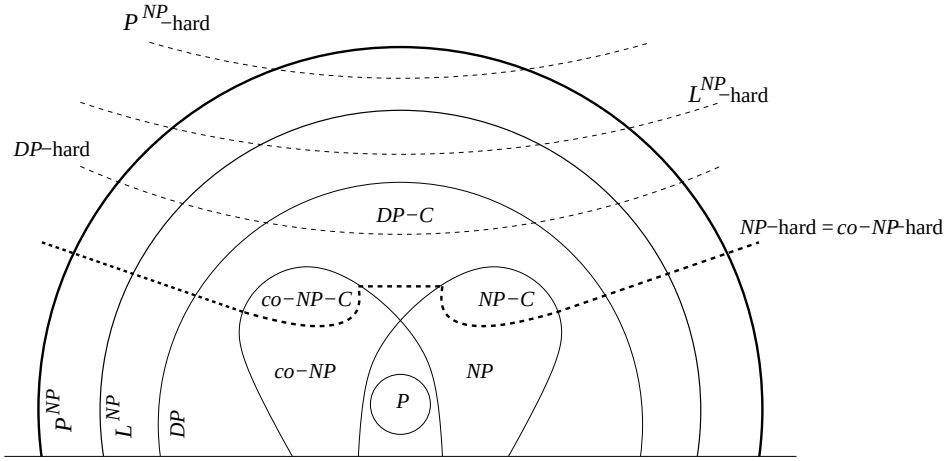

Figure 23: Some classes of complexity.

classes other than *NP* or co-*NP*. *NP*-hardness is defined differently in [15] and [27]: there, a problem π is *NP-hard* if there is a *polynomial* reduction from some *NP*-complete problem to π; this may lead to confusion (see Section 16).

We also introduce the classes P^{NP} (also known as Δ_2 in the hierarchy of classes) and L^{NP} (also denoted by $P^{NP[O(\log n)]}$ or Θ_2), which contain the decision problems which can be solved by applying, with a number of calls which is polynomial (respectively, logarithmic) with respect to the size of the instance, a subprogram able to solve an appropriate problem in *NP* (usually, an *NP*-complete problem); and the class *DP* [46] (or DIF^P [8] or BH_2 [38], [52], ...) as the class of languages (or problems) L such that there are two languages $L_1 \in NP$ and $L_2 \in$ co-*NP* satisfying $L = L_1 \cap L_2$. This class is not to be confused with $NP \cap$ co-*NP* (see the warning in, e.g., [45, p. 412]); actually, *DP* contains $NP \cup$ co-*NP* and is contained in L^{NP}. See Figure 23.

Membership to *P*, *NP*, co-*NP*, *DP*, L^{NP} or P^{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 the sense that we do not know whether or where the classes of complexity collapse.

We now consider some of the problems from Section 14.1.

Proposition 88 *[39], [21, pp. 56–60 and pp. 199-200] The decision prob-*

lems HAMC and HAMP, in an undirected, directed or oriented graph, are NP-complete. △

The problems VC, SAT and 3-SAT are also three of the basic and most well-known *NP*-complete problems [14], [21, p. 39, p. 46, p. 190 and p. 259]. More generally, k-SAT is *NP*-complete for $k \geq 3$ and polynomial for $k = 2$. The problem 1-3-SAT, which is obvioulsy in *NP*, is also *NP*-complete [48, Lemma 3.5], [21, p. 259], *and Remark 3 in this Report.*

The following results will be used in the sequel.

Proposition 89 *[34]* (= Theorem 10 in this Report) *For every integer $k \geq 3$, the decision problems U-SAT, U-k-SAT and U-1-3-SAT have equivalent complexity, up to polynomials.* △

Using the previous proposition and results from [8] and [45, p. 415], it is rather simple to obtain the following two results.

Proposition 90 *For every integer $k \geq 3$, the decision problems U-SAT, U-k-SAT and U-1-3-SAT are NP-hard (and co-NP-hard by Remark 87).* △

Proposition 91 *For every integer $k \geq 3$, the decision problems U-SAT, U-k-SAT and U-1-3-SAT belong to the class DP.* △

Remark 92 *It is not known whether these problems are DP-complete. In [45, p. 415], it is said that "U-SAT is not believed to be DP-complete".*

Proposition 93 *[35]* (see Theorems 30 and 34 in this Report) *The decision problems U-SAT and U-VC have equivalent complexity, up to polynomials. In particular, there exists a polynomial reduction from U-1-3-SAT to U-VC: U-1-3-SAT $\rightarrow_p$ U-VC.* △

After the following remark is made, we shall be ready to investigate the problems of uniqueness of Hamiltonian cycle or path.

Remark 94 *In [44], it is shown that the following problem is P^{NP}-complete (or Δ_2-complete).*

Problem *U-OTS (Unique Optimal Travelling Salesman):*
Instance: *A set of n vertices, a $n \times n$ symmetric matrix $[c_{ij}]$ of (nonnegative) integers giving the distance between any two vertices i and j.*
Question*: Is there a unique optimal tour, that is, a unique way of visiting every vertex exactly once and coming back, with the smallest distance sum?*

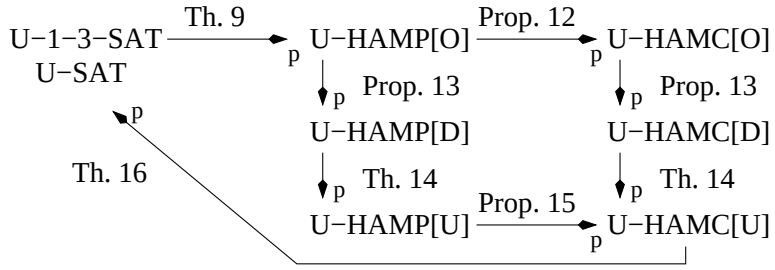

Figure 24: The chain of polynomial reductions.

At best, a polynomial reduction from any instance $G = (V, E)$ of U-HAMC[U] to U-OTS would show that U-HAMC[U] belongs to P^{NP}, but we have a better result in Theorem 103(b), with U-HAMC[U] belonging to DP; no useful information for our Hamiltonian problems can be induced from this result on U-OTS.

15 Locating the Problems of Uniqueness

We prove that our six Hamiltonian problems have the same complexity as any of the three problems U-SAT, U-k-SAT ($k \geq 3$) and U-1-3-SAT by proving the chain of polynomial reductions given by Figure 24.

Theorem 95 *There exists a polynomial reduction from U-1-3-SAT to U-HAMP[O]: U-1-3-SAT $\rightarrow_p$ U-HAMP[O].*

Proof. We describe a polynomial reduction from the problem U-1-3-SAT to U-HAMP[O], via U-VC: to do this, we use a polynomial reduction from U-1-3-SAT to U-VC, which is simpler than the one given in the proof of *Theorem 30 in this Report,* and mentioned in Proposition 93 of this paper; that proof had the advantage to cope with U-VC and U-OVC (Unique Optimal Vertex Cover) at the same time, but would give here a much more complicated proof for U-HAMP[O].

From an arbitrary instance of U-1-3-SAT with m clauses and n variables, we mimick the reduction from 3-SAT to VC in [39], [21, pp. 54–56], and we construct the instance $G_{VC} = (V_{VC}, E_{VC})$ of U-VC as follows (see Figure 25 for an example): we construct for each clause c_j a triangle $T_j = \{a_j, b_j, d_j\}$, and for each variable x_i a component $G_i = (V_i = \{x_i, \overline{x}_i\}, E_i = \{x_i\overline{x}_i\})$. Then we link the components G_i on the one hand, and the triangles T_j on

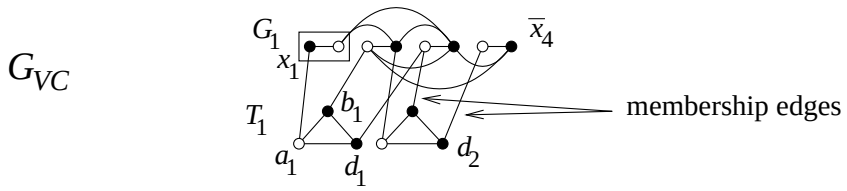

Figure 25: Illustration of the undirected graph constructed for the reduction from U-1-3-SAT to U-VC, with four variables and two clauses, $c_1 = \{x_1, x_2, x_3\}$, $c_2 = \{\overline{x}_2, x_3, x_4\}$. Here, $k = 8$, and the black vertices form the (not unique) vertex cover V^* of size eight corresponding to the (not unique) truth assignment $x_1 = \mathrm{T}$, $x_2 = \mathrm{F}$, $x_3 = \mathrm{F}$, $x_4 = \mathrm{F}$ 1-3-satisfying the clauses. As soon as we set $V^* \cap (V_1 \cup V_2 \cup V_3 \cup V_4) = \{x_1, \overline{x}_2, \overline{x}_3, \overline{x}_4\}$, the other vertices in V^* are forced.

the other hand, according to which literals appear in which clauses ("membership edges"). For each clause $c_j = \{\ell_1, \ell_2, \ell_3\}$, we also add the triangular set of edges $E'_j = \{\overline{\ell}_1\overline{\ell}_2, \overline{\ell}_1\overline{\ell}_3, \overline{\ell}_2\overline{\ell}_3\}$. Finally, we set $k = n + 2m$.

The order of G_{VC} is $3m + 2n$ and its number of edges is at most $n + 9m$ (the edge sets E'_j are not necessarily disjoint).

Note already that if V^* is a vertex cover, then each triangle T_j contains at least two vertices, each component G_i at least one vertex, and $|V^*| \geq 2m + n = k$; if $|V^*| = 2m + n$, then each triangle contains exactly two vertices, and each component G_i exactly one vertex. We can also observe that, because of the edge sets E'_j, at least two vertices among $\overline{\ell}_1, \overline{\ell}_2, \overline{\ell}_3$ belong to any vertex cover.

(a) 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 $\mathcal{C}$. Then, by taking, in each G_i, the vertex corresponding to the literal which is TRUE, and in every triangle T_j, the two vertices which are linked to the two false literals of c_j, we obtain a vertex cover V^* whose size is equal to k. Moreover, once we have put the n vertices corresponding to the true literals in the vertex cover V^* in construction, we have *no choice* for the completion of V^* with $k - n = 2m$ vertices: when we take two vertices in T_j, we *must* take the two vertices which cover the membership edges linked to the two false literals (in the example of Figure 25, the vertices b_1, d_1 and b_2, d_2). So, if another vertex cover V^+ of size k exists, it must have a different distribution of its vertices over the components G_i, still with exactly one vertex in each G_i; this in turn defines a valid truth assignment, by setting $x_i = \mathrm{T}$ if $x_i \in V^+$,

$x_i = \mathrm{F}$ if $\overline{x}_i \in V^+$. Now this assignment 1-3-satisfies $\mathcal{C}$, thanks in particular to our observation on the covering of the edges in E'_j. So we have two truth assignments 1-3-satisfying $\mathcal{C}$, contradicting the YES answer to U-1-3-SAT; therefore, V^* is the only vertex cover of size k.

(b) 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, this would lead, using the same argument as in the previous paragraph, to at least two vertex covers of size k, and a NO answer to U-VC. So we are left with the case when the set of clauses $\mathcal{C}$ cannot be 1-3-satisfied. But again, we have already seen that this would imply that no vertex cover of size (at most) k exists, since such a hypothetical vertex cover V^+ would imply the existence of a suitable assignment.

We are now ready to construct an instance of U-HAMP[O]. In the sequel, we shall say "path" for "directed Hamiltonian path".

We look deeper into the proof of the *NP*-completeness of the problem Hamiltonian Cycle (see [21, pp. 56–60]), which uses a polynomial reduction from VC to HAMC[U] that, due to the so-called "selector vertices", cannot cope with the problem of uniqueness; step by step, we construct an oriented graph $H = (X, A)$ for which we will prove that:

(i) if there is a YES answer for the instance of U-1-3-SAT (which implies that there is a unique vertex cover V^* in G_{VC}, with cardinality at most k), then there is a unique path in H;

(ii) if there are at least two assignments 1-3-satisfying all the clauses (i.e., there are at least two vertex covers in G_{VC}, with cardinality at most k), then there are at least two paths in H;

(iii) if there is no assignment 1-3-satisfying the clauses (and no vertex cover in G_{VC} with cardinality at most k), then there is no path in H.

Step 1. For each edge $e = uv \in E_{VC}$, we build one component $H_e = (X_e, A_e)$ with 12 vertices and 14 arcs: $X_e = \{(u,e,i), (v,e,i) : 1 \leq i \leq 6\}$, $A_e = \{((u,e,i),(u,e,i+1)), ((v,e,i),(v,e,i+1)) : 1 \leq i \leq 5\} \cup \{((v,e,3),(u,e,1)), ((u,e,3),(v,e,1))\} \cup \{((v,e,6),(u,e,4)), ((u,e,6),(v,e,4))\}$; see Figure 26, which is the oriented copy of Figure 3.4 in [21, p. 57].

In the completed construction, the only vertices from this component that will be involved in any additional arcs are the vertices $(u,e,1)$, $(u,e,6)$, $(v,e,1)$, and $(v,e,6)$. This, together with the fact that there will be two particular vertices, α_1 and δ, which will necessarily be the ends of any path, will imply that any path in the final graph H will have to meet the vertices in X_e in exactly one of the three configurations shown in Figure 27, which

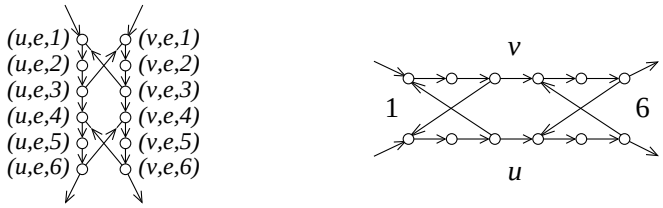

Figure 26: Two possible representations of the same component H_e for the edge $e = uv \in E_{VC}$ (Step 1).

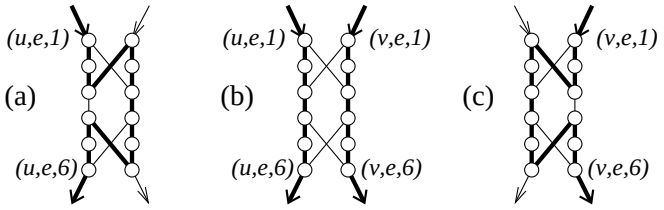

Figure 27: The three ways of going through the component H_e (Step 1). The arrows inside H_e are not represented.

is the oriented copy of Figure 3.5, in [21, p. 58]. Thus, when the path meets the component H_e at $(u, e, 1)$, it will have to leave at $(u, e, 6)$ and go through either (a) all 12 vertices in the component, in which case we shall say that the component is *completely visited from the u-side*, or (b) only the 6 vertices (u, e, i), $1 \leq i \leq 6$, in which case we shall say that the component is visited *in parallel* and needs two visits, i.e., another section of the path will re-visit the component, meeting the 6 vertices (v, e, i), $1 \leq i \leq 6$.

Step 2. We create n vertices α_i, $1 \leq i \leq n$, and $2n$ arcs $(\alpha_i, (x_i, x_i\overline{x}_i, 1))$, $(\alpha_i, (\overline{x}_i, x_i\overline{x}_i, 1))$, that is, we link α_i to the "first" vertices of the component H_e whenever $e = x_i\overline{x}_i$. The vertices α_i can be seen as literal selectors that will choose between x_i and $\overline{x}_i$. The vertex α_1 will have no other neighbours; this means in particular that it will have no in-neighbours, thus it will necessarily be the starting vertex of any Hamiltonian path, if such a path exists.

We choose an arbitrary order on the $3m$ vertices of the triangles T_j in the graph G_{VC}, say $\mathcal{O}_T =< a_1, b_1, d_1, a_2, \ldots, d_m >$ and an arbitrary order on the literals $x_i, \overline{x}_i$, say $\mathcal{O}_\ell =< x_1, x_2, \ldots, x_n, \overline{x}_1, \ldots, \overline{x}_n >$. For each literal ℓ_i equal to x_i or $\overline{x}_i$, we do the following (see Figure 28 for an example):

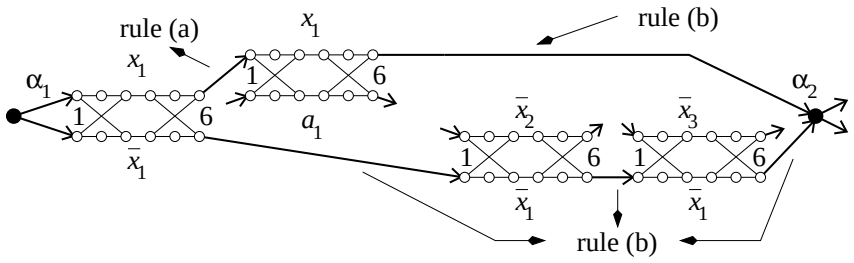

Figure 28: The example of the literals x_1 and $\overline{x}_1$ from Figure 25 (Step 2); here, Rule (a) applies with $q(x_1) = 1$, $q(\overline{x}_1) = 0$, Rule (b) with $s(\overline{x}_1) = 2$. The arrows inside H_e are not represented.

Rule (a): If ℓ_i appears $q = q(\ell_i) \geq 0$ times in the clauses and is linked in G_{VC} to $t_1, \ldots, t_q$ where the t's belong to the triangles T_j and follow the order $\mathcal{O}_T$, then we create the arcs $((\ell_i, \ell_i\overline{\ell}_i, 6), (\ell_i, \ell_i t_1, 1))$, $((\ell_i, \ell_i t_1, 6), (\ell_i, \ell_i t_2, 1))$, $\ldots$, $((\ell_i, \ell_i t_{q-1}, 6), (\ell_i, \ell_i t_q, 1))$.

Rule (b): We consider the triangular sets of edges E'_j described in the construction of G_{VC}.

• If ℓ_i does not belong to any such edge, we create the arc $((\ell_i, \ell_i t_q, 6), \alpha_{i+1})$ —or $((\ell_i, \ell_i\overline{\ell}_i, 6), \alpha_{i+1})$ if ℓ_i does not apppear in any clause— unless $i = n$, in which case we create $((\ell_i, \ell_i t_q, 6), \beta_1)$ or $((\ell_i, \ell_i\overline{\ell}_i, 6), \beta_1)$, where β_1 is a new vertex that will be spoken of at the beginning of Step 3.

• If ℓ_i belongs to $s = s(\ell_i) > 0$ edges from E'_j, which link ℓ_i to s literals $\ell_{i_1}, \ldots, \ell_{i_s}$ that follow the order $\mathcal{O}_\ell$, then we build the arc $((\ell_i, \ell_i t_q, 6), (\ell_i, \ell_i\ell_{i_1}, 1))$ —or the arc $((\ell_i, \ell_i\overline{\ell}_i, 6), (\ell_i, \ell_i\ell_{i_1}, 1))$ if $q = 0$; next, the arcs $((\ell_i, \ell_i\ell_{i_1}, 6), (\ell_i, \ell_i\ell_{i_2}, 1))$, $\ldots$, $((\ell_i, \ell_i\ell_{i_{s-1}}, 6), (\ell_i, \ell_i\ell_{i_s}, 1))$ and $((\ell_i, \ell_i\ell_{i_s}, 6), \alpha_{i+1})$, unless $i = n$, in which case we create $((\ell_i, \ell_i\ell_{i_s}, 6), \beta_1)$.

Remark 96 *In the example of Figure 28, one can see that if a path takes, e.g., the arc* $(\alpha_1, (x_1, x_1\overline{x}_1, 1))$*, then it visits the vertices* $(x_1, x_1\overline{x}_1, 6)$*,* $(x_1, x_1a_1, 1)$*,* $(x_1, x_1a_1, 6)$*, and* α_2*. If on the other hand, we use the arc* $(\alpha_1, (\overline{x}_1, x_1\overline{x}_1, 1))$*, we also go to* α_2*. The same is true between* α_2 *and* α_3*,* $\ldots$*, between* α_{n-1} *and* α_n*, between* α_n *and* β_1*.*

We can see that so far, α_1 has (out-)degree 2, α_2, $\ldots$, α_n have degree 4 (in- and out-degrees equal to 2), and β_1 has (in-)degree 2.

Step 3. We consider the m clauses and the m corresponding triangles T_j.

We create $2m$ vertices β_j, β'_j, $1 \leq j \leq m$. As we have seen in the previous step, β_1 has already two in-neighbours, which can be $(\ell_n, \ell_n t_q, 6)$,

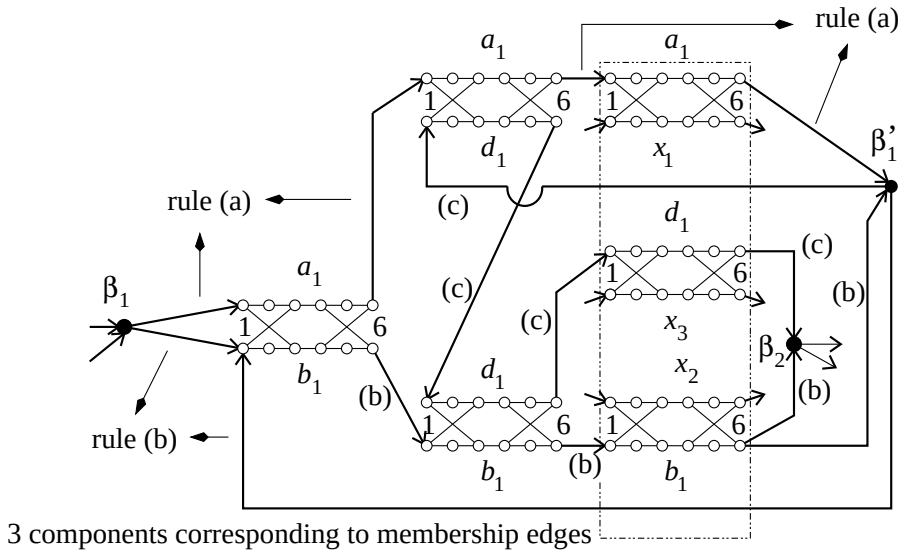

Figure 29: The treatment of the triangle T_1 from Figure 25 (Step 3). The arrows inside H_e are not represented.

or $(\ell_n, \ell_n\overline{\ell}_n, 6)$, or $(\ell_n, \ell_n\ell_{n_s}, 6)$. We also create one more vertex δ, which will have only in-neighbours, so that α_1 and δ will necessarily be the ends of any directed Hamiltonian path, if such a path exists.

Now for the triangle $T_j = \{a_j, b_j, d_j\}$, $1 \leq j \leq m$, associated to the clause $c_j = \{\ell_{j_1}, \ell_{j_2}, \ell_{j_3}\}$ in the graph G_{VC}, we consider the six corresponding components $H_{a_jb_j}$, $H_{a_jd_j}$, $H_{b_jd_j}$, $H_{a_j\ell_{j_1}}$, $H_{b_j\ell_{j_2}}$ and $H_{d_j\ell_{j_3}}$. The vertices β_j and β_j' can be seen as triangle selectors, intended to choose two vertices among three. With this in mind, we create the following arcs (see Figure 29), for $j \in \{1, 2, \ldots, m\}$:

Rule (a): $(\beta_j, (a_j, a_jb_j, 1))$, $((a_j, a_jb_j, 6), (a_j, a_jd_j, 1))$, $((a_j, a_jd_j, 6), (a_j, a_j\ell_{j_1}, 1))$, $((a_j, a_j\ell_{j_1}, 6), \beta_j')$.

Rule (b): $(\beta_j, (b_j, a_jb_j, 1))$, $(\beta_j', (b_j, a_jb_j, 1))$, $((b_j, a_jb_j, 6), (b_j, b_jd_j, 1))$, $((b_j, b_jd_j, 6), (b_j, b_j\ell_{j_2}, 1))$, $((b_j, b_j\ell_{j_2}, 6), \beta_j')$, plus the arc $((b_j, b_j\ell_{j_2}, 6), \beta_{j+1})$, unless $j = m$, in which case it is $((b_j, b_j\ell_{j_2}, 6), \delta)$.

Rule (c): $(\beta_j', (d_j, a_jd_j, 1))$, $((d_j, a_jd_j, 6), (d_j, b_jd_j, 1))$, $((d_j, b_jd_j, 6), (d_j, d_j\ell_{j_3}, 1))$, $((d_j, d_j\ell_{j_3}, 6), \beta_{j+1})$, unless $j = m$, in which case it is $((d_j, d_j\ell_{j_3}, 6), \delta)$.

Remark 97 *On the example of Figure 29, there are three ways for going from β_1 to β_2 through the components $H_{a_1b_1}$, $H_{a_1d_1}$ and $H_{d_1b_1}$.*

• *If a path starts by taking the arc* $(\beta_1, (a_1, a_1b_1, 1))$*, then there are two possibilities, according to how we visit* $H_{a_1b_1}$*:*

◦ *The first possibility corresponds to taking* a_1 *and* d_1*, not* b_1*, in a vertex cover: the path completely visits the component* $H_{a_1b_1}$ *from the* a_1*-side, then the component* $H_{a_1d_1}$ *in parallel, then the component* $H_{a_1x_1}$ *in a so far unspecified way, then* β'_1*.*

Next, it takes the arc $(\beta'_1, (d_1, d_1a_1, 1))$*, re-visits* $H_{d_1a_1}$ *in parallel, completely visits* $H_{d_1b_1}$ *from the* d_1*-side, then* $H_{d_1x_3}$*, and ends this path section at* β_2*. One can see that the three components corresponding to edges incident to* b_1 *must all be completely visited from the side opposite* b_1*, including the* x_2*-side.*

◦ *The second possibility corresponds to taking* a_1 *and* b_1*, not* d_1*, in a vertex cover: the path follows the arc* $(\beta_1, (a_1, a_1b_1, 1))$*, visits* $H_{a_1b_1}$ *in parallel, visits completely* $H_{a_1d_1}$ *from the* a_1*-side, then* $H_{a_1x_1}$*, and* β'_1*.*

Next, it takes the arc $(\beta'_1, (b_1, b_1a_1, 1))$*, goes through* $H_{b_1a_1}$ *in parallel, goes completely through* $H_{b_1d_1}$ *from the* b_1*-side, then* $H_{b_1x_2}$*, and ends this path section at* β_2*. The component* $H_{d_1x_3}$ *is not yet visited.*

• *Alternatively, a path can start by taking the arc* $(\beta_1, (b_1, a_1b_1, 1))$*; this corresponds to taking* b_1 *and* d_1*, not* a_1*, in a vertex cover and constitutes the third way for going from* β_1 *to* β_2*. The path then completely visits* $H_{b_1a_1}$ *from the* b_1*-side,* $H_{b_1d_1}$ *in parallel,* $H_{b_1x_2}$*, and* β'_1*.*

Next, it completely visits $H_{d_1a_1}$ *from the* d_1*-side,* $H_{d_1b_1}$ *in parallel,* $H_{d_1x_3}$*, and this path section ends at* β_2*. The component* $H_{a_1x_1}$ *is not yet visited.*

It is easy to see that these are the only *three ways for going from* β_1 *to* β_2 *through the components* $H_{a_1b_1}$*,* $H_{a_1d_1}$ *and* $H_{d_1b_1}$*, not taking into account the ways of going through the components* $H_{a_1x_1}$*,* $H_{b_1x_2}$ *and* $H_{d_1x_3}$ *(this issue will be treated later on, in the general case): indeed, the only possibility left would be to follow the arc* $(\beta_1, (b_1, a_1b_1, 1))$ *and visit* $H_{b_1a_1}$ *in parallel, but then the* a_1*-side of* $H_{b_1a_1}$ *cannot be reached.*

The same will be true for the components $H_{a_jb_j}$*,* $H_{a_jd_j}$ *and* $H_{d_jb_j}$ *and the corresponding triangles* T_j*,* $1 \leq j \leq m$*, between* β_j *and* β_{j+1} *(or between* β_m *and* δ*).*

The description of the oriented graph H is complete. Now β_1 has increased its degree to 4, and β_2, ..., β_m and β'_1, ..., β'_m have degree 4. *Actually, all the selectors but* α_1 *have in-degree* 2 *and out-degree* 2 *in* H*.* These n selectors α_i, $1 \leq i \leq n$, and $2m$ selectors β_j, β'_j, $1 \leq j \leq m$, translate the choices we have to make when constructing a vertex cover with size $2m + n$: we have one choice among the n variables (take x_i or $\overline{x}_i$); as for the m triangles T_j

associated to the clauses, Remark 97 has shown how the selectors β_j, β'_j, $1 \leq j \leq m$, can be used to choose two vertices among three. The *number* of selectors is one reason why there is no directed Hamiltonian path in H when the vertex covers in G_{VC} have size at least $2m + n + 1$.

The order of H is $12|E_{VC}| + n + 2m + 1$, which is at most $12(n + 9m) + n + 2m + 1$, so that the transformation is polynomial indeed.

We are now going to prove our claims about the existence or non-existence, uniqueness or non-uniqueness, of a directed Hamiltonian path in H.

Assume first that there is an assignment satisfying the instance of U-1-3-SAT, and therefore that there is a vertex cover V^* in G_{VC} with size $2m+n$. We construct a path in H in a straightforward way: every component H_{uv} ($uv \in E_{VC}$) with $\{u, v\} \subset V^*$ is visited in parallel, whereas H_{uv} is completely visited from the u-side whenever $u \in V^*$, $v \notin V^*$. Let us have a closer look on how this works:

We start at α_1, and visit completely the component $H_{x_1\overline{x}_1}$ from the x_1-side if $x_1 = \mathrm{T}$, from the $\overline{x}_1$-side if $x_1 = \mathrm{F}$ (or, equivalently, if $x_1 \in V^*$ or $\overline{x}_1 \in V^*$, respectively). If, say, $x_1 = \mathrm{F}$, we then completely go through all the components corresponding to triangles T_j and involving $\overline{x}_1$, all from the $\overline{x}_1$-side; note that all the components just completely visited involve $\overline{x}_1$ and a vertex not in V^*, by the very construction of the vertex cover V^*, which is possible because it stems from an assignment 1-3-satisfying all the clauses. Then we go through the components constructed from the edge sets E'_j and involving $\overline{x}_1$; those involving a second vertex in V^* (i.e., a true literal) are visited in parallel, whereas those involving a vertex not in V^* are completely visited from the $\overline{x}_1$-side; then the path arrives at α_2. The components involving x_1, apart from $H_{x_1\overline{x}_1}$, remain completely unvisited for the time being, and the components that have been visited in parallel will have to be re-visited.

We act similarly between α_2 and α_3, ..., α_n and β_1; cf. Remark 96. When doing this, we re-visit all the components that had been visited only in parallel, and completely visit the components involving a literal not in V^* and corresponding to edges in E'_j. The only components not visited yet between α_1 and β_1 are those corresponding to edges between a false literal (not in V^*) and its neighbours in the triangles T_j.

Next, starting from β_1, we use Remark 97 according to the three possible cases: (a) $\{a_1, d_1\} \subset V^*$, $b_1 \notin V^*$, (b) $\{a_1, b_1\} \subset V^*$, $d_1 \notin V^*$, (c) $\{b_1, d_1\} \subset V^*$, $a_1 \notin V^*$. We give in detail only the third case, for the clause $c_1 = \{\ell_1, \ell_2, \ell_3\}$: we use the arc $(\beta_1, (b_1, a_1b_1, 1))$ and completely visit the component $H_{a_1b_1}$ from the b_1-side, then the component $H_{b_1d_1}$ in

parallel, then the complete component $H_{b_1\ell_2}$ from the b_1-side (because if $b_1 \in V^*$, then $\ell_2 \notin V^*$ and this component had not yet been visited) and end at β'_1. Next, we take the arc $(\beta'_1, (d_1, d_1a_1, 1))$, we completely visit $H_{d_1a_1}$ from the d_1-side, re-visit $H_{d_1b_1}$ in parallel, completely visit $H_{d_1\ell_3}$ from the d_1-side, and this path section ends at β_2. Note that (a) the three components involving a_1 between β_1 and β_2 have been completely visited, from the b_1-, d_1- or ℓ_1-sides (because $a_1 \notin V^*$ implies that $\ell_1 \in V^*$); (b) any so far unvisited component involving a false literal (here, these are ℓ_2 and ℓ_3) and one of the vertices of the triangle T_1 (here b_1 and d_1) has now been completely visited from the triangle sides (here from the b_1- and d_1-sides).

We act similarly between β_2 and β_3, ..., β_m and δ; cf. the end of Remark 97. The ultimate section takes us between β_m and δ, the final vertex, and we have indeed built a directed Hamiltonian path, from α_1 to δ, in the oriented graph H.

Obviously, two different assignments 1-3-satisfying all the clauses lead, following the above process, to two different paths in H. We still want to prove that 1) if no assignment 1-3-satisfying all the clauses exists, then no path exists, and 2) a unique assignment 1-3-satisfying all the clauses leads to a unique path.

1) We assume that there is a directed Hamiltonian path $\mathcal{HP}$ in H, and exhibit an assignment 1-3-satisfying all the clauses.

Let us consider the vertex α_1; its two out-neighbours in H are $(x_1, x_1\overline{x}_1, 1)$ and $(\overline{x}_1, x_1\overline{x}_1, 1)$. So exactly one of the arcs $(\alpha_1, (x_1, x_1\overline{x}_1, 1))$, $(\alpha_1, (\overline{x}_1, x_1\overline{x}_1, 1))$ is part of $\mathcal{HP}$. The same is true for α_i, $1 < i \leq n$. As a consequence, we can define a valid assignment of the variables x_i, $1 \leq i \leq n$, by setting $x_i = \mathrm{T}$ if and only if the arc $(\alpha_i, (x_i, x_i\overline{x}_i, 1))$ belongs to $\mathcal{HP}$.

Next, we address the vertices β_j, $1 \leq j \leq m$. The construction in Steps 2 and 3 is such that each vertex β_j, $1 \leq j \leq m$, has two out-neighbours, $(a_j, a_jb_j, 1)$ and $(b_j, a_jb_j, 1)$.

This implies that the assignment defined above is such that there is at least one true literal in each clause. Indeed, if we assume that the clause $c_j = \{\ell_{j_1}, \ell_{j_2}, \ell_{j_3}\}$ does not contain any true literal, then the component $H_{a_j\ell_{j_1}}$ is completely visited by $\mathcal{HP}$ from the a_j-side, because $\ell_{j_1} = \mathrm{F}$ implies that the arc $(\alpha_j, (\ell_{j_1}, \ell_{j_1}\overline{\ell}_{j_1}, 1))$ is not part of $\mathcal{HP}$ and does not give access to the ℓ_{j_1}-side. Similarly, the components $H_{b_j\ell_{j_2}}$ and $H_{d_j\ell_{j_3}}$ are completely visited by $\mathcal{HP}$ from the b_j- and d_j-sides, respectively. This in turn implies that in $\mathcal{HP}$ we have the arcs $((a_j, a_j\ell_{j_1}, 6), \beta'_j)$, $(\beta_j, (a_j, a_jb_j, 1))$, $((d_j, d_j\ell_{j_3}, 6), \beta_{j+1})$ and $(\beta'_j, (d_j, d_ja_j, 1))$ —replace β_{j+1} by δ if $j = m$. Now how does $\mathcal{HP}$ go through $(b_j, b_j\ell_{j_2}, 6)$? It cannot be with the help of the ℓ_{j_2}-

side of $H_{b_j\ell_{j_2}}$, so there are only two possibilities left: but if it is with the arc $((b_j, b_j\ell_{j_2}, 6), \beta'_j)$, then β'_j has three neighbours in $\mathcal{HP}$, which is impossible; and if it is with the arc $((b_j, b_j\ell_{j_2}, 6), \beta_{j+1})$, then in $\mathcal{HP}$, the vertex β_{j+1} has two in-neighbours, which is impossible —including when $j = m$ and β_{j+1} is replaced by δ. From this we can conclude that the clause $c_j = \{\ell_{j_1}, \ell_{j_2}, \ell_{j_3}\}$ contains at least one true literal.

Assume next that one clause has at least two true literals: without loss of generality, $c_j = \{\ell_{j_1}, \ell_{j_2}, \ell_{j_3}\}$ is such that $\ell_{j_1} = \ell_{j_2} = \mathrm{T}$. Then $\mathcal{HP}$ has no access to the $\overline{\ell}_{j_1}$- and $\overline{\ell}_{j_2}$- sides of the components involving $\overline{\ell}_{j_1}$ or $\overline{\ell}_{j_2}$, but, since there is the edge $\overline{\ell}_{j_1}\overline{\ell}_{j_2}$ in G_{VC}, this means that $\mathcal{HP}$ has no way of visiting the component $H_{\overline{\ell}_{j_1}\overline{\ell}_{j_2}}$. Therefore, we have just established that the assignment derived from the path $\mathcal{HP}$ 1-3-satisfies all the clauses. This, together with the fact that two assignments 1-3-satisfying the clauses lead to two paths, shows that a NO answer to the instance of U-1-3-SAT implies a NO answer for the constructed instance H of U-HAMP[O].

2) We want to show that a unique assignment $\mathcal{A}$ 1-3-satisfying all the clauses leads to a unique path in H. This assignment leads to a unique vertex cover V^*, of size $n + 2m$, in G_{VC}, and to a path in H, as already seen. Now assume that we have a second path, so that these two paths, which we call $\mathcal{HP}_1$ and $\mathcal{HP}_2$, both lead, with the above description in 1), to the same $\mathcal{A}$ and the same V^*.

The two paths must behave in the same way over the components $H_{x_i\overline{x}_i}$, $1 \leq i \leq n$: otherwise, from them we could define two different valid assignments, which would both, as seen previously, 1-3-satisfy the clauses.

Next, consider the clause $c_j = \{\ell_{j_1}, \ell_{j_2}, \ell_{j_3}\}$ and assume without loss of generality that $\mathcal{A}(\ell_{j_1}) = \mathrm{T}$, $\mathcal{A}(\ell_{j_2}) = \mathcal{A}(\ell_{j_3}) = \mathrm{F}$; this implies, for both $\mathcal{HP}_1$ and $\mathcal{HP}_2$, that the components $H_{\ell_{j_1}\overline{\ell}_{j_1}}$, $H_{\ell_{j_2}\overline{\ell}_{j_2}}$ and $H_{\ell_{j_3}\overline{\ell}_{j_3}}$ are completely visited from the ℓ_{j_1}-, $\overline{\ell}_{j_2}$- and $\overline{\ell}_{j_3}$-sides, respectively, so that both paths have no access to the ℓ_{j_2}- nor ℓ_{j_3}-sides. As a consequence, between β_j and β_{j+1} (or β_m and δ), the components $H_{b_j\ell_{j_2}}$ and $H_{d_j\ell_{j_3}}$ are completely visited from the b_j- and d_j-sides, respectively. Then necessarily the following arcs belong to $\mathcal{HP}_1$ and $\mathcal{HP}_2$, going along the d_j-side:
$((d_j, d_j\ell_{j_3}, 6), \beta_{j+1})$ —or $((d_j, d_j\ell_{j_3}, 6), \delta)$—, $((d_j, d_jb_j, 6), (d_j, d_j\ell_{j_3}, 1))$, $((d_j, d_ja_j, 6), (d_j, d_jb_j, 1))$, $(\beta'_j, (d_j, d_ja_j, 1))$;

and going along the b_j-side:
$((b_j, b_j\ell_{j_2}, 6), \beta'_j)$ (because β_{j+1} —or δ— cannot have two in-neighbours), and $((b_j, b_jd_j, 6), (b_j, b_j\ell_{j_2}, 1))$, $((b_j, b_ja_j, 6), (b_j, b_jd_j, 1))$.

The component $H_{b_jd_j}$ must be visited in parallel, and it is $(\beta_j, (b_j, b_ja_j, 1))$ that belongs to the two paths.

We can see that all the components containing a_j, in particular $H_{a_j\ell_{j_1}}$, must be completely visited from the sides opposite a_j. So far, we have proved that the two paths $\mathcal{HP}_1$ and $\mathcal{HP}_2$ behave identically between β_j and β_{j+1} (or β_m and δ), including on the components corresponding to membership edges (between literals and triangles).

Consider now what happens between α_i and α_{i+1} (or α_n and β_1). Assume without loss of generality that, say, $\mathcal{A}(x_i) = \mathrm{T}$, so that $(\alpha_i, (x_i, x_i\overline{x}_i, 1))$ is part of the two paths. Consider the components involving x_i in H: there are first those involving vertices of type a, b or d, which translate the membership of x_i to a certain number of clauses, and which we called $t_1, \ldots, t_q$ in Step 2(a); we have already seen in the previous paragraph that these components must be completely visited from the x_i-side.

Then we consider the components created from the edges in E'_j, cf. Step 2(b); here, some edges in G_{VC} can have both ends in V^*, but, using similar arguments as before, we can see that the two paths will visit all these components in the same way: consider the clause $c_j = \{\ell_{j_1}, \ell_{j_2}, \ell_{j_3}\}$ and the corresponding set E'_j, and assume without loss of generality that $x_i = \ell_{j_1}$, so that $\mathcal{A}(\ell_{j_1}) = \mathrm{T}$, which implies that $\mathcal{A}(\ell_{j_2}) = \mathcal{A}(\ell_{j_3}) = \mathrm{F}$; then $H_{\bar{\ell}_{j_1}\bar{\ell}_{j_2}}$ and $H_{\bar{\ell}_{j_1}\bar{\ell}_{j_3}}$ must be completely visited from the $\overline{\ell}_{j_2}$- and $\overline{\ell}_{j_3}$-sides, respectively, and $H_{\bar{\ell}_{j_2}\bar{\ell}_{j_3}}$ in parallel, i.e., the two paths have no choice but to behave identically on all three components. As for the components with $\overline{x}_i$, they must be completely visited from the side which is not the side of $\overline{x}_i$.

So we have just proved that the two paths are identical between α_1 and α_2, …, α_n and β_1.

Therefore, the two paths (between α_1 and δ) are one and the same. △

Proposition 98 *There exists a polynomial reduction from U-HAMP[O] to U-HAMC[O]: U-HAMP[O] $\rightarrow_p$ U-HAMC[O].*

Proof. We start from an oriented graph $H = (X, A)$ which is an instance of U-HAMP[O] and build a graph which is an instance of U-HAMC[O] by adding two extra vertices y, z, together with the arc (y, z) and all the arcs (x, y) and (z, x), $x \in X$. This transformation is polynomial and clearly preserves the number of solutions, in particular the uniqueness. △

Proposition 99 *There is a polynomial reduction from U-HAMP[O] to U-HAMP[D] and from U-HAMC[O] to U-HAMC[D]:*
U-HAMP[O] $\rightarrow_p$ U-HAMP[D] and U-HAMC[O] $\rightarrow_p$ U-HAMC[D].

Proof. It suffices to consider the identity as the polynomial reduction. △

Theorem 100 *There is a polynomial reduction from U-HAMP[D] to U-HAMP[U] and from U-HAMC[D] to U-HAMC[U]:*
U-HAMP[D] $\rightarrow_p$ *U-HAMP[U] and U-HAMC[D]* $\rightarrow_p$ *U-HAMC[U].*

Proof. The method is borrowed from [39].

Consider any instance of U-HAMP[D] or U-HAMC[D], i.e., a directed graph $H = (X, A)$ on n vertices. We build the undirected graph $G = (V, E)$, the instance of U-HAMP[U] or U-HAMC[U], as follows: every vertex $x \in X$ is triplicated into three vertices $x^- \in V$ (a *minus-type vertex*), $x^* \in V$ (a *star-type vertex*) and $x^+ \in V$, linked by the edges $x^-x^* \in E$ and $x^*x^+ \in E$; for every arc $(x, y) \in A$ is created the edge x^+y^- in E. The graph G thus constructed has order $3n$.

We claim that there is a unique Hamiltonian cycle (respectively, path) in G if and only if there is a unique directed Hamiltonian cycle (respectively, path) in H.

(1) Assume first that H admits a directed Hamiltonian cycle $< x_1x_2 \dots x_n(x_1) >$. Then

$$< x_1^- x_1^* x_1^+ x_2^- x_2^* x_2^+ \dots x_{n-1}^+ x_n^- x_n^* x_n^+ (x_1^-) >$$

is a Hamiltonian cycle in G. Moreover, two different directed Hamiltonian cycles in H provide two different Hamiltonian cycles in G.

Conversely, assume that G admits a Hamiltonian cycle $\mathcal{HC}$. This cycle must go through all the star-type vertices x^*, so it necessarily goes through all the edges x^-x^* and x^*x^+. Without loss of generality, $\mathcal{HC}$ reads:

$$\mathcal{HC} =< x_1^- x_1^* x_1^+ x_2^- x_2^* x_2^+ \dots x_{n-1}^+ x_n^- x_n^* x_n^+ (x_1^-) > ; \tag{6}$$

indeed, we may assume that we "start" with the edge $x_1^-x_1^*$, then $x_1^*x_1^+$; now, because the edges which have no star-type vertex as one of their extremities are necessarily of the type x^+y^-, the other neighbour of x_1^+ is a minus-type vertex, say x_2^-; step by step, we see that $\mathcal{HC}$ has necessarily the previous form (6). Now we claim that $< x_1x_2 \dots x_{n-1}x_n(x_1) >$ is a directed Hamiltonian cycle in H.

Indeed, for every $i \in \{1, \dots, n-1\}$, the edge $x_i^+x_{i+1}^-$ in G implies the existence of the arc (x_i, x_{i+1}) in H; the same is true for the arc (x_n, x_1) in H, thanks to the edge $x_n^+x_1^-$ in G. Furthermore, observe that two different Hamiltonian cycles in G provide two different directed Hamiltonian cycles in H.

So, G admits a unique Hamiltonian cycle if and only if H admits a unique directed Hamiltonian cycle.

(2) Exactly the same argument works with paths, apart from the fact that we need not consider the arc (x_n, x_1) in H, nor the edge $x_n^+ x_1^-$ in G. $\triangle$

Proposition 101 *There exists a polynomial reduction from U-HAMP[U] to U-HAMC[U]: U-HAMP[U] $\rightarrow_p$ U-HAMC[U].*

Proof. We start from an undirected graph $G = (V, E)$ which is an instance of U-HAMP[U] and build a graph which is an instance of U-HAMC[U] by adding the extra vertex y, together with all the edges xy, $x \in V$. This transformation is polynomial and clearly preserves the number of solutions, in particular the uniqueness. $\triangle$

Theorem 102 *There exists a polynomial reduction from U-HAMC[U] to U-SAT: U-HAMC[U] $\rightarrow_p$ U-SAT.*

Proof. We start from an instance of U-HAMC[U], an undirected graph $G = (V, E)$ with $V = \{x^1, \ldots, x^{|V|}\}$; we assume that $|V| \geq 3$. We create the set of variables $\mathcal{X} = \{x_j^i : 1 \leq j \leq |V|, 1 \leq i \leq |V|\}$ and the following clauses:

(a_1) for $1 \leq i \leq |V|$, clauses of size $|V|$: $\{x_1^i, x_2^i, \ldots, x_{|V|}^i\}$;

(a_2) for $1 \leq i \leq |V|$, $1 \leq j < j' \leq |V|$, clauses of size two: $\{\overline{x}_j^i, \overline{x}_{j'}^i\}$;

(b_1) for $1 \leq j \leq |V|$, clauses of size $|V|$: $\{x_j^1, x_j^2, \ldots, x_j^{|V|}\}$;

(b_2) for $1 \leq i < i' \leq |V|$, $1 \leq j \leq |V|$, clauses of size two: $\{\overline{x}_j^i, \overline{x}_j^{i'}\}$;

(c) for $1 \leq i < i' \leq |V|$ such that $x^i x^{i'} \notin E$, for $1 \leq j \leq |V|$, clauses of size two: $\{\overline{x}_j^i, \overline{x}_{j+1}^{i'}\}$ and $\{\overline{x}_j^i, \overline{x}_{j-1}^{i'}\}$, with computations carried modulo $|V|$;

(d_1) $\{x_1^1\}$;

(d_2) for $2 \leq j < j' \leq |V|$, clauses of size two: $\{\overline{x}_{j'}^2, \overline{x}_j^3\}$.

Assume that we have a unique Hamiltonian cycle in G, $\mathcal{HC}_1 =< x^{p_1} x^{p_2} x^{p_3} \ldots x^{p_{|V|-1}} x^{p_{|V|}} (x^{p_1}) >$. Note that for the time being, we could also write $\mathcal{HC}_1 =< x^{p_1} x^{p_{|V|}} x^{p_{|V|-1}} \ldots x^{p_3} x^{p_2} (x^{p_1}) >$, or "start" on a vertex other than x^{p_1}, cf. Introduction. This is why, without loss of generality, we set $p_1 = 1$, i.e., we "fix" the first vertex, and we also choose the "direction" of the cycle, by deciding, e.g., that x^2 appears "before" x^3 in the cycle —cf. (d_1)-(d_2). Define the assignment $\mathcal{A}_1$ by $\mathcal{A}_1(x_q^{p_q}) = \mathrm{T}$ for $1 \leq q \leq |V|$, and all the other variables are set FALSE by $\mathcal{A}_1$. We claim that $\mathcal{A}_1$ satisfies all the clauses.

(a_1) for $1 \leq i \leq |V|$, if the vertex x^i has position j in the cycle, then the variable x_j^i satisfies the clause; (a_2) if $\{\overline{x}_j^i, \overline{x}_{j'}^i\}$ is not satisfied by $\mathcal{A}_1$ for some i, j, j', then $\mathcal{A}_1(x_j^i) = \mathcal{A}_1(x_{j'}^i) = \mathrm{T}$, which means that the vertex x^i appears at least twice in the cycle;

(b_1) for $1 \leq j \leq |V|$, if the position j is occupied by the vertex x^i, then the variable x^i_j satisfies the clause; (b_2) if $\{\overline{x}^i_j, \overline{x}^{i'}_j\}$ is not satisfied by $\mathcal{A}_1$ for some i, i', j, then two different vertices are the j-th vertex in the cycle.

(c) If one of the two clauses is not satisfied, say the first one, then the positions j and $j+1$ in the cycle are occupied by two vertices not linked by any edge in G.

(d_1) $\{x^1_1\}$ is satisfied by $\mathcal{A}_1$ thanks to the assumption on the first vertex of the cycle; (d_2) if for some $j < j'$, the clause $\{\overline{x}^2_{j'}, \overline{x}^3_j\}$ is not satisfied, then the vertex x^3 occupies a position j smaller than the position j' of x^2, which contradicts our assumption on x^2 and x^3.

Is $\mathcal{A}_1$ unique? Assume on the contrary that another assignment, $\mathcal{A}_2$, also satisfies the constructed instance of U-SAT. Then by (a_1) and (a_2), for every $i \in \{1, \ldots, |V|\}$, there is at least, then at most, one $j = j(i)$ such that $\mathcal{A}_2(x^i_j) = \mathrm{T}$; by ($b_1$) and ($b_2$), for every $j \in \{1, \ldots, |V|\}$, there is at least, then at most, one $i = i(j)$ such that $\mathcal{A}_2(x^i_j) = \mathrm{T}$; so we have "a place for everything and everything in its place", with exactly $|V|$ variables which are TRUE by $\mathcal{A}_2$ and an ordering of the vertices according to the one-to-one correspondence given by $\mathcal{A}_2$: the vertex x^i is on position j if and only if $\mathcal{A}_2(x^i_j) = \mathrm{T}$. Next, thanks to the clauses (c), two vertices following each other in this ordering, including the last and first ones, are necessarily neighbours, so that this ordering is a Hamiltonian cycle, $\mathcal{HC}_2$. Since we have assumed the uniqueness of the Hamiltonain cycle $\mathcal{HC}_1$ in G, the two cycles can differ only by their starting points or their "directions". However these differences are ruled out by the clauses (d_1) and (d_2), so that the two cycles coincide vertex to vertex, and $\mathcal{A}_1 = \mathcal{A}_2$. So a YES answer for U-HAMC[U] leads to a YES answer for U-SAT.

Assume now that the answer to U-HAMC[U] is negative. If it is negative because there are at least two Hamiltonian cycles, 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 cycle, and different cycles obviously lead to different assignments. If there is no Hamiltonian cycle, then there is no assignment satisfying U-SAT, because such an assignment would give a cycle, as we have seen above with $\mathcal{A}_2$. So in both cases, a NO answer to U-HAMC[U] implies a NO answer to U-SAT. △

Gathering all our previous results, we obtain the following theorem.

Theorem 103 *For every integer $k \geq 3$, the decision problems U-SAT, U-k-SAT and U-1-3-SAT have the same complexity as U-HAMP[U], U-HAMC[U], U-HAMP[O], U-HAMC[O], U-HAMP[D], and U-HAMC[D], up to polynomials. Therefore,*

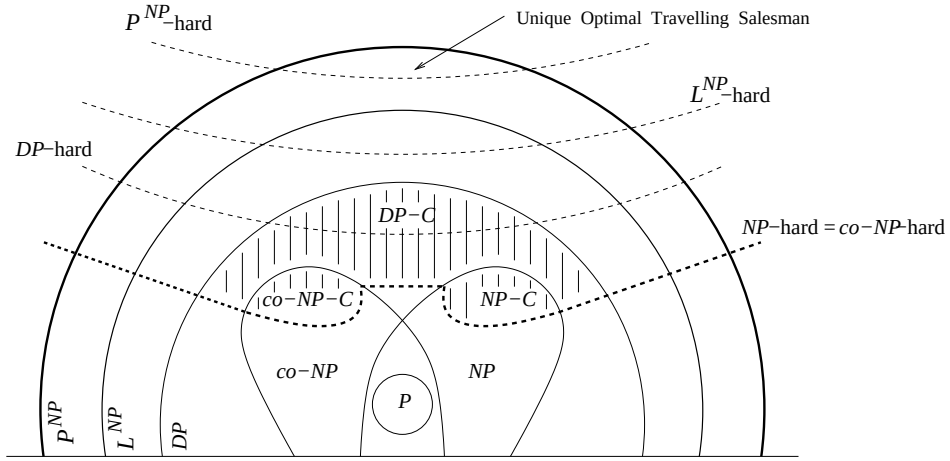

Figure 30: Some classes of complexity: Figure 23 re-visited.

(a) U-HAMP[U], U-HAMC[U], U-HAMP[O], U-HAMC[O], U-HAMP[D], and U-HAMC[D] are NP-hard (and co-NP-hard by Remark 87);

(b) U-HAMP[U], U-HAMC[U], U-HAMP[O], U-HAMC[O], U-HAMP[D], and U-HAMC[D] belong to the class DP. △

Note that the membership to *DP* could have been proved directly.

16 Conclusion

By Theorem 103, for every integer $k \geq 3$, the three decision problems U-SAT, U-k-SAT, U-1-3-SAT have the same complexity, up to polynomials, as the problem of the uniqueness of a path or of a cycle in a graph, undirected, directed, or oriented; all are *NP*-hard (and co-*NP*-hard by Remark 87) and belong to the class *DP*, and it is thought that they are not *DP*-complete. Anyway, they can be found somewhere in the hatched area of Figure 30.

Open problem. Find a better location for any of these problems inside the hierarchy of complexity classes.

In [8], the authors wonder whether

(A) U-SAT is *NP*-hard, but here we believe that what they mean is: does there exist a *polynomial* reduction from an *NP*-complete problem to U-SAT ? i.e., they use the *second* definition of *NP*-hardness;

finally, they show that (A) is true if and only if

(B) U-SAT is *DP*-complete.

So, if one is careless and considers that U-SAT is *NP*-hard without checking according to which definition, one might easily jump too hastily to the conclusion that U-SAT is *DP*-complete, which, to our knowledge, is not known to be true or not. As for U-3-SAT, we do not know where to locate it more precisely either; in [10] the problems U-k-SAT and more particularly U-3-SAT are studied, but it appears that they are versions where the given set of clauses has zero or one solution, which makes quite a difference with our problem.

References

[1] B. ASPVALL, M. F. PLASS and R. E. TARJAN: A linear-time algorithm for testing the truth of certain quantified Boolean formulas, *Information Processing Letters*, Vol. 8, pp. 121–123, 1979.

[2] D. AUGER: Identifying codes in trees and planar graphs, *Electronic Notes in Discrete Mathematics*, Vol. 34, pp. 585–588, 2009.

[3] D. AUGER: Minimal identifying codes in trees and planar graphs with large girth, *European Journal of Combinatorics*, Vol. 31, pp. 1372–1384, 2010.

[4] D. AUGER, I. CHARON, O. HUDRY and A. LOBSTEIN: Complexity results for identifying codes in planar graphs, *International Transactions in Operational Research*, Vol. 17, pp. 691–710, 2010.

[5] J. BANG-JENSEN and G. Z. GUTIN: *Digraphs: Theory, Algorithms and Applications*, Springer-Verlag: Berlin, 2009.

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

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

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

[9] 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.

[10] 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.

[11] I. CHARON, O. HUDRY and A. LOBSTEIN: Minimizing the size of an identifying or locating-dominating code in a graph is *NP*-hard, *Theoretical Computer Science*, Vol. 290, pp. 2109–2120, 2003.

[12] G. COHEN, I. HONKALA, A. LOBSTEIN and G. ZÉMOR: On identifying codes, *Proceedings of DIMACS Workshop on Codes and Association Schemes '99*, Piscataway, USA, Vol. 56, pp. 97–109, 2001.

[13] C. J. COLBOURN, P. J. SLATER and L. K. STEWART: Locating dominating sets in series parallel networks, *Congressus Numerantium*, Vol. 56, pp. 135–162, 1987.

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

[15] T. CORMEN: Algorithmic complexity, in: K. H. Rosen (ed.) *Handbook of Discrete and Combinatorial Mathematics*, pp. 1077–1085, Boca Raton: CRC Press, 2000.

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

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

[18] F. FOUCAUD: Aspects combinatoires et algorithmiques des codes identifiants dans les graphes, Thèse de Doctorat, Université de Bordeaux 1, France, 194 pages, December 2012 (in English).

[19] F. FOUCAUD: Decision and approximation complexity for identifying codes and locating-dominating sets in restricted graph classes, *Journal of Discrete Algorithms*, Vol. 31, pp. 48–68, 2015.

[20] M. FRANCES and A. LITMAN: On covering problems of codes, *Theory of Computing Systems*, Vol. 30, No. 2, pp. 113–119, 1997.

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

[22] 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.

[23] S. GRAVIER, R. KLASING and J. MONCEL: Hardness results and approximation algorithms for identifying codes and locating-dominating codes in graphs, *Algorithmic Operations Research*, Vol. 3, pp. 43–50, 2008.

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

[25] T. W. HAYNES, S. T. HEDETNIEMI and P. J. SLATER: *Fundamentals of Domination in Graphs*, New York: Marcel Dekker, 1998.

[26] E. HEMASPAANDRA, H. SPAKOWSKI and J. VOGEL: The complexity of Kemeny elections, *Theoretical Computer Science*, Vol. 349, pp. 382–391, 2005.

[27] L. HEMASPAANDRA: Complexity classes, in: K. H. Rosen (ed.) *Handbook of Discrete and Combinatorial Mathematics*, pp. 1085–1090, Boca Raton: CRC Press, 2000.

[28] I. HONKALA, O. HUDRY and A. LOBSTEIN: On the number of optimal identifying codes in a twin-free graph, *Discrete Applied Mathematics*, Vol. 180, pp. 111–119, 2015.

[29] I. HONKALA, O. HUDRY and A. LOBSTEIN: On the ensemble of optimal dominating and locating-dominating codes in a graph, *Information Processing Letters*, Vol. 115, pp. 699–702, 2015.

[30] I. HONKALA, O. HUDRY and A. LOBSTEIN: On the ensemble of optimal identifying codes in a twin-free graph, *Cryptography and Communications – Discrete Structures, Boolean Functions and Sequences*, Vol. 8, pp. 139–153, 2016.

[31] I. S. HONKALA and A. C. LOBSTEIN: On the complexity of calculating the minimum norm of a binary code, *Proc. Workshop on Coding and Cryptography '99*, pp. 21–27, Paris, 1999.

[32] O. HUDRY and A. LOBSTEIN: More results on the complexity of identifying problems in graphs, *Theoretical Computer Science*, Vol. 626, pp. 1–12, 2016.

[33] O. HUDRY and A. LOBSTEIN: More results on the complexity of domination problems in graphs, *International Journal of Information and Coding Theory*, to appear.

[34] O. HUDRY and A. LOBSTEIN: Some complexity considerations on the uniqueness of solutions for satisfiability and colouring problems, submitted.

[35] O. HUDRY and A. LOBSTEIN: Complexity of unique (optimal) solutions in graphs: Vertex Cover and Domination, submitted.

[36] O. HUDRY and A. LOBSTEIN: Unique (optimal) solutions: complexity results for identifying and locating-dominating codes, submitted.

[37] O. HUDRY and A. LOBSTEIN: On the complexity of determining whether there is a unique Hamiltonian cycle in a graph, submitted.

[38] D. S. JOHNSON: A catalog of complexity classes, in: *Handbook of Theoretical Computer Science, Vol. A: Algorithms and Complexity*, van Leeuwen, Ed., Chapter 2, Elsevier, 1990.

[39] R. M. KARP: Reducibility among combinatorial problems, in: R. E. Miller and J. W. Thatcher (eds.) *Complexity of Computer Computations*, pp. 85–103, New York: Plenum Press, 1972.

[40] M. G. KARPOVSKY, K. CHAKRABARTY and L. B. LEVITIN: On a new class of codes for identifying vertices in graphs, *IEEE Transactions on Information Theory*, Vol. IT-44, pp. 599–611, 1998.

[41] A. LOBSTEIN: Watching systems, identifying, locating-dominating and discriminating codes in graphs, a bibliography, perso.enst.fr/∼lobstein/debutBIBidetlocdom.pdf

[42] A. McLOUGHLIN: The complexity of computing the covering radius of a code, *IEEE Trans. Inform. Th.*, Vol. 30, pp. 800–804, 1984.

[43] A. MERTZ: On the complexity of multicovering radii, *IEEE Trans. Inform. Th.*, Vol. 50, pp. 1804–1808, 2004.

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

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

[46] 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.

[47] A. PARREAU: Problèmes d'identification dans les graphes, Thèse de Doctorat, Université de Grenoble, France, 214 pages, July 2012.

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

[49] P. J. SLATER: Domination and location in graphs, National University of Singapore, Research Report No. 93, April 1983.

[50] J. SUOMELA: Approximability of identifying codes and locating-dominating codes, *Information Processing Letters*, Vol. 103, pp. 28–33, 2007.

[51] R. E. TARJAN: Depth-first search and linear graph algorithms, *SIAM Journal on Computing*, Vol. 1, pp. 146–160, 1972.

[52] https://complexityzoo.uwaterloo.ca/Complexity_Zoo

Dépôt légal : 2017 - 2e trimestre
Imprimé à Télécom ParisTech – Paris
ISSN 0751-1345 ENST D (Paris) (France 1983-9999)

Télécom ParisTech

Institut Mines-Télécom - membre de l'Université Paris Saclay

46, rue Barrault - 75634 Paris Cedex 13 - Tél. + 33 (0)1 45 81 77 77 - www.telecom-paristech.fr

Département INFRES