More Results on the Complexity of Domination Problems in Graphs

Olivier Hudry
Institut Télécom - Télécom ParisTech & CNRS - LTCI UMR 5141
46, rue Barrault, 75634 Paris Cedex 13 - France
hudry@telecom-paristech.fr

Antoine Lobstein
CNRS - LTCI UMR 5141 & Institut Télécom - Télécom ParisTech
46, rue Barrault, 75634 Paris Cedex 13 - France
lobstein@telecom-paristech.fr

April 17, 2015

Abstract

We investigate, and locate in the complexity classes of the polynomial hierarchy, several problems linked with domination in graphs, such as, given an integer $r \geq 1$ and a graph $G = (V, E)$, the existence of, or search for, optimal r-dominating codes in G, or optimal r-dominating codes in G containing a subset of vertices $X \subset V$.

Key Words: Graph Theory, Complexity, Complexity Classes, Polynomial Hierarchy, *NP*-Completeness, Hardness, Dominating Codes, Covering Radius.

1 Introduction and Preliminary Results

Following [8], which investigates the complexity of Slater's problems in tournaments, our goal in this paper is to study the algorithmic complexity of different variants of the domination problem in graphs.

In [9], we do the same work for identifying problems.

1.1 Outline of the Paper

In Subsection 1.2, we present the necessary notation and definitions about dominating codes; Subsection 1.3 gives preliminary results on dominating codes. In Section 2, we study the complexity of seven problems related to domination. We shall provide the necessary notions of complexity as we go along. The conclusion recapitulates our results.

1.2 Definitions and Notation

We first give the necessary definitions and notation for domination in graphs; see also [4].

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 graph G is called *connected* if for any two vertices x and y, there is a path between them; it is called *disconnected* otherwise. 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, since such a path exists. 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 any subset of vertices $X \subseteq V$, the *subgraph induced by* X is the graph with vertex set X, and edge set $F = \{uv \in E : u \in X, v \in X\}$.

A subset $X \subseteq V$ is called an *independent set* if $F = \emptyset$; it is called a *clique* if F contains all the possible edges.

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

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.

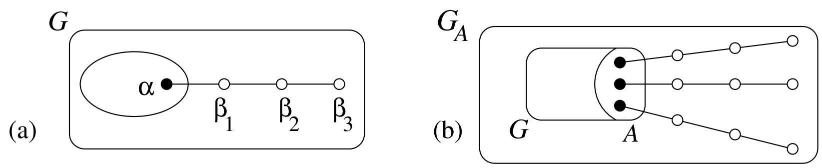

Figure 1: (a) Illustration for Lemma 1, with $r = 3$; (b) illustration for Corollary 2, with $r = 3$, $k = 3$.

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. We denote by $\gamma_r(G)$ the smallest cardinality of an r-dominating code in G, and any r-dominating code C with $|C| = \gamma_r(G)$ is said to be *optimal*. By convention, $\gamma_r(\emptyset) = 0$. The parameter $\gamma_r(G)$ is called the r-*domination number* of G.

1.3 Some Useful Facts on Domination

In the sequel, we shall use the following results on domination.

Lemma 1 *Let $G = (V, E)$ be a graph, and let $r \geq 1$ be an integer. Assume that $Z = \{\alpha, \beta_1, \ldots, \beta_r\} \subseteq V$ induces the path $P_{r+1} = \alpha\beta_1 \ldots \beta_r$ in G, see Figure 1(a). Then G admits at least one optimal r-dominating code which contains α and does not contain any of the vertices β_i, $1 \leq i \leq r$.*

Proof. Let C be an optimal r-dominating code in G. Then $|C \cap Z| \leq 1$, because

$$B_r(\beta_r) \subseteq B_r(\beta_{r-1}) \subseteq \ldots \subseteq B_r(\beta_2) \subseteq B_r(\beta_1) \subseteq B_r(\alpha),$$

and $|C \cap Z| \geq 1$, because β_r must be r-dominated by some codeword. If $C \cap Z = \{\alpha\}$, we are done. If $C \cap Z = \{\beta_i\}$ for some i, then $C^* = (C \setminus \{\beta_i\}) \cup \{\alpha\}$ is also an r-dominating code, and $|C^*| = |C| = \gamma_r(G)$. △

Corollary 2 *Let $G = (V, E)$ be a graph, and let $r \geq 1$ be an integer. For a given set of vertices $A = \{\alpha_1, \ldots, \alpha_k\} \subseteq V$, we consider the following graph, which depends on A (see Figure 1(b)): $G_A = (V_A, E_A)$, with*

$$V_A = V \cup \{\beta_{\alpha_j,i} : 1 \leq j \leq k, 1 \leq i \leq r\},$$

$$E_A = E \cup \{\alpha_j\beta_{\alpha_j,1} : 1 \leq j \leq k\} \cup \{\beta_{\alpha_j,i}\beta_{\alpha_j,i+1} : 1 \leq j \leq k, 1 \leq i \leq r-1\},$$

where for $i \in \{1, \ldots, r\}$ and $j \in \{1, \ldots, k\}$, $\beta_{\alpha_j,i} \notin V$.

Then $A \subseteq V$ is included in at least one optimal r-dominating code in G if and only if $\gamma_r(G) = \gamma_r(G_A)$.

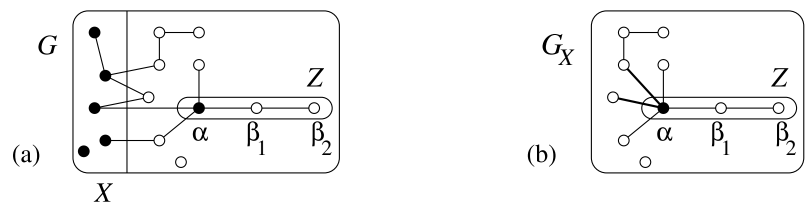

Figure 2: Illustration of Lemma 3, with $r = 2$: (a) the graph G; (b) the graph G_X.

Proof. (a) Let $A = \{\alpha_1, \ldots, \alpha_k\}$ be a set of vertices which is included in at least one optimal r-dominating code C in G; then C is an r-dominating code in G_A as well, and $\gamma_r(G_A) \leq |C| = \gamma_r(G)$.

On the other hand, let C^* be an optimal r-dominating code in G_A. By the previous lemma, we can assume that $A \subseteq C^*$ and none of the $\beta_{\alpha_j,i}$'s belongs to C^*. Then $C^* \subseteq V$, C^* is an r-dominating code in G, and $\gamma_r(G) \leq |C^*| = \gamma_r(G_A)$.

Therefore, with this assumption on A, we have: $\gamma_r(G) = \gamma_r(G_A)$.

(b) Conversely, assume that $\gamma_r(G) = \gamma_r(G_A)$ for a set $A = \{\alpha_1, \ldots, \alpha_k\} \subseteq V$. Let C^* be an optimal r-dominating code in G_A; again by the previous lemma, we can assume that $A \subseteq C^*$, and none of the $\beta_{\alpha_j,i}$'s belongs to C^*. Then C^* is an r-dominating code in G, it has size $\gamma_r(G_A) = \gamma_r(G)$, and it contains A. △

Thus, the characterization of a set of vertices included in at least one optimal dominating code is obtained through the equality of two domination numbers. Another characterization is available in the next lemma, which will be useful in the proof of Proposition 17.

Lemma 3 *Let $G = (V, E)$ be a graph, and let $r \geq 1$ be an integer. Assume that $Z = \{\alpha, \beta_1, \ldots, \beta_r\} \subseteq V$ induces the path $P_{r+1} = \alpha\beta_1 \ldots \beta_r$ in G, see Figure 2(a). For a set of vertices $X \subseteq (V \setminus Z)$, let $G_X = (V_X, E_X)$ be the following graph (see Figure 2(b)): $V_X = V \setminus X$, and E_X is constructed by adding to E the edges αy whenever $xy \in E$, for $x \in X$ and $y \in (V \setminus Z) \setminus X$; note that some of these edges αy may already exist in G.*

Then $\gamma_r(G) = \gamma_r(G_X) + |X|$ if and only if X is included in at least one optimal r-dominating code in G.

Proof. (a) Assume that X is included in at least one optimal r-dominating code C in G. By Lemma 1, we can assume that $C \cap Z = \{\alpha\}$. Then $C \setminus X$ is still an r-dominating code in G_X, because every path between two vertices u and v that was going through a vertex in X in G now goes through α (in particular, in G_X, α r-dominates all the vertices r-dominated by X in G). So $\gamma_r(G_X) \leq |C| - |X| = \gamma_r(G) - |X|$.

On the other hand, let C^* be an optimal r-dominating code in G_X. By Lemma 1, we can assume that $C^* \cap Z = \{\alpha\}$. Then $C^* \cup X$ is an r-dominating code in G, because the paths going through α in G_X can be replaced when necessary by paths going through a vertex in X, and $\gamma_r(G) \leq |C^*| + |X| = \gamma_r(G_X) + |X|$.

Therefore, the assumption that X is included in at least one optimal r-dominating code in G implies that $\gamma_r(G) = \gamma_r(G_X) + |X|$.

(b) Conversely, assume that $\gamma_r(G) = \gamma_r(G_X) + |X|$ and consider an optimal r-dominating code C^* in G_X. Again, by Lemma 1, we can assume that $\alpha \in C^*$, and so $C^* \cup X$ is an r-dominating code in G, with size $\gamma_r(G_X) + |X| = \gamma_r(G)$, i.e., $C^* \cup X$ is optimal (and contains X). △

Lemma 4 *Let $r \geq 1$ be an integer and let $G = (V, E)$ be the graph defined as follows:*

$$V = \{u, v\} \cup \{\alpha_i : 1 \leq i \leq r-1\} \cup \{\beta_{i,j} : 1 \leq i \leq r-1, 1 \leq j \leq 2r\},$$

$$E = \{u\alpha_1, \alpha_1\alpha_2, \ldots, \alpha_{r-1}v\} \cup \{\alpha_i\beta_{i,1}, \beta_{i,1}\beta_{i,2}, \ldots, \beta_{i,2r-1}\beta_{i,2r} : 1 \leq i \leq r-1\},$$

cf. Figure 3. Then $\gamma_r(G) = r$, and $C_1 = \{u\} \cup \{\beta_{i,r} : 1 \leq i \leq r-1\}$ and $C_2 = \{v\} \cup \{\beta_{i,r} : 1 \leq i \leq r-1\}$ are optimal r-dominating codes in G.

As a consequence, if D is an r-dominating code in G, optimal or not, we can replace it when convenient by an r-dominating code C such that $|C| \leq |D|$ and $\{\beta_{i,r} : 1 \leq i \leq r-1\} \subset C$.

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. On the other hand, it is quite straightforward to check that C_1 and C_2 are r-dominating codes, of size r. In particular, the vertices $\beta_{i,r}$, $1 \leq i \leq r-1$, r-dominate exactly all the vertices α_i and $\beta_{i,j}$. △

2 Complexity Results for Dominating Codes

We present seven problems on dominating codes. The question in the first problem is justified by the fact that any r-dominating code is also $(r+1)$-dominating; covering radius has been extensively expounded in [1] in the case of the hypercube; see also the references given in the last sentence of this article.

1) Problem CR (Covering Radius):
Instance: A graph $G = (V, E)$ and a code $C \subseteq V$.
Question: What is the smallest nonnegative integer r such that C is an r-dominating code in G?

The following six problems are stated for a fixed integer r, $r \geq 1$; their names are indexed by r.

2) 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?

3) Problem DN$_r$ (r-Domination Number):
Instance: A graph G.
Output: The r-domination number of G, $\gamma_r(G)$.

4) Problem ODCS$_r$ (Search for an Optimal r-Dominating Code):
Instance: A graph G.
Search: Determine an optimal r-dominating code in G.

5) Problem Sub-ODCE$_r$ (Existence of an Optimal r-Dominating Code containing a given Subset):
Instance: A graph $G = (V, E)$ and a nonempty subset of vertices $X \subseteq V$.
Question: Does G admit an optimal r-dominating code containing X?

6) Problem Sub-ODCS$_r$ (Search for an Optimal r-Dominating Code containing a given Subset):
Instance: A graph $G = (V, E)$ and a nonempty subset of vertices $X \subseteq V$.
Search: Determine, when it exists, an optimal r-dominating code in G containing X.

An algorithm solving Sub-ODCS$_r$ outputs a suitable code if there is one, or states that no such code exists.

7) Problem Sub-SmDCS$_r$ (Search for a Smallest r-Dominating Code containing a given Subset):
Instance: A graph $G = (V, E)$ and a nonempty subset of vertices $X \subseteq V$.
Search: Determine an r-dominating code in G containing X, with the smallest size.

We want to locate these problems inside the polynomial hierarchy of problems. For the general theory of completeness and hardness in the polynomial hierarchy, we refer to [3]; see also [6] for a comprehensive survey of the main complexity classes, [10] and [14]. From a practical viewpoint, we do not know of polynomial algorithms solving exactly a *NP*-hard problem (and such algorithms simply do not exist if *P*≠*NP*): the time required can grow exponentially with the size of the instance (here, the size of the instance is polynomially linked to n, the order of the graph).

Before we try to locate these problems inside the polynomial hierarchy, we can already make a few easy remarks about their respective compared complexities; here, the meaning of "at least as difficult as" is the following: a problem π_1 is at least as difficult as a problem π_2 if an algorithm solving π_1 provides an algorithm for solving π_2 with the same qualitative complexity.

Lemma 5 *Let $r \geq 1$ be an integer.*
(0) The problem DN$_r$ is at least as difficult as DC$_r$.

(1) The problem DN_r is at least as difficult as Sub-ODCE$_r$.

(2) The problem ODCS$_r$ is at least as difficult as DN_r.

(3) The problem Sub-ODCS$_r$ is at least as difficult as Sub-ODCE$_r$.

(4) The problem Sub-SmDCS$_r$ is at least as difficult as DN_r, even in the case when X is a singleton.

Proof. (0) With only one call to any algorithm providing $\gamma_r(G)$, the answer to DN$_r$, we can give the answer to DC$_r$, by comparing $\gamma_r(G)$ and the integer k in the instance of DC$_r$. So DN$_r$ is at least as difficult as DC$_r$.

(1) Consider an instance $(G, X \subseteq V)$ of Sub-ODCE$_r$. If G_X is defined as in Corollary 2, then, by this same corollary, it is sufficient to compute and compare $\gamma_r(G)$ and $\gamma_r(G_X)$: the answer to Sub-ODCE$_r$ is "yes" if and only if equality holds. Now this can be done by using twice an algorithm solving the problem DN$_r$, together with negligible operations such as constructing G_X.

The statements (2) and (3) are obvious.

(4) Consider an algortithm solving Sub-SmDCS$_r$ and run it separately n times, each time with a different singleton $X = \{x\} \subset V$. The smallest code thus obtained gives the r-domination number of G.

Therefore, Sub-SmDCS$_r$, with $X = \{x\}$, is at least as difficult as DN$_r$. △

We give the following lemma without proof.

Lemma 6 *Given an integer $r \geq 1$ and a graph $G = (V, E)$, checking that a given code $C \subseteq V$ is r-dominating is polynomial in the order of the graph.*

Proposition 7 *The problem CR is polynomial.*

Proof. Here, all we have to do in order to solve CR is to check whether C is r-dominating, for $r = 0$, $r = 1$, …, and the number of these checkings cannot exceed $|V|$. Even better, a standard dichotomous process on r is possible. △

The status of DC$_1$ is already known.

Proposition 8 *[3, p. 75 and p. 190] The decision problem DC$_1$ is NP-complete.*

Proof. We give the proof here, because we shall use the polynomial reduction also in the proof of Proposition 12. The membership to *NP* is straightforward (Lemma 6). We describe a polynomial reduction from the *NP*-complete problem Vertex Cover [11], [3, p. 46 and p. 190].

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

A *vertex cover* is a subset $V^* \subseteq V$ such that for each edge $uv \in E$, at least one of u and v belongs to V^*. The polynomial reduction from VC to

DC_1 is the following: if $(G = (V, E), k)$ is an instance of VC, we take as an instance for DC_1 the integer $k^+ = k$ and the graph $G^+ = (V^+, E^+)$ defined by $V^+ = V \cup \{x_e : e = uv \in E\}$, $E^+ = E \cup \{ux_e, x_ev : e = uv \in E\}$. In other words, for each edge $e = uv$ in G, we create in G^+ the triangle uv, ux_e, x_ev. We prove that an instance in VC is positive if and only if the corresponding instance in DC_1 is.

Assume that VC admits a vertex cover V^* of size at most k in $G = (V, E)$: for each edge $e = uv \in E$, $u \in V^*$ or $v \in V^*$. In G^+, each of the three vertices u, v, x_e is 1-dominated by u and v, and therefore, V^* is a 1-dominating code in G^+, of size at most $k = k^+$.

Conversely, if C is a 1-dominating code of size at most k^+ in G^+, then, since x_e must be 1-dominated by some codeword, at least one of the three vertices u, v, x_e is a codeword. If only x_e is a codeword, then $(C \setminus \{x_e\}) \cup \{u\}$ or $(C \setminus \{x_e\}) \cup \{v\}$ is also a 1-dominating code. This means that there is a 1-dominating code C^*, of size $|C|$, which contains u or v for each triangle uv, ux_e, x_ev in G^+, i.e., for each edge $e = uv \in E$. Therefore, C^* is a vertex cover in G, of size at most k. $\triangle$

Once the membership of DC_1 to *NP*, the class of nondeterministic polynomial problems, is established, the *NP*-completeness gives a sort of lower bound on its complexity: the problem DC_1 is at least as difficult as well-known difficult problems, such as “3-Satisfiability”, “3-Dimensional Matching”, “Hamiltonian Circuit” or “Partition”, and more generally, at least as difficult as any problem in *NP*. Still, *NP*-completeness results are conditional in some sense; if for example *P*=*NP*, they would lose their interest.

Next, we generalize Proposition 8 and show that for any integer $r \geq 2$, the problem DC_r is *NP*-complete.

Proposition 9 *Let $r \geq 1$ be an integer. The decision problem DC_r is NP-complete.*

Proof. The case $r = 1$ has already been studied, so we can assume that $r \geq 2$. Again, Lemma 6 gives the membership to *NP*. The polynomial reduction from DC_1 to DC_r is the following (see Figure 3) : if $(G = (V, E), k)$ is an instance of DC_1, we construct the instance $(G^* = (V^*, E^*), k^*)$ of DC_r by setting, for each edge $e = uv \in E$,

$$V_e^* = \{\alpha_{e,i} : 1 \leq i \leq r-1\} \cup \{\beta_{e,i,j} : 1 \leq i \leq r-1, 1 \leq j \leq 2r\}, \quad (1)$$

$$\begin{aligned} E_e^* = \{u\alpha_{e,1}, \alpha_{e,1}\alpha_{e,2}, \ldots, \alpha_{e,r-2}\alpha_{e,r-1}, \alpha_{e,r-1}v\} \cup \\ \{\alpha_{e,i}\beta_{e,i,1}, \beta_{e,i,1}\beta_{e,i,2}, \ldots, \beta_{e,i,2r-1}\beta_{e,i,2r} : 1 \leq i \leq r-1\}. \end{aligned} \quad (2)$$

Then we set

$$V^* = (\cup_{e \in E} V_e^*) \cup V, \; E^* = \cup_{e \in E} E_e^*, \quad (3)$$

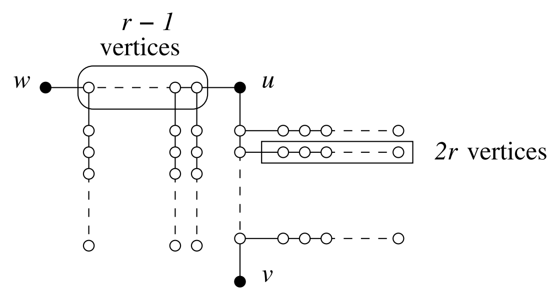

Figure 3: Construction of V^* and E^*, starting from the edges $uv \in E$ and $uw \in E$.

and $k^* = k + (r-1)|E|$. We claim that an instance of DC_1 is positive if and only if the corresponding instance of DC_r is.

Assume first that C is a 1-dominating code in G, of size at most k. Then (cf. Lemma 4) it is straightforward to check that $C \cup \{\beta_{e,i,r} : e \in E, 1 \leq i \leq r-1\}$ is an r-dominating code in G^*, and its size is at most $k + |E|(r-1) = k^*$.

Conversely, assume that C^* is an r-dominating code of size at most k^* in G^*. Following the idea in Lemma 4, we can assume that, for $e \in E$ and $1 \leq i \leq r-1$, all the vertices $\beta_{e,i,r}$ belong to C^*. This subset of C^* r-dominates exactly all the vertices of type α and β, i.e., all the vertices in $\cup_{e \in E} V_e^* = V^* \setminus V$. Therefore, the purpose of (some of —remember that C^* is not necessarily optimal) the remaining codewords of C^* is to r-dominate all the vertices in V. If $e = uv \in E$, the vertices $\beta_{e,i,j}$, $r+1 \leq j \leq 2r$, do not r-dominate any vertex in V, and a vertex $\alpha_{e,i}$ or $\beta_{e,i,j}$, $1 \leq j \leq r-1$, r-dominates at most two vertices in V, namely u and v, and this task can be performed by u or v. So if one (or more) $\alpha_{e,i}$ or $\beta_{e,i,j}$, $j \neq r$, belong(s) to C^*, we can replace it (them) by one of the vertices u or v.

Therefore, we can construct a new r-dominating code, $C^\times$, in G^*, which is included in $V \cup \{\beta_{e,i,r} : e \in E, 1 \leq i \leq r-1\}$ and has as many elements as C^*, or fewer. Now in G^*, the vertices in V are r-dominated by codewords belonging to $C^\times \cap V$, which proves that in G, $C^\times \cap V$ is a 1-dominating code, the size of which is at most $k^* - (r-1)|E| = k$. $\triangle$

To go further, we need the following notation and additional notions of complexity (see, e.g., [10] or [14]).

The class P^{NP} (also known as Δ_2 in the polynomial hierarchy) contains the decision problems which can be solved by applying, with a number of calls which is polynomial with respect to the size of the instance, a subprogram able to solve an appropriate problem in *NP* (usually, an *NP*-complete problem). The class L^{NP} (also known as Θ_2 and $P^{NP}_{||}$) contains the decision problems which can be solved by applying, with a number of calls which is logarithmic with respect to the size of the instance, a subprogram able to solve an appropriate problem in *NP*. For problems which are not decison problems, these classes are generalized, using a "*F*" (for "function") in front of their names; thus, the class FP^{NP} (respectively, FL^{NP}) contains the optimization problems and the search 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*. Membership to *NP*, L^{NP}, P^{NP}, FL^{NP} or FP^{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 ...).

The next proposition is easy and uses a standard argument, see for instance [8].

Proposition 10 *For $r \geq 1$, the problem DN_r belongs to the class FL^{NP}.*

Proof. Let $\mathcal{A}_r$ be an algorithm which solves the problem DC_r: for any instance (G, k) of DC_r, it says whether there is an r-dominating code of size k or less in G. This algorithm can be used to solve DN_r with a number of calls bounded from above by a logarithm in the size of the instance. If n is the order of G, for the instance $(G, k = n)$ of DC_r, the answer is "yes". Thanks to the standard dichotomous process starting from this initial value, we may compute the size of an optimal r-dominating code in G with at most $\lceil \log n \rceil$ calls to $\mathcal{A}_r$. Since DC_r is in *NP* (it is actually *NP*-complete, see Proposition 9), we can conclude that $\mathrm{DN}_r \in FL^{NP}$. △

Proposition 11 *For $r \geq 1$, the problem Sub-ODCE$_r$ belongs to the class L^{NP}.*

Proof. We have seen in the proof of Lemma 5(1) that an instance $(G, X \subseteq V)$ of Sub-ODCE$_r$ can be solved by using twice an algorithm solving the problem DN_r, together with negligible operations. In turn, as we have just seen, solving DN_r can be done with a logarithmic number of calls to an algorithm solving DC_r, which is in *NP* (Proposition 9). △

We can even show that Sub-ODCE$_r$ is among the most difficult problems in its class L^{NP}, thus establishing a lower bound on the complexity of this problem, and locating it exactly in the hierarchy; consequently, we have also a lower bound for the complexity of DN_r (Corollary 15).

Proposition 12 *For $r \geq 1$, the decision problem Sub-ODCE$_r$ is L^{NP}-complete.*

Proof. The membership to L^{NP} having just been established, we describe polynomial reductions from the L^{NP}-complete problem Vertex Cover Member [5, Cor. 4.13] to Vertex Cover Subset (see below), then from Vertex

Cover Subset to Sub-ODCE$_1$, and finally from Sub-ODCE$_1$ to Sub-ODCE$_r$, for $r \geq 2$.

Problem VCM (Vertex Cover Member):
Instance: A graph $G = (V, E)$ and a vertex $x \in V$.
Question: Does G admit an optimal vertex cover containing x?

Problem VCS (Vertex Cover Subset):
Instance: A graph $G = (V, E)$ and a nonempty subset of vertices $X \subseteq V$.
Question: Does G admit an optimal vertex cover containing X?

To go from VCM to VCS, it is sufficient to note that VCM is a subproblem of VCS, with $X = \{x\}$.

The polynomial reduction from VCS to Sub-ODCE$_1$ is very similar to the one in the proof of Proposition 8: if $(G = (V, E), X)$ is an instance of VCS, we take as an instance for Sub-ODCE$_1$ the set of vertices $X^+ = X$ and the graph $G^+ = (V^+, E^+)$ defined by $V^+ = V \cup \{x_e : e = uv \in E\}$, $E^+ = E \cup \{ux_e, x_ev : e = uv \in E\}$. We prove that an instance in VCS is positive if and only if the corresponding instance in Sub-ODCE$_1$ is.

Let V^* be a (not necessarily optimal) vertex cover V^*, containing X, in G (such a set always exists): the argument in the proof of Proposition 8 shows that V^* is a 1-dominating code in G^+, containing X.

Conversely, if C is a (not necessarily optimal) 1-dominating code in G^+, containing X^+, the argument in the proof of Proposition 8 shows that, after replacing if necessary some vertices of type x_e, we can find a 1-dominating code C^* which contains u or v for each edge $e = uv \in E$. Since $X^+ = X$ does not contain vertices of type x_e, C^* is a vertex cover in G which still contains X^+, and $|C^*| = |C|$.

Therefore, a vertex cover in G, containing X, leads to a 1-dominating code in G^+, containing X, and both sets have the same size; a 1-dominating code in G^+, containing X, leads to a vertex cover in G, containing X, and both sets have the same size. The optimality for one implies the optimality for the other.

Next, we polynomially reduce Sub-ODCE$_1$ to Sub-ODCE$_r$, for $r \geq 2$. If $(G = (V, E), X)$ is an instance of Sub-ODCE$_1$, we construct the instance $(G^* = (V^*, E^*), X^*)$ of Sub-ODCE$_r$ by constructing G^* exactly as in the proof of Proposition 9, see (1), (2), (3) and Figure 3, and setting $X^* = X$.

We claim that an instance of Sub-ODCE$_1$ is positive if and only if the corresponding instance of Sub-ODCE$_r$ is. We have already seen that we can choose to have, in an r-dominating code C^* in G^*, all the vertices $\beta_{e,i,r}$, $e \in E$, $1 \leq i \leq r - 1$.

First, let C be a 1-dominating code in G (not necessarily optimal), containing X. Then it is straightforward to check that $C \cup \{\beta_{e,i,r} : e \in E, 1 \leq i \leq r - 1\}$ is an r-dominating code in G^*, containing $X^* = X$.

Conversely, let C^* be a (not necessarily optimal) r-dominating code containing X^* in G^*. We have seen in the proof of Proposition 9 that we can replace exclusively vertices of type α or β by vertices in V and construct a code $C^\times \subseteq V \cup \{\beta_{e,i,r} : e \in E, 1 \leq i \leq r-1\}$ which is r-dominating in G^*; this code still contains X^*. In G^*, the vertices in V are exclusively r-dominated by codewords of $C^\times$ belonging to V, which proves that in G, $C^\times \cap V$ is a 1-dominating code.

Assume now that C is optimal in G. Then obviously, $C \cup \{\beta_{e,i,r} : e \in E, 1 \leq i \leq r-1\}$ is optimal in G^*. Conversely, if C^* is optimal in G^*, then $|C^\times| = |C^*|$ and $C^\times$ is also optimal, which implies that $C^\times \cap V$ is an optimal 1-dominating code in G. $\triangle$

Remark 13 *Note that, using exactly the same ideas, we could have proved Proposition 12 by going from VCM to Sub-ODCE$_r$ with the following chain of polynomial reductions: from VCM to Sub-ODCE$_1$ with $X = \{x\}$, from Sub-ODCE$_1$ with $X = \{x\}$ to Sub-ODCE$_r$ with $X = \{x\}$, and from Sub-ODCE$_r$ with $X = \{x\}$ to Sub-ODCE$_r$; in doing so, we would have proved the following result, which strengthens Proposition 12.*

Proposition 14 *For $r \geq 1$, the decision problem Sub-ODCE$_r$ remains L^{NP}-complete when X is a singleton.*

Corollary 15 *For $r \geq 1$, the problem DN$_r$ is L^{NP}-hard.*

Proof. Use Lemma 5(1) and the fact that Sub-ODCE$_r$ is L^{NP}-complete. $\triangle$

We now turn to the three search problems ODCS$_r$ (determine an optimal r-dominating code), Sub-ODCS$_r$ (given a subset of vertices X, determine an optimal r-dominating code containing X) and Sub-SmDCS$_r$ (given a subset of vertices X, determine a smallest r-dominating code containing X). The previous results, together with Lemma 5, immediately imply the following corollary, which gives a lower bound on the complexity of these problems.

Corollary 16 *(a) For $r \geq 1$, the problem ODCS$_r$ is L^{NP}-hard.*

(b) For $r \geq 1$, the problem Sub-ODCS$_r$ is L^{NP}-hard, even in the case when X is a singleton.

(c) For $r \geq 1$, the problem Sub-SmDCS$_r$ is L^{NP}-hard, even in the case when X is a singleton.

Proof. (a) Use Lemma 5(2) and the fact that DN$_r$ is L^{NP}-hard.

(b) Use Lemma 5(3) and the fact that Sub-ODCE$_r$ is L^{NP}-complete, even when X is a singleton.

(c) Use Lemma 5(4) and the fact that DN$_r$ is L^{NP}-hard. $\triangle$

Then we show that the complexity of these three problems does not go beyond FP^{NP}.

Proposition 17 *(i) For $r \geq 1$, the problem ODCS$_r$ belongs to the class FP^{NP}.*
(ii) For $r \geq 1$, the problem Sub-ODCS$_r$ belongs to the class FP^{NP}.
(iii) For $r \geq 1$, the problem Sub-SmDCS$_r$ belongs to the class FP^{NP}.

Proof. (i) Let $\mathcal{A}_r$ be an algorithm solving Sub-ODCE$_r$. In particular, $\mathcal{A}_r$ can solve instances of Sub-ODCE$_r$ for which X is a singleton. In a first stage, we show how to solve ODCS$_r$ by calling $\mathcal{A}_r$ a polynomial number of times, which will prove that Sub-ODCE$_r$ is at least as difficult as ODCS$_r$, polynomials apart.

Let $G_0 = (V_0, E_0)$ be an instance of ODCS$_r$, with n vertices.

In a first step, we run $\mathcal{A}_r$ with G_0 and different vertices of V_0 until we get a positive answer, i.e., we find a vertex α belonging to at least one (unknown) optimal r-dominating code in G_0. We then construct the graph $G_1 = (V_1, E_1)$ as follows: $V_1 = V_0 \cup \{\beta_1, \ldots, \beta_r\}$, where $\beta_k \notin V_0$, and $E_1 = E_0 \cup \{\alpha\beta_1, \beta_1\beta_2, \ldots, \beta_{r-1}\beta_r\}$. Because α belongs to an optimal r-dominating code in G_0, we have, by Corollary 2, $\gamma_r(G_0) = \gamma_r(G_1)$.

In a second step, we run $\mathcal{A}_r$ and look for a vertex, x_1 (with $x_1 \neq \alpha, x_1 \neq \beta_k$, $1 \leq k \leq r$), belonging to at least one optimal r-dominating code in G_1. If there is none, we stop. Otherwise, once we have found x_1, we construct the graph $G_2 = (V_2, E_2)$ from G_1 in the following way (cf. Figure 2): $V_2 = V_1 \setminus \{x_1\}$ (so that G_2 has one vertex less than G_1) and to E_1 we add the edges αy whenever $x_1 y \in E_1$, for $y \in V_1 \setminus \{\alpha, x_1, \beta_k : 1 \leq k \leq r\}$; note that some of these edges αy may already exist in G_1. By Lemma 3, we have: $\gamma_r(G_2) = \gamma_r(G_1) - 1$.

At Step i ($i \geq 3$), we have the graph $G_{i-1} = (V_{i-1}, E_{i-1})$ and we look for a vertex x_{i-1} (with $x_{i-1} \neq \alpha, x_{i-1} \neq \beta_k$), contained in at least one optimal r-dominating code in G_{i-1}. If there is none, we stop; if there is one, then we construct the graph $G_i = (V_i, E_i)$ from G_{i-1} as follows: $V_i = V_{i-1} \setminus \{x_{i-1}\}$ and to E_{i-1} we add the edges αy whenever $x_{i-1} y \in E_{i-1}$, for $y \in V_{i-1} \setminus \{\alpha, x_{i-1}, \beta_k\}$. Again, some of these edges may have been constructed previously, and again, by Lemma 3, we have: $\gamma_r(G_i) = \gamma_r(G_{i-1}) - 1 = \gamma_r(G_0) - i + 1$.

Thus, step after step, we add the neighbours of $x_1, x_2, \ldots$ to $N(\alpha)$. After at most n steps, we come to a stop, because the graphs thus constructed have fewer and fewer vertices. If we stop at Step j, we claim that $C = \{\alpha, x_1, x_2, \ldots, x_{j-2}\}$ is an optimal r-dominating code in G_0.

Indeed, we stop at Step j because none of the vertices in $V_{j-1} \setminus \{\alpha, \beta_k : 1 \leq k \leq r\}$ belongs to any optimal r-dominating code in G_{j-1}; this shows that $\{\alpha\}$ is such a code (cf. Lemma 1), and $\gamma_r(G_{j-1}) = 1 = \gamma_r(G_0) - j + 2$, i.e., $\gamma_r(G_0) = j - 1 = |C|$, so that C has the right size.

Finally, assume that there is a vertex z in $V_0 \setminus C$ which is not r-dominated by any of the codewords in C. This implies that $d_{G_0}(z, \alpha) > r$; the construction of G_1 shows that also, $d_{G_1}(z, \alpha) > r$. Then $d_{G_0}(z, x_1) > r$ implies that $d_{G_1}(z, x_1) > r$, which in turn implies that $d_{G_2}(z, \alpha) > r$. We can similarly show that $d_{G_\ell}(z, \alpha) > r$, $3 \leq \ell \leq j-1$; in particular, $d_{G_{j-1}}(z, \alpha) > r$, which contradicts the fact that we stopped at Step j.

How many times do we need to call $\mathcal{A}_r$? If n is the order of the graph, we have at most n steps, in which we call $\mathcal{A}_r$ a decreasing number of times, starting with at most n calls in the first step, so that we have something like at most $n^2/2$ calls to $\mathcal{A}_r$, plus operations such as the deletion of vertices and edges. Note however that, since a vertex which has been tried and rejected because it does not belong to any optimal r-dominating code in the current graph needs not be tested again in the following steps, the number of calls can be reduced to n, approximately.

This proves that, by calling the algorithm $\mathcal{A}_r$ a polynomial number of times (polynomial with respect to n), we have designed an algorithm which outputs an optimal r-dominating code in G_0, i.e., solves ODCS$_r$. This ends our first stage.

In turn, Sub-ODCE$_r$ can be solved using a logarithmic number of calls to an algorithm solving DC$_r$ (Proposition 11), which is in *NP*. So, all in all, we can solve ODCS$_r$ by calling a polynomial number of times an algorithm solving a problem in *NP*. This proves that ODCS$_r$ belongs to FP^{NP}.

(ii) Now we want to call $\mathcal{A}_r$ in order to solve Sub-ODCS$_r$. First, we run $\mathcal{A}_r$ with X. If the answer is negative, we know that no optimal r-dominating code contains X, and we stop. We assume now that the answer is positive. Then we proceed as in (i): we choose a first vertex in X which will play the part of α, we choose a second vertex in X for x_1, and so on until we have used all the vertices in X. Then we look for a vertex belonging to at least one optimal r-dominating code in the current graph, and we can go on running the algorithm and conclude exactly as previously.

The only difference with (i) is that the first vertices must belong to a specific subset, provided that this subset is contained in an optimal r-dominating code.

(iii) Finally, we want to call $\mathcal{A}_r$ in order to solve Sub-SmDCS$_r$. We proceed exactly as in (ii), except that we do not need to check whether X is included in an optimal r-dominating code. △

Remark 18 *In the item (i) of the proof of Proposition 17, we prove that*

(a) Sub-ODCE$_r$ is at least as difficult as ODCS$_r$, polynomials apart*.*
Proposition 11 states that

(b) the problem Sub-ODCE$_r$ belongs to the class L^{NP}.
These two facts, (a) and (b), do not imply however that ODCS$_r$ would belong to L^{NP}, because of the polynomial *number of calls necessary to solve ODCS$_r$ using an algorithm solving Sub-ODCE$_r$.*

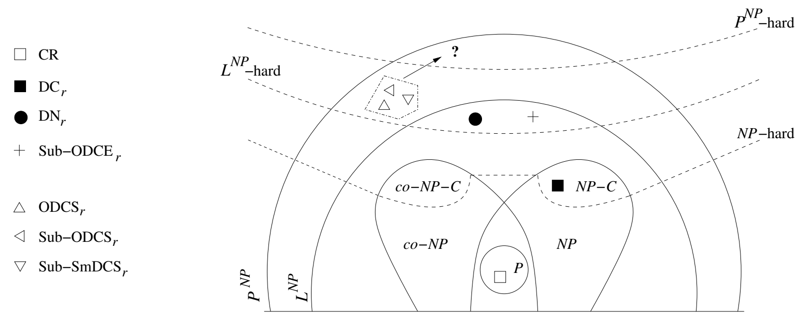

Figure 4: The locations of our problems in the classes of complexity.

3 Conclusion

We recapitulate our results, and present one conjecture. For any fixed integer r, $r \geq 1$,

- CR is polynomial (Proposition 7).
- DC$_r$ is *NP*-complete (Propositions 8 [3] and 9).
- DN$_r$ belongs to FL^{NP} (Proposition 10) and is L^{NP}-hard (Corollary 15).
- Sub-ODCE$_r$ is L^{NP}-complete (Proposition 12), even if $|X| = 1$ (Proposition 14).
- ODCS$_r$ is L^{NP}-hard (Corollary 16(a)) and belongs to the class FP^{NP} (Proposition 17(i)).
- Sub-ODCS$_r$ is L^{NP}-hard, even if $|X| = 1$ (Corollary 16(b)), and belongs to the class FP^{NP} (Proposition 17(ii)).
- Sub-SmDCS$_r$ is L^{NP}-hard, even if $|X| = 1$ (Corollary 16(c)), and belongs to the class FP^{NP} (Proposition 17(iii)).

These results are represented in Figure 4, though in a simplified and thus improper way: we make no difference between decision problems and non-decision problems, between P^{NP} and FP^{NP}, ... The four problems CR, DC$_r$, DN$_r$, and Sub-ODCE$_r$ are located exactly. About the three search problems ODCS$_r$, Sub-ODCS$_r$, and Sub-SmDCS$_r$, we only know that all are "between" L^{NP}-hard and FP^{NP}; so each may be in one of three areas: (a) inside FL^{NP} and above the line L^{NP}-hard, or (b) outside FL^{NP} and below the line P^{NP}-hard, as is the case in Figure 4 for lack of a better knowledge, or (c) inside FP^{NP} and above the line P^{NP}-hard, which is our conjecture, represented by an arrow and a question mark in the Figure:

Conjecture 19 *For $r \geq 1$, the problems ODCS$_r$, Sub-ODCS$_r$ and Sub-SmDCS$_r$ are P^{NP}-hard, even when X is a singleton.*

Having seen in the proofs of Propositions 8 and 12 how close the domination and vertex cover problems are, it would be easy to extend our results to the corresponding vertex cover problems, when these results were not previously established —we have seen that VCM, the equivalent of Sub-ODCE$_r$ (with $X = \{x\}$) for vertex covers, was L^{NP}-complete [5] and we have proved in passing, in the proof of Proposition 12, that VCS is also L^{NP}-complete. The same is true for two problems closely related to Vertex Cover, namely Clique (about the size of the largest clique in a graph, see [3, p. 47 and p. 194]) and Independent Set (about the size of the largest independent set in a graph, see [3, p. 53–54 and p. 194–195]).

See also [2], [7], [12] and [13] for a study of the complexity of some problems related to domination in the binary hypercube.

References

[1] G. D. COHEN, I. S. HONKALA, S. LITSYN and A. C. LOBSTEIN: *Covering Codes*, Amsterdam: Elsevier, 1997.

[2] M. FRANCES and A. LITMAN: On covering problems of codes, *Theory of Computing Systems*, vol. 30, No. 2, pp. 113–119, 1997.

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

[4] T. W. HAYNES, S. T. HEDETNIEMI and P. J. SLATER: *Fundamentals of Domination in Graphs*, New York: Marcel Dekker, 1998.

[5] E. HEMASPAANDRA, H. SPAKOWSKI and J. VOGEL: The complexity of Kemeny elections, *Theoretical Computer Science*, Vol. 349, pp. 382–391, 2005.

[6] L. HEMASPAANDRA: Complexity classes, in: K. H. Rosen (ed.) *Handbook of Discrete and Combinatorial Mathematics*, pp. 1085–1090, Boca Raton: CRC Press, 2000.

[7] 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.

[8] O. HUDRY: On the complexity of Slater's problems, *European Journal of Operational Research*, Vol. 203, pp. 216–221, 2010.

[9] O. HUDRY and A. LOBSTEIN: More results on the complexity of identifying problems in graphs, submitted.

[10] D. S. JOHNSON: A catalog of complexity classes, in: J. van Leeuwen (ed.) *Handbook of Theoretical Computer Science, Vol. A: Algorithms and Complexity*, pp. 67–161, Amsterdam: Elsevier, 1990.

[11] 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.

[12] A. MCLOUGHLIN: The complexity of computing the covering radius of a code, *IEEE Trans. Inform. Th.*, vol. 30, pp. 800–804, 1984.

[13] A. MERTZ: On the complexity of multicovering radii, *IEEE Trans. Inform. Th.*, vol. 50, pp. 1804–1808, 2004.

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