Master of Science Thesis

# Identifying Codes in Special Graph Classes

Florent Foucaud

*Supervisors:*
Ralf Klasing, André Raspaud

June 2009

## Acknowledgements

I wish to thank my supervisors Ralf Klasing and André Raspaud for the time they put into a quality supervising. With their help, this first step into the world of research has been passionating. My gratefulness also goes to Adrian Kosowski forthe nice time spent on working together. I also thank the members of the IDEA project (Identifying coDes in Evolving grAphs) for their comments and patience towards my presentations. Thanks as well to Olivier Baudon for his advice and the time spent reading my reports. Finally, my gratefulness goes to all the people, family or friends, who supported my during these years of studies, of which this thesis constitutes the conclusion - or, maybe, the beginning.

## Abstract

Identifying codes in graphs are related to the classical notion of dominating sets and its locating variant [SR84]. They have a property which allows unique identification of all vertices of the graph. Identifying codes were first introduced in 1998 [KCL98], and have since been studied widely in the communities of both graph theory and coding theory.
In this thesis, we will first give some definitions and summarize the most important known results on identifying codes, before considering identifying codes in some specific graph classes. In particular, we discuss the relationship between the maximum degree of a graph and the lower and upper bounds for the minimum cardinality of an identifying code in this graph.

# Contents

# Chapter 1

# Introduction to identifying codes

In this chapter, we give some definitions and summarize the most important known results on the topic of identifying codes.

## 1.1 Graphs

We first recall some standard notions and definitions of graph theory which will be used in this thesis. For a complete introduction to graph theory, see [Ber58].

An *undirected graph* $G = (V, E)$ consists of a pair of sets. $V(G)$ is called the *vertex set* of $G$ and $E(G)$, the *edge set* of $G$, is a set of unordered pairs of vertices of $V(G)$. If $u$ and $v$ are two vertices of $G$, an edge between $u$ and $v$ is denoted by $\{u, v\}$ or $uv$.

A *directed* or *oriented graph* $G = (V, A)$ consists of a vertex set $V(G)$ and an *arc set* $A(G)$ of ordered pairs of vertices of $G$. If $u$ and $v$ are two vertices of $G$, an arc between $u$ and $v$ is denoted as $(u, v)$ or $\overrightarrow{uv}$.

The cardinality of $V(G)$ of a graph $G$, usually denoted by $n$, is called the *order* of $G$. A graph $G$ is *simple* if all edges of $E(G)$ are distinct ($G$ has no *multiple edges*), and if no edge contains the same vertex twice ($G$ has no *loops*). It is *finite* if $V(G)$ and $E(G)$ are both finite. Otherwise, it is *infinite*.

Typically, a graph is visualised as a set of points representing its vertices, which are linked by lines representing the edges or the arcs. If the graph is directed, the lines are in forms of arrows pointing from the first element towards the second element of the corresponding arc. Graphs are useful for representing elements and their connections. For example, social or physical networks are modelled with graphs. See the figures of this section for various examples of graphs (for example Figures 1.1, 1.6 or 1.8).

Two vertices $u$ and $v$ of a graph $G$ are called *neighbours* if they are connected by an edge in $G$. Let us denote by $deg_G(v)$ (or $deg(v)$ if there is no ambiguity) the *degree* of $v$ in $G$: the number of neighbours of $v$. The *maximum degree* of $G$, usually denoted by $\Delta(G)$, is the maximum degree of a vertex of $V$. Similarly, the *minimum degree* of $G$, usually denoted by $\delta(G)$, is the minimum degree of a vertex of $V$. A graph in which all vertices have the same degree $k$, is called *$k$-regular*. A graph having maximum degree 3 is called *subcubic*, and a 3-regular graph is called *cubic*.

A *path* of length $k$ in $G$ between two vertices $v_0$ and $v_k$ is a sequence $\{v_0, ..., v_k\}$ of $k+1$ vertices of $G$ such that for all $i \in \{0, ..., k-1\}$, the edge $\{v_i, v_{i+1}\} \in E(G)$.

The *distance* between two vertices $u$ and $v$ in a graph $G$, denoted as $d_G(u, v)$ (or simply $d(u, v)$ if there is no ambiguity), is the smallest length of a path between $u$ and $v$ in $G$. If such a path does not exist, then the distance between $u$ and $v$ is not defined.

Let $v$ be a vertex of $G$. We denote by $B_r(v)$ the *ball* of radius $r$ of $v$: all vertices of $V$ which are at distance less or equal than $r$ of $v$. Let us also denote by $N[v] = B_1(v)$, the *closed neighbourhood* of $v$, and by $N(v) = N[v]\backslash v$, the *neighbourhood* of $v$.

$G$ is said to be *connected* if for every pair of distinct vertices $u$ and $v$, there exists a path between $u$ and $v$. If $G$ is not connected, then it consists of the union of a finite number of connected graphs. Each of these graphs is called a *connected component* of $G$. A graph $G$ is called *$k$-connected* if the graph obtained after deletion of any $k-1$ vertices of $V(G)$ (and the edges involving these vertices), is still connected.

A *cycle* of length $k$ in a graph $G$ is a sequence $\{v_0, ..., v_{k-1}\}$ of $k$ distinct vertices of $G$ such that for all $i \in \{0, ..., k-1\}$, $\{v_i, v_{(i+1) \bmod k}\} \in E(G)$. A connected graph having no cycle is called a *tree*.

An *Hamiltonian cycle* of $G$ is a cycle of length $|V(G)|$ in $G$. A graph $G$ is called *Hamiltonian* if there exists an Hamiltonian cycle in $G$.

A graph $H = (V_H, E_H)$ is a *subgraph* of $G$ if $V_H \subseteq V$ and $E_H \subseteq E$. A *spanning subgraph* of $G$ is a subgraph $H$ of $G$ such that $V(H) = V(G)$. A spanning subgraph of $G$ is called a *spanning tree* of $G$ if it is a tree.

Let $k$ be an integer. A *$k$-factor* of a graph $G$ is a $k$-regular subgraph of $G$. A 1-factor (which is a set of independent edges) is called a *matching* of $G$. A matching $M$ of $G$ is called *perfect* if every vertex of $G$ is contained in an edge of $M$.

Let $S \subseteq V$ be a subset of vertices of $G$. The subgraph of $G$ *induced* by $S$, denoted by $G[S]$, is the graph $G[S] = (S, E_S)$, where $E_S$ is the set of edges of $E$ having both endpoints in $S$.

The *complete graph* of order $n$, denoted by $K_n$, is the graph in which all vertices are connected to each other by an edge.

In a graph $G = (V, E)$, an *induced triangle* or *triangle* is a set of three distinct vertices $u, v, w \in V$ such that $uv, uw, vw \in E$: the three vertices are connected to each other. In other words, the induced subgraph $G[\{u, v, w\}]$ is the complete graph $K_3$. A graph is called *triangle-free* if it does not contain any triangle.

The *girth* of a graph $G = (V, E)$, denoted by $g(G)$, is the smallest cardinality of a cycle in $G$, that is to say, the smallest sequence $v_0, ..., v_{g-1}$ of vertices of $V$ such that for all $i \in \{0, ..., g-1\}$, $v_i v_{(i+1) \bmod g} \in E$.

Let $G = (V, E)$ be a graph and $k \geq 1$. A set $S \subseteq V$ is a *k-independent set* if for all $x, y \in S$, $d(x, y) \geq k+1$. A 1-independent set is simply called an *independent set*.

A graph is said to be *bipartite* if there exists a partition of $V(G)$ into two nonempty, disjoint sets $A$ and $B$, such that every edge of $E(G)$ contains a vertex of $A$ and a vertex of $B$. In other words, $A$ and $B$ are two disjoint and nonempty independent sets of $G$.

A graph $G$ is called *planar* if it is possible to draw it on the Euclidian plane such that no edges interesect each other. If $G$ is drawn in such a way, a *face* of $G$ is a closed region of the plane delimited by a set of edges. A planar graph is *outerplanar* if it can be drawn such that all vertices are on the same face.

All graphs considered from now on are simple, connected, undirected and finite, unless otherwise stated.

## 1.2 Identifying codes in graphs

We now give some more specific definitions of the field of domination and identification in graphs. For a complete introduction to the topic of domination in graphs, see [HHS98]. For more details on identifying codes, consult for example [KCL98, Mon05a, Ska07, BCHL04, CHL07, BWLT06]. For a survey, see [TXXH06]. Note that an updated online bibliography on locating-dominating sets and identifying codes can be found in [Lob09].

Let $G = (V, E)$ be a graph. A vertex $v \in V$ *dominates* or *covers* a vertex $u$ if $u \in N[v]$ ($u$ is at distance at most 1 of $v$).
Let $C, C' \subseteq V$ be two sets, we say that $C$ dominates, or covers $C'$ if all vertices of $C'$ are dominated by a vertex of $C$.
A set $C \subseteq V$ is a *dominating set* or *covering code* of G if $C$ dominates $V$.

A pair $\{u, v\}$ of vertices of $V$ are *separated* by $x \in V$ if $x$ dominates exactly one of the vertices $u$ and $v$.
Let $C, C' \subseteq V$ be to sets, we say that $C$ separates $C'$ if for every pair $\{u, v\}$ of vertices of $C'$, $u$ and $v$ are separated by at least a vertex of $C$.

A set $C \subseteq V$ is a *locating-dominating set* or *locating-dominating code* of $G$ if:
1) $C$ is a dominating set of $G$, and
2) $C$ separates $V \backslash C$.
In other words, for every vertex $v \in V$, $N[v] \cap C \neq \emptyset$, and for every pair $\{u, v\}$ of vertices of $V \backslash C$, $N[u] \cap C \neq N[v] \cap C$.

A set $C \subseteq V$ is an *identifying code* (sometimes refered to as a *differentiating-dominating set*, as in [GVGN$^+$01] or [Ska07]) of $G$ if:
1) $C$ is a dominating set of $G$, and
2) $C$ separates $V$.
In other words, for every vertex $v \in V$, $N[v] \cap C \neq \emptyset$, and for every pair $\{u, v\}$ of vertices of $V$, $N[u] \cap C \neq N[v] \cap C$.

In an identifying code $C$ of $G$, for a vertex $v \in V$, the set $I_C(v) = N[v] \cap C$ is called the *identifying set* of $v$. One can give an alternative definition of an identifying code in the following way. Let $G = (V, E)$ be a graph; a set $C \subseteq V$ is an identifying code of $G$, if for every vertex $v \in V$, $I_C(v) \neq \emptyset$, and if for every pair of distinct vertices $u, v \in V$, $I_C(u) \neq I_C(v)$.

Let us denote by $\oplus$, the *symmetric difference* between two sets: $A \oplus B = A \backslash B \cup B \backslash A$. An alternative formulation of the separation condition (2) of the definition is the following:
2') for every pair $\{u, v\}$ of vertices of $V$, $I_C(u) \oplus I_C(v) \neq \emptyset$.

See Figures 1.1, 1.2 and 1.3 for examples of a dominating set, a locating-dominating set, and an identifying code. Vertices of these sets are colored in gray, and the identifying sets are in braces. Note that the dominating set is not a locating-dominating set since $a$ and $e$ are not separated. Similarly, the locating-dominating set is not an identifying code since $c$ and $f$ are not separated.

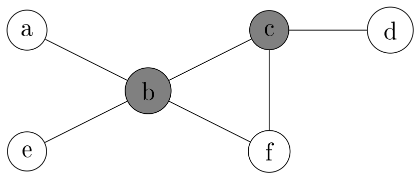


Figure 1.1: Example of a dominating set

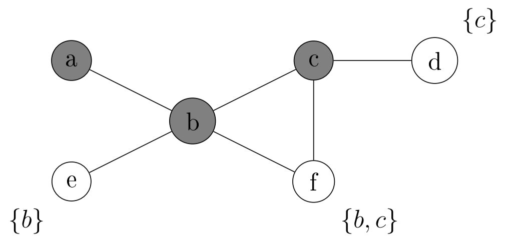


Figure 1.2: Example of a locating-dominating set

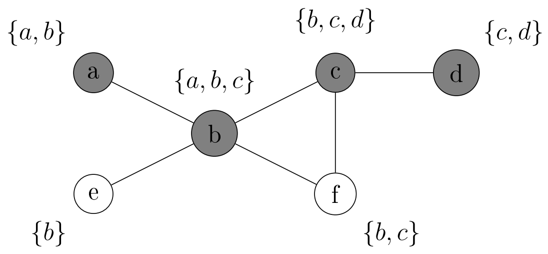


Figure 1.3: Example of an identifying code

## 1.3 Basic properties and remarks

Whereas all graphs admit a dominating set or a locating-dominating set (simply take $C = V$), not all graphs admit an identifying code. In particular, if there exist two vertices $u, v \in V$ such that $N[u] = N[v]$ ($u$ and $v$ are called *twins*), then they cannot be separated. Graphs which admit an identifying code are called *twin-free* or *identifiable*. The structure of twin-free graphs has been studied in [CHHL07]. As an easy counterexample, in the complete graph $K_n$ consisting of $n$ vertices all connected to each other, all vertices are twins, therefore $K_n$ does not admit an identifying code.

A natural question is, given a graph $G$, to determine the minimum cardinality of a dominating set, a locating-dominating set, or an identifying code of $G$, and to construct such an optimal sets. Let us denote by $\gamma(G)$, the minimum cardinality of a dominating set of $G$, by $\gamma_L(G)$, the minimum cardinality of a locating-dominating set of $G$, and by $M_1(G)$, the minimum cardinality of an identifying code of $G$. Note that any identifying code of $G$ is also a locating-dominating set of $G$, and any locating-dominating set of $G$ is a dominating set of $G$. From this relation of inclusion between these three types of sets, the following inequality follows : given a twin-free graph $G$, $\gamma(G) \leq \gamma_L(G) \leq M_1(G)$.

Despite the close relationship between dominating sets, locating-dominating sets and identifying codes, it is in general not possible to obtain easily an identifying code from a dominating set or even a locating-dominating set; this makes it necessary to study specifically those problems. There exist some graphs where the minimum cardinalities of these sets are very different. For example, in the star $K_{1,n-1}$ on $n$ vertices consisting of a central vertex connected to $n-1$ independent vertices, the optimal size of a dominating set is 1 (simply take the central vertex), whereas $n-1$ vertices are needed for both a locating-dominating set and an identifying code (see Figure 1.4). In the complete graph $K_n$, the optimal size of a dominating set is again 1, whereas $n-1$ vertices are needed for a locating-dominating set, and as previously mentioned, $K_n$ does not admit an identifying code at all (see Figure 1.5).

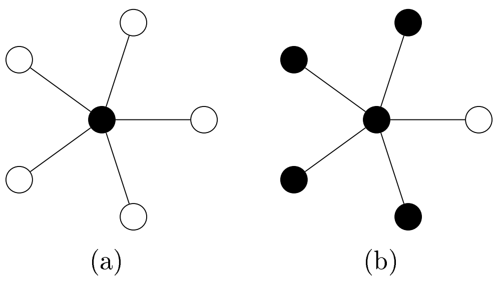


Figure 1.4: Examples of an optimal dominating set (a) and an optimal locating-dominating set (b) in the star $K_{1,5}$

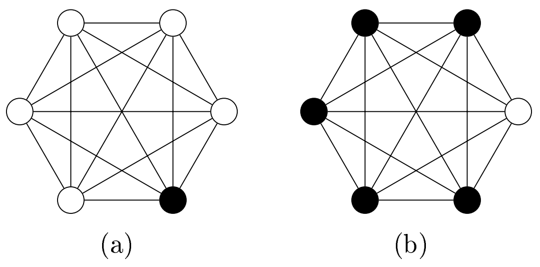


Figure 1.5: Examples of an optimal dominating set (a) and an optimal locating-dominating set (b) in the complete graph $K_6$

Note that if $G$ is not connected, then a minimum identifying code is the union of minimum identifying codes of all its connected components. Hence, it is sufficient to study identifying codes in connected graphs.

Let us now introduce identifying codes from another point of vue. Let $G$ be a graph on $n$ nodes with vertex set $\{v_0, ..., v_{n-1}\}$, and $C$ an identifying code of $G$ of cardinality $k$. Consider a matrix $M$ of $n$ columns and $k$ rows containing only zeros and ones: $M[i,j] = 1$ if and only if $v_i \in I_C(v_j)$. Then $C$ is an identifying code if all columns of $M$ are distinct and contain at least a 1. The problem of finding a minimum identifying code is equivalent to the one of finding such a matrix of $n$ columns and a minimum number of rows and such that if $v_i$ and $v_j$ are not connected in $G$, $M[i,j] = 0$.

## 1.4 Generalizations and variants

There exist several generalizations of identifying codes. We now present the most common ones.

Identifying codes can be generalized to infinite graphs. Let $G$ be an infinite, simple, twin-free graph. Then, instead of considering $M_1(G)$ (which has no sense in an infinite graph), one can consider the minimum *density* of an identifying code in $G$. Let us denote the density of an identifying code $C$ of $G$, as $d_C(G)$, and as $d^*(G)$, the minimum density of an identifying code of $G$. Intuitively, the density is the ratio between the number of code vertices and the total number of vertices within a local part of the graph. Let $v$ be an arbitrary vertex of $G$. Formally, $d_C(G) = \limsup_{n\to\infty} \dfrac{|C \cap B_n(v)|}{|B_n(v)|}$ [CHLZ01]. If $G$ is finite and has $n$ vertices, the density of an identifying code $C$ of $G$ is simply defined as $d_C(G) = \dfrac{|C|}{n}$.

The most important generalization of identifying codes is a distance-$r$ generalization, similar to the classical distance-$r$ generalization of dominating sets. Let $G$ be a graph and $r \geq 1$. An *$r$-dominating set* or *distance-$r$-dominating set* (also called *$r$-covering code* or *covering code of radius $r$*), is a subset $C$ of $V(G)$ such that for every vertex $v$ of $G$, $B_r(v) \cap C \neq \emptyset$.
An *$r$-locating-dominating set* is a subset $C$ of $V(G)$ such that $C$ is an $r$-dominating set and for every vertices $u, v$ of $V \setminus C$, the sets $B_r(u) \cap C$ and $B_r(v) \cap C$ are distinct.
An *$r$-identifying code* of $G$ [KCL98] is a subset $C$ of $V(G)$ such that $C$ is an $r$-dominating set of $G$, and every pair of vertices is separated by $C$ at distance $r$. In other words:
1) for every vertex $v \in V$, $B_r(v) \cap C \neq \emptyset$, and
2) for every pair $\{u, v\}$ of vertices of $V$, $B_r(u) \cap C \neq B_r(v) \cap C$.
Let us denote by $M_r(G)$, the minimum cardinality of an $r$-identifying code of $G$.

Another generalization is the identification of sets of vertices [KCL98]. If $C$ and $X$ are two subsets of vertices of a graph $G$, the *identifying set* of $X$, denoted as $I_C(X)$, is defined as the union of the identifying sets $I_C(x)$ of all the vertices $x$ of $X$.
Let $C, X, Y$ be three subsets of vertices of $G$. We say that $C$ separates $X$ and $Y$ if $I_C(X) \neq I_C(Y)$. An *$(\leq l)$-identifying code* is a subset $C$ of vertices of $G$ which dominates $V(G)$ and separates every two subsets of vertices having cardinality at most $l$.

Note that the two previous generalizations can be combined to consider *$(r, \leq l)$-identifying codes*.

A definition of identifying codes can be given for directed graphs. Let $G = (V, A)$

be a directed graph. Then, simply replace in the definition, the ball of a vertex $v$, $B_1(v)$, by its *incoming ball* $B_1^-(v) = \{x \in V \mid \overrightarrow{xv} \in A\}$.

Additionally, a generalization of identifying codes with weights and costs on vertices called *d-identifying codes* is given in [TXXH06]. A weighted and directed generalization called *source identification* is given in [BWLT06].

However, in this thesis we will focus on the classical case where the graph is finite, undirected, $r = 1$ and $l = 1$.

## 1.5 Applications

Identifying codes have a wide range of applications; we briefly present the most important ones.

A first application for fault diagnosis in multiprocessor systems is given in [KCL98]. Consider a network of processors modelled as an undirected graph. Some of the processors (which are part of the identifying code) are able to test wether a neighbouring processor (or themselves) are faulty; if this is the case, they enter a special state. If some of the testing processors have detected an error, it is possible to identify exactly the faulty processor: it is the one whose identifying set is equal to the set of processors which have detected the error. In the case of networks of thousands of processors, it is extremely useful to minimize the set of testing processors.

A second application is given in [RUDP$^+$03] and extends an application of locating-dominating sets given in [CSS87, Sla87]. Consider a facility or a building modelled by a graph: vertices represent areas or rooms, and edges indicate connexions between the areas. A sensor is placed in some rooms; the sensors are able to detect a danger or a threat, for example a fire, within their closed neighbourhood. The goal is to minimize the number of sensors needed in order to locate the position of an eventual threat. If the sensors are three-state sensors able to distinguish between no danger, a danger in a nearby room, and a danger in the same room, then a locating-dominating set is enough to uniquely determine the position of a danger. If the sensors can only distinguish between a danger in their closed neighbourhood, and no danger, then an identifying code is needed in order to achieve location. The danger is located by checking the set of sensors which are in alarm state; since this set is unique for every vertex, the location of the danger is immediate.

## 1.6 Identifying codes in specific graph classes

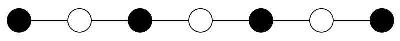
The problem of constructing a minimum identifying code has been studied in some specific classes of graphs.

The *path* on $n$ vertices, denoted by $P_n$, has $\{v_0, ..., v_{n-1}\}$ as its vertex set and $\big\{\{v_i, v_{i+1}\}, i \in \{0, ..., n-2\}\big\}$ as its edge set. Note that $P_2$ is the complete graph $K_2$ and does therefore not admit any identifying code. The cardinality of a minimum identifying code in paths is given in the following theorem:

**Theorem 1.1 ([GVGN$^+$01, BCHL04])**

- *Let $n \geq 1$ be odd. Then $M_1(P_n) = \dfrac{n+1}{2}$ and an optimal code is $\big\{v_i \mid i \text{ is even}\big\}$.*
- *Let $n \geq 4$ be even. Then $M_1(P_n) = \dfrac{n}{2} + 1$ and an optimal code is $\big\{v_i \mid i \text{ is even}\big\} \cup \{v_{n-3}\}$.*

Examples of optimal codes in $P_7$ and $P_8$ are given in Figure 1.6 (code vertices are black).


(a) $P_7$

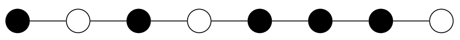

(b) $P_8$

Figure 1.6: Examples of optimal codes in the paths $P_7$ and $P_8$

The *infinite path* $P_\infty$ is the infinite graph of vertex set $\mathbb{Z}$ and edge set $\big\{\{i, i+1\}, i \in \mathbb{Z}\big\}$. The following theorem holds:

**Theorem 1.2 (BCHL04)**
*$d^*(P_\infty) = \dfrac{1}{2}$ and an optimal identifying code of $P_\infty$ is $\{i \in \mathbb{Z} \mid i \text{ is even}\}$.*

The *cycle* on $n$ vertices, denoted by $C_n$, has $\{v_0, ..., v_{n-1}\}$ as its vertex set and $\big\{\{v_i, v_{(i+1) \bmod n}\}, i \in \{0, ..., n-1\}\big\}$ as its edge set. Note that $C_3$ is the complete graph $K_3$ and does therefore not admit any identifying code. The cardinality of a minimum identifying code in cycles is given in the following theorem:

**Theorem 1.3 ([GVGN$^+$01, BCHL04, GMS06])**

- *$M_1(C_4) = 3$ and an optimal code is $\{v_0, v_1, v_2\}$*

- *$M_1(C_5) = 3$ and an optimal code is $\{v_0, v_1, v_2\}$*
- *Let $n \geq 6$ be even. Then $M_1(C_n) = \dfrac{n}{2}$ and an optimal code is $\{v_i \mid i \text{ is even}\}$.*
- *Let $n \geq 7$ be odd. Then $M_1(C_n) = \dfrac{n+1}{2} + 1$ and an optimal code is $\{v_i \mid i \text{ is even}\} \cup \{v_{n-2}\}$.*

Examples of optimal codes in $C_8$ and $C_9$ are given in Figure 1.7 (the black vertices are in the code).

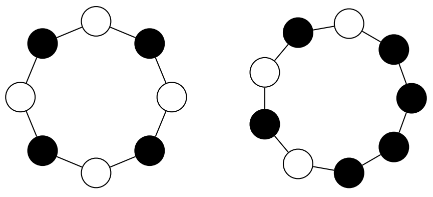

(a) $C_8$ (b) $C_9$

Figure 1.7: Examples of optimal codes in the cycles $C_8$ and $C_9$

$M_r(C_n)$ has been determined as well for some values of $r$ and $n$. The case where $n$ is odd and $2r + 5 \leq n \leq 3r + 2$, however, remains open. See [TXXH06] for a survey.

Identifying codes of trees have been studied in [KCL98, BCHL05, BCM$^+$07, CGH$^+$06, GVGN$^+$01]. In particular, a linear algorithm for constructing a minimum identifying code of a directed tree is given in [CGH$^+$06]. This algorithm is inspired from a linear algorithm for constructing a minimum locating-dominating set of an abitrary tree given in [Sla87]. A general lower bound for the minimum cardinality of an identifying code in a tree is the following:

**Theorem 1.4 ([BCHL05])** *Let $T$ be a tree on $n$ vertices. Then $M_1(T) \geq \dfrac{3(n+1)}{7}$.*

Let $T_k^h$ be the *complete $k$-ary tree* with $h$ levels and $n = \dfrac{k^h - 1}{k - 1}$ vertices, in which all internal vertices have $k$ sons. Exact values of the minimum cardinality of an identifying code in the complete tree are stated in the following theorem:

**Theorem 1.5 ([BCHL05])**

- $M_1(T_2^h) = \left\lceil \dfrac{20 \cdot n}{31} \right\rceil$

- *Let* $k \geq 3$. $M_1(T_k^h) = \left\lceil \dfrac{k^2 \cdot n}{k^2 + k + 1} \right\rceil$

Let us number the levels of $T_k^h$ from 1 to $h$, level 1 being the level of the root, and level $h$ being the level of the leaves. For $k \geq 3$, a construction of an optimal identifying code of $T_k^h$ is given in [BCHL05]:

- if $h = 1 \bmod 3$, the root is in the code.
- for all levels $i > 1$ such that $i = h \bmod 3$, each vertex on level $i-1$ has exactly $k-1$ of its $k$ sons in the code.
- in all levels $i$ such that $i = h - 1 \bmod 3$, each vertex is in the code.
- in all levels $i$ such that $i = h - 2 \bmod 3$, no vertex is in the code.

An example of an optimal codes in the complete tree $T_3^4$ is given in Figure 1.8. Code vertices are in black.

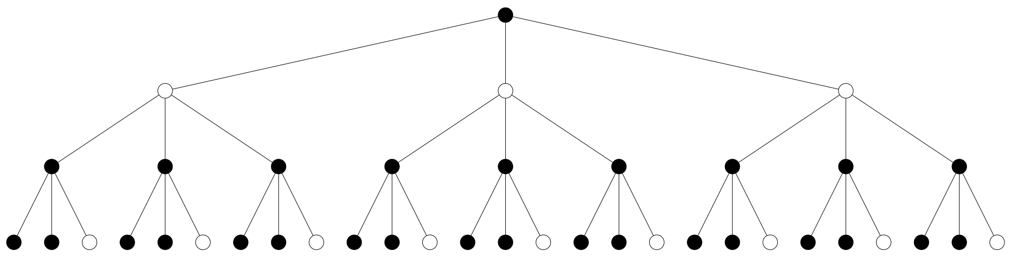


Figure 1.8: Example of an optimal code in the complete tree $T_3^4$

It is also shown in [KCL98] that there is no identifying code at distance $r > 1$ in the complete tree.

The case of the *binary hypercube* $H_m$ of dimension $m$ is of particular interest in code theory and has been studied in [KCL98]. Let us first define the vertex and edge sets of $H_m$:

$$\begin{aligned} V(H_m) = &\ \{0,1\}^m \\ E(H_m) = &\ \Big\{\big\{\alpha, \alpha(i)\big\}\Big| i \in \{0, ..., m-1\}, \alpha \in \{0,1\}^m\Big\} \end{aligned}$$

where for $\alpha = \alpha_0\alpha_1...\alpha_{m-1} \in \{0,1\}^m, i \in \{0,...,m-1\}, \alpha(i) = \alpha_0\alpha_1...\overline{\alpha_i}...\alpha_{m-1}$ (the *ith* bit of $\alpha$ is switched).

For an example, the hypercube of dimension 3, $H_3$, is represented in Figure 1.9.

The exact cardinality of $M_1(H_m)$ has not been determined yet, but the following lower and upper bounds hold:

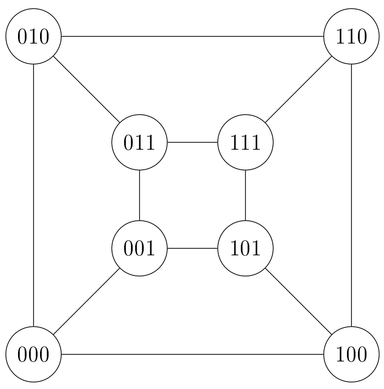


Figure 1.9: The hypercube $H_3$

**Theorem 1.6 ([KCL98])** *Let $m \geq 3$ and $H_m$ be the hypercube of dimension $m$. Then $M_1(H_m) \geq \dfrac{m \cdot 2^{m+1}}{m \cdot (m+1) + 2}$.*

**Theorem 1.7 ([KCL98])** *Let $m \geq 3$ and $H_m$ be the hypercube of dimension $m$. Let $C^*$ be an optimal covering code of radius 2 of $H_m$, of cardinality $K(m, 2)$. Then the code $C = \{u \in V(H_m) \mid \exists\, v \in C^*, d(u, v) = 1\}$ is a valid identifying code of $H_m$. Therefore $M_1(H_m) \leq m \cdot K(m, 2)$.*

Note that known values of $K(m, 2)$ for small $m$ as well as bounds on $K(m, 2)$ can be found in [Lit09, CHLL97].

The monotonicity of the minimum cardinality of an identifying code in the hypercube has been shown in [Mon06a]: for all $m \geq 2$, $M_1(H_m) \leq M_1(H_{m+1})$.

The *infinite grid* $Gr_{\infty,\infty}$ is the infinite graph of vertex set $\mathbb{Z} \times \mathbb{Z}$ and edge set $\big\{\{(i, j), (i', j')\} \mid i, j, i', j' \in \mathbb{Z} \text{ and } |i - i'| + |j - j'| = 1\big\}$. The following theorem holds:

**Theorem 1.8 ([CGH$^+$99, BHL05])**
*$d^*(Gr_{\infty,\infty}) = \dfrac{7}{20}$.*

However, the case of the finite grid $Gr_{n,m}$ remains open.

Some infinite meshes like the hexagonal and the triangular mesh have also been studied in [KCL98, Mon05a, CHLZ01].

## 1.7 Structural results

There exists a general lower bound for the cardinality of an identifying code in a twin-free graph of $n$ vertices. Indeed, consider a graph having an identifying code of size $k$. One can see the identifying sets of the vertices as binary words of size $k$. The number of distinct, nonempty words which can be coded with $k$ bits is exactly $2^k - 1$. This leads to the following bound:

**Theorem 1.9 ([KCL98])** *Let $G$ be a twin-free graph on $n$ vertices. Then $M_1(G) \geq \lceil \log_2(n+1) \rceil$.*

The previous bound is tight and all graphs reaching it have been characterized in [Mon05a, Mon05b, Mon06b]. Let $n \geq 1$ be an integer and $k = \lceil \log_2(n+1) \rceil$. The following construction given in [Mon06b] allows us to construct all graphs on $n$ vertices having an identifying code of cardinality $k$:

1. Let $H$ be a twin-free graph on $k$ vertices $\{c_0, ..., c_{k-1}\}$. Put all these vertices into the code.
2. Consider all characteristic vectors of the identifying sets of the code vertices: for a vertex $c_i$, the vector $w(c_i) = (w_0, ..., w_{k-1})$ where for all $j \in \{0, ..., k-1\}$, $w_j = 1$ if and only if $c_j \in N[c_i]$.
3. Take $H$ and $n - k$ other vertices. For each of these vertices, choose a distinct characteristic vector, and connect them to the code vertices using this information
4. Eventually add edges between the non-code vertices (this has no influence on the code).

An example for $n = 7$ having an optimal identifying code of cardinality $\lceil \log_2(n + 1) \rceil = 3$) is given in Figure 1.10. Code vertices are in gray, and the characteristic vectors are in brackets.

A lower bound for the minimum cardinality of a locating-dominating set of planar and outerplanar graphs is given in the following theorem:

**Theorem 1.10 ([SR84])** *Let $G$ be a graph on $n$ vertices and $L$ a locating-dominating set of $G$ of cardinality $k$. The following bounds are tight:*

- *If $G$ is planar and $k \geq 4$, $n \leq 7k - 10$.*
- *If $G$ is outerplanar, $n \leq \left\lfloor \frac{7k-3}{2} \right\rfloor$.*

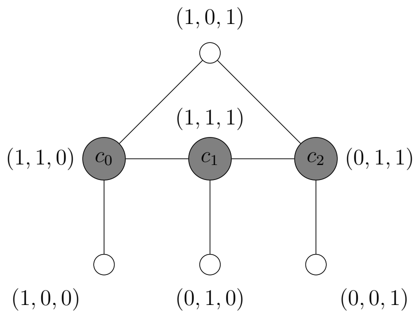


Figure 1.10: Example of an optimal graph

Since any identifying code is a locating-dominating set, the previous lower bounds also hold for identifying codes; we reformulate the bounds as lower bounds for $M_1(G)$:

**Corollary 1.11** *Let $G$ be a twin-free graph on $n$ vertices.*

- *If $G$ is planar and $M_1(G) \geq 4$, $M_1(G) \geq \dfrac{n+10}{7}$.*
- *If $G$ is outerplanar, $M_1(G) \geq \dfrac{2n+3}{7}$.*

There exists a general upper bound for the minimum cardinality of an identifying code:

**Theorem 1.12 ([CHL07, GM07])** *Let $G$ be a finite, twin-free connected graph on $n$ vertices.*
*Then $M_1(G) \leq n-1$.*

This bound is tight, and there are infinitely many graphs reaching it [CHL07, GM07]. However, the set of these graphs has not been characterized yet. For a given $n$, three graphs reaching this upper bound are described in [CHL07]. The first one is the *star* $K_{1,n-1}$ consisting of a central vertex connected to $n-1$ independent vertices. For $n > 3$, it is easy to see that any set of $n-1$ vertices of $K_{1,n-1}$ can be taken as a code. The second graph can be constructed for even $n$. Consider the complete graph $K_n$, and remove a perfect matching ($\frac{n}{2}$ independent edges) from $E(K_n)$. Let us denote this graph by $G_n^2$. Again, any set of $n-1$ vertices is an identifying code of $G_n^2$. Finally, for $n$ odd, consider the graph $G_{n-1}^2$, plus an extra vertex $v$ which is connected to all other vertices. Let us denote this graph by $G_n^3$. In $G_n^3$, all vertices but $v$ must be in the code; on the other hand, this is enough.

Examples of these graphs are presented in Figure 1.11 with minimum identifying codes (in black).

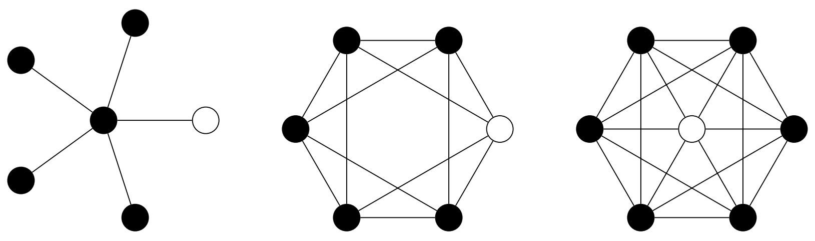

(a) The graph $K_{1,5}$ (b) The graph $G_6^2$ (c) The graph $G_7^3$

Figure 1.11: Examples of graphs having a minimum identifying code of cardinality $n-1$

It has also been shown that all possible values of $M_1(G)$ lying between the bounds of Theorems 1.9 and 1.12 are reached for some graphs [CHL05].

## 1.8 Hardness results

We now present known hardness results for decision and optimizations problems related to identifying codes.

A *decision problem* is defined by an input, or *instance*, and a question which can only be answered by "yes" or "no". The *complexity class* of decision problems which can be solved by an algorithm in polynomial time regarding the size of the instance is called $P$. Let $A$ be a decision problem. A *certificate* $C$ of $A$ is a piece of information which can be associated to every instance $I_A$ of $A$, such that the size of $C$ is polynomial in the size of $I_A$, and it is possible to use $C$ to check what is the answer to the question of $A$ applied to $I_A$ in polynomial time in the size of $I_A$ and $C$. The class of problems having such a certificate is called $NP$. It is known that $P \subseteq NP$.

However, there exist a variety of problems of $NP$, called *$NP$-complete* problems, which are hard to solve: for each of these problems, there exists no polynomial time algorithm for solving it, unless $P = NP$. To prove that a problem $A$ is $NP$-complete, it is possible to give a polynomial-time procedure $f$, called *polynomial reduction*, which transforms any instance $I_B$ of an $NP$-complete problem $B$ into an instance $I_A = f(I_B)$ of $A$, such that the answer to the question of $B$ applied to $I_B$ is positive if and only if the answer to the question of $A$ is positive when applied to $I_A$.

For further details on these questions, see [GJ79].

Let us state the following decision problems:

DOMINATING SET
INSTANCE: A graph $G$ and a positive integer $k$
QUESTION: Does $G$ have a dominating set of cardinality $\leq k$?

LOCATING-DOMINATING SET
INSTANCE: A graph $G$ and a positive integer $k$
QUESTION: Does $G$ have a locating-dominating set of cardinality $\leq k$?

IDENTIFYING CODE
INSTANCE: A graph $G$ and a positive integer $k$
QUESTION: Does $G$ have an identifying code of cardinality $\leq k$?

$r$-LOCATING-DOMINATING SET
INSTANCE: A graph $G$ and a positive integer $k$
QUESTION: Does $G$ have an $r$-locating-dominating set of cardinality $\leq k$?

$r$-IDENTIFYING CODE
INSTANCE: A graph $G$ and a positive integer $k$
QUESTION: Does $G$ have an $r$-identifying code of cardinality $\leq k$?

DIRECTED $r$-LOCATING-DOMINATING SET
INSTANCE: A directed graph $G$ and a positive integer $k$
QUESTION: Does $G$ have an $r$-locating-dominating set of cardinality $\leq k$?

DIRECTED $r$-IDENTIFYING CODE
INSTANCE: A directed graph $G$ and a positive integer $k$
QUESTION: Does $G$ have an $r$-identifying code of cardinality $\leq k$?

We summarize the known hardness results on these problems in the following theorem:

**Theorem 1.13**

- **([GJ79, KKY80, Ber84])** *DOMINATING SET is NP-complete* [GJ79]*, even when restricted to cubic planar graphs* [KKY80] *or bipartite graphs* [Ber84].
- **([CSS87])** *LOCATING-DOMINATING SET is NP-complete.*
- **([CHLZ01, MS06])** *IDENTIFYING CODE is NP-complete* [CHLZ01]*, even when restricted to planar bipartite unit disk graphs*[1] [MS06].
- **([CHL03])** *$r$-LOCATING-DOMINATING SET and $r$-IDENTIFYING CODE are NP-complete, even when restricted to bipartite graphs.*

[1]A *unit disk graph* is a graph for which each vertex is associated to a point of the Euclidian plane $\mathbb{R}^2$. There exists an edge between two vertices $u$ and $v$ if and only if the euclidian distance between the points associated to $u$ and $v$ is at most 1.

- **([CHL02])** *DIRECTED r-LOCATING-DOMINATING SET and DIRECTED r-IDENTIFYING CODE are NP-complete, even when restricted to strongly connected, asymmetric, bipartite directed graphs or to asymmetric, bipartite directed graphs without directed cycles.*

An *optimization problem* is described by an input and an output which is optimum with respect to some measure function $m$. The optimum can either be a minimum (then the problem is a *minimization problem*) or a maximum (then it is a *maximization problem*).

By analogy with the classes $P$ and $NP$ of decision problems, the classes $PO$ and $NPO$ are defined for optimization problems. An optimization problem is in $PO$ if there exists an algorithm which gives an optimal solution to the problem in polynomial time with respect to the size of the input. An optimization problem $A$ is in $NPO$ if, given an instance $I_A$ of $A$ and a potential solution $S$ of $A$, it is possible to decide in polynomial time with respect to $I_A$ and $S$, if $S$ is a feasible (not necessarily optimum) solution of $A$, and if the measure of the output is computable in polynomial time.

Let $r$ be a real number, $A$ an optimization problem, $I_A$ an instance of $A$ and $m_{opt}$ the value of the measure of the optimum solution. An *$r$-approximation* of $I_A$ for $A$ is a solution for $A$ with input $I_A$ which measure is at most $r \cdot m_{opt}$ for a minimization problem, and at least $m_{opt}/r$ for a maximization problem. An algorithm which, given any input $I_A$ of $A$, is able to compute an $r$-approximation of $I_A$ in polynomial time with respect to $|I_A|$ is called a polynomial time *$r$-approximation algorithm* for $A$.

The class sometimes refered to as log-$APX$ is the set of optimization problems for which there exists a function $f(n) = O(\ln(n))$ and a polynomial time approximation algorithm which computes an $f(|I|)$-approximation of $I$ for the problem, where $I$ is the input. log-$APX \subseteq NPO$.

The class $APX$ contains all optimization problems for which there exists a real number $\alpha > 0$ and a polynomial time approximation algorithm which computes an $\alpha$-approximation of $I$ for the problem, where $I$ is the input. $APX \subseteq$ log-$APX$.

The class $PTAS$ is the class of optimization problems for which there exists an algorithm which, given an input $I$ of the problem together with a rational value $r > 1$, computes an $r$-approximation of $I$ for the problem in time polynomial in $|I|$. Such an alogorithm is called a *polynomial time approximation scheme*. $PTAS \subseteq APX$.

A decision problem can always be associated to an optimization problem. An optimization problem is said to be *$NP$-hard* if the corresponding decision problem is $NP$-complete. It is *$APX$-hard* if it is not in the class $PTAS$, and it is $APX$-

*complete* if it is $APX$-hard and in $APX$. Similarly, it is log-$APX$*-hard* if it is not in $APX$, and log-$APX$*-complete* if it is log-$APX$-hard and in log-$APX$.

For more details on hardness and approximation, consult [ACG+99] or [Hro02].

Let us state the following optimization problems:

MINIMUM DOMINATING SET
INPUT: A graph $G$
OUTPUT: The minimum cardinality of a dominating set of $G$

MINIMUM LOCATING-DOMINATING SET
INPUT: A graph $G$
OUTPUT: The minimum cardinality of a locating-dominating set of $G$

MINIMUM IDENTIFYING CODE
INPUT: A graph $G$
OUTPUT: The minimum cardinality of an identifying code of $G$

MINIMUM $r$-LOCATING-DOMINATING SET
INPUT: A graph $G$
OUTPUT: The minimum cardinality of an $r$-locating-dominating set of $G$

MINIMUM $r$-IDENTIFYING CODE
INPUT: A graph $G$
OUTPUT: The minimum cardinality of an $r$-identifying code of $G$

We summarize the known hardness results on these problems in the following theorem:

**Theorem 1.14**

- **([Joh74, LY94])** *MINIMUM DOMINATING SET is* log-$APX$*-complete, even when restricted to bipartite graphs*
- **([PY91])** *MINIMUM DOMINATING SET is $APX$-complete when restricted to graphs having their degree bounded by a constant*
- **([Bak94, HIMR+98])** *MINIMUM DOMINATING SET is in PTAS when restricted to planar graphs* [Bak94] *or unit disk graphs* [HIMR+98]
- **([BWLT06, Suo07])** *MINIMUM IDENTIFYING CODE and MINIMUM LOCATING-DOMINATING SET are* log-$APX$*-complete*
- **([Suo07, GKM08])** *MINIMUM IDENTIFYING CODE and MINIMUM LOCATING-DOMINATING SET are $APX$-complete when restricted to graphs having their degree bounded by a constant*

- **([Suo07])** *MINIMUM $r$-IDENTIFYING CODE and MINIMUM $r$-LOCATING-DOMINATING SET are in PTAS when restricted to local graphs*[2]

Note that in [RUDP$^+$03], a greedy linear-time algorithm is provided to compute a minimal identifying code (a *minimal* or *irreducible* identifying code $C$ is a code such that there exists no vertex $c$ of $C$ such that $C\setminus\{c\}$ is an identifying code, whereas a *minimum* identifying code is an identifying code of minimum cardinality). However, it has been shown in [Mon06b] that the gap between a minimal and a minimum identifying code can be arbitrarily large. Therefore the mentioned algorithm does not guarantee any approximation ratio.

[2]A $(d, N)$-*local graph* is a graph $G$ for which each vertex is associated to a point in $\mathbb{R}^d$ so that within any ball $B$ of radius 1 in $\mathbb{R}^d$, at most $N$ vertices of $G$ are associated to points inside of $B$. Moreover, two vertices $u$ and $v$ of $G$ can only be connected by an edge if the euclidian distance between their correspondant points is at most 1.

# Chapter 2

# A lower bound depending on the maximum degree

Natural questions regarding identifying codes are extremal problems: the determination of lower and upper bounds for $M_1(G)$, and the characterization of graphs reaching these bounds.

Whereas most of these questions have already been solved in the general case (see the previous section for details), almost no results have been published on the relationship between this problem and the maximum degree of the graph. Though, it is an interesting question since many common graphs have a bounded maximum degree, for example grids, tori, cycles.

In this section, we first restate the known lower bound for an identifying code $M_1(G)$ in graphs of maximum degree $\Delta$, before discussing the characterization of graphs reaching this bound.

## 2.1 The lower bound

The following theorem has been stated in [KCL98] for $\Delta$-regular graphs, but it is also valid for graphs of maximum degree $\Delta$. We restate the theorem in the case of 1-identifying codes and present an alternative, more combinatorial proof.

**Theorem 2.1 ([KCL98])** *Let $G = (V, E)$ be an undirected, connected graph on $n$ vertices and maximum degree $\Delta$. Then $M_1(G) \geq \dfrac{2n}{\Delta + 2}$.*

**Proof** Let $C \subseteq V$ be an identifying code of $G$ of cardinality $k$. We partition $V$ into the four following sets, as shown in Figure 2.1:
$C_1 = \Big\{x \in C \mid |I_C(x)| = 1\Big\}$, $C_2 = \Big\{x \in C \mid |I_C(x)| \geq 2\Big\}$
$N_1 = \Big\{v \in V \setminus C \mid |I_C(v)| = 1\Big\}$, $N_2 = \Big\{v \in V \setminus C \mid |I_C(v)| \geq 2\Big\}$

It is easy to see that the following (in)equalities hold:
$|C_1| + |C_2| = k$ (1)
$|C_1| + |N_1| \leq k$ (2)

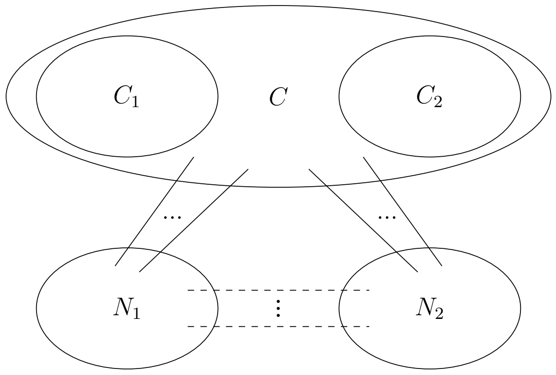


Figure 2.1: Partition of $V$

Let $m$ be the number of edges between $C$ and $V \setminus C = N_1 \cup N_2$. Let us make the following observations:

- Every vertex of $C_2$ is linked to at least one other vertex of $C_2$. Moreover, it is not possible to have two vertices of $C_2$ linked only to each other; indeed, this would imply that they have the same identifying set and they would not be separated. Thus, at least one of them must be linked to another vertex of $C_2$, and thus there are at least $\frac{2|C_2|}{3}$ edges within $C_2$. So, $m \leq k \cdot \Delta - 2 \cdot \frac{2|C_2|}{3} = k \cdot \Delta - \frac{4|C_2|}{3}$ since the maximum degree of $G$ is $\Delta$.

- There are no edges from any vertex of $C_1$ to another vertex of $C$.

- There are at least $2|N_2|$ edges between $C$ and $N_2$ since every vertex of $N_2$ has at least two elements of $C$ as its codeword, and there are exactly $|N_1|$ edges between $C$ and $N_1$.

Thus, we have $m \geq |N_1| + 2|N_2|$.

This gives us $k \cdot \Delta - \frac{4|C_2|}{3} \geq |N_1| + 2|N_2|$

$$k \cdot \Delta \geq \frac{4|C_2|}{3} + |N_1| + 2|N_2|$$

Since by (1), $|C_2| = k - |C_1|$, we get

$$k \cdot \Delta \geq k - |C_1| + |N_1| + 2|N_2| + \frac{|C_2|}{3}$$

By (2): $-|C_1| \geq |N_1| - k$, thus:

$$k \cdot \Delta \geq 2 \cdot (|N_1| + |N_2|) + \frac{|C_2|}{3}$$

$$k \cdot \Delta \geq 2 \cdot (n - k) + \frac{|C_2|}{3}$$

$$k \cdot (\Delta + 2) \geq 2n + \frac{|C_2|}{3} \geq 2n \qquad (*)$$

$$k \geq \frac{2n}{\Delta + 2} \qquad \square$$

## 2.2 Graphs reaching the lower bound

The lower bound of Theorem 2.1 is tight. Graphs which reach it can be easily constructed: given any $\Delta$ and any $k$, one can construct a graph of $n = k \cdot \frac{\Delta + 2}{2}$ vertices having an identifying code of cardinality $k$.

To construct such a graph, take an independent set $C$ of $k$ vertices as the identifying code, and $k \cdot \frac{\Delta}{2}$ other vertices. Then, connect the vertices of $V \setminus C$ to those of $C$ such that no two vertices have the same neighbours in $C$, all vertices of $C$ have less than $\Delta$ neighbours, and all vertices of $V \setminus C$ have at least two neighbours in $C$. Eventually, edges between non-code vertices can be added. An example of such a graph is given for $k = 4$ and $\Delta = 3$ in Figure 2.2. Code vertices are black, and the dashed edges are optional edges between non-code vertices.

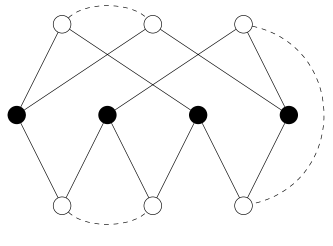


Figure 2.2: A graph reaching the lower bound of Thm. 2.1

Note that the general lower bound of Theorem 1.9 on $k$ ($k \geq \lceil \log_2(n+1) \rceil$ [KCL98]) still holds, so the following relationships must hold for these graphs:

$$k \cdot \frac{\Delta+2}{2} \leq 2^k - 1 \Leftrightarrow \Delta \leq \frac{2^{k+1}-2}{k} - 2 \text{ or } \Delta \leq \frac{2n}{\lceil \log_2(n+1) \rceil} - 2.$$

Let us now consider the case of graphs strictly reaching the lower bound, meaning that they admit an identifying code of cardinality exactly $k = \frac{2n}{\Delta+2}$ (without ceiling). We are able to characterize the structure of such graphs. Assume that $G$ has $n$ vertices, maximum degree $\Delta$ and an identifying code of cardinality $k = \frac{2n}{\Delta+2}$. Consider the partition of $V(G)$ illustrated in Figure 2.1, and let us recall line $(*)$ from the previous proof of Theorem 2.1. We had:

$$\begin{aligned} 2n + \frac{|C_2|}{2} &\leq k \cdot (\Delta+2) \\ \frac{2n}{\Delta+2} + \frac{|C_2|}{2(\Delta+2)} &\leq k = \frac{2n}{\Delta+2} \end{aligned}$$

So, if equality holds, clearly $|C_2| = 0$. This implies that $|C_1| = k$, and thus $|N_1| = 0$ since $|C_1| + |N_1| = k$. So a graph reaching the lower bound has an independent set as its code, and every non-code vertex is connected to at least two code vertices. Note that since $k = \frac{2n}{\Delta+2}$, $k \cdot \Delta = 2 \cdot (n-k)$, all non-code vertices are connected to exactly two code vertices, whereas all code vertices have $\Delta$ neighbours.

Let $\Delta$ be an integer. To construct a graph strictly reaching the lower bound, one can use the following construction method:

1. take any simple, $\Delta$-regular graph $D$ on $k$ vertices; these vertices will be the code vertices.
2. place an additional vertex on each edge of $D$: so, the new vertices are all connected to two different vertices of the regular graph $D$.
3. arbitrary edges can be set between the non-code vertices; this has no influence on the code.

It is clear that this construction gives a graph having exactly the properties described in the previous discussion.

Note that if one wants the constructed graph to be $\Delta$-regular, there must exist a $(\Delta-2)$-regular graph $H$ on $n-k$ vertices, and the non-code vertices must be embedded into $H$ in the last step of the construction. This is not always possible: for example, if the original graph $D$ has an odd number $n_e$ of edges, $H$ must have $n_e$ vertices, which is impossible since there does not exist any regular graph having an odd number of vertices. So in that case the constructed graph cannot be regular.

For an example of a 4-regular optimal graph constructed using the graph $D = K_5$ for the first step of the construction and the cycle $H = C_{10}$ in the last step, see Figure 2.3. An example of an optimal graph of maximum degree 3 which cannot be made regular, constructed using the *Petersen graph* $P_{10}$ having 10 vertices and 15 edges as the graph $D$, and no additional edges between non-code vertices, is given in Figure 2.4.

Conversely, all graphs strictly reaching the lower bound can be constructed using this method. Indeed, consider such a graph, $G$, and its optimal identifying code $C$. Remove all edges between non-code vertices. Following the previous discussion all non-code vertices are connected to exactly two code vertices and every code vertex has $\Delta$ neighbours. For every non-code vertex $v$, create an edge between its two neighbours in $C$, and delete $v$; the obtained graph is a $\Delta$-regular graph on $k$ vertices. Therefore the previous construction can be applied to obtain $G$. So, all such graphs can be obtained using this construction.

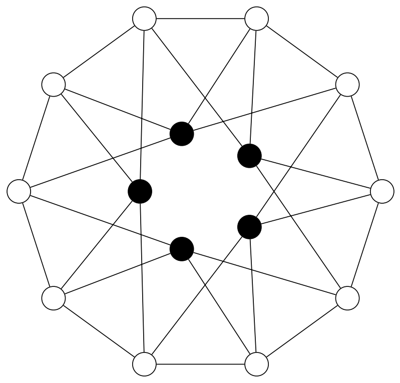


Figure 2.3: Example of a 4-regular optimal graph

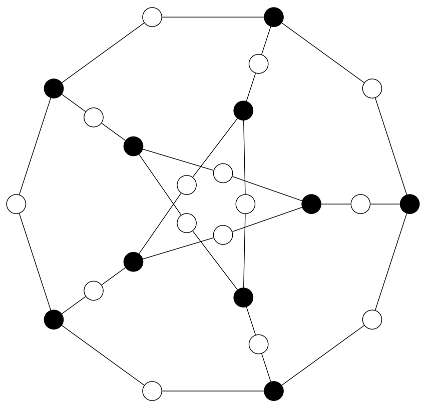


Figure 2.4: Example of an optimal graph of maximum degree 3

# Chapter 3

# Identifying codes in some particular graph families

We now investigate identifying codes of some specific graphs related to the binary hypercube. Identifying codes in the hypercube of dimension $m$, $H_m$, have been studied, but $M_1(H_m)$ is not known (see section 1.6). We study the problem in two families of graphs related to the hypercube: the *cube-connected cycles graph* and the *butterfly graph.*

## 3.1 Cube-Connected-Cycles graph

The *cube-connected-cycles graph* of dimension $m$, denoted as $CCC_m$, is a variant of the hypercube graph of dimension $m$. It has been introduced in 1981 for use in parallel computing [PV81]. For more details, consult [dR94] or [HKP+05].

$CCC_m$ is the hypercube $H_m$ in which all vertices are replaced by a cycle $C_m$:

$$\begin{aligned} V(CCC_m) = {} & \{0,...,m-1\} \times \{0,1\}^m \\ E(CCC_m) = {} & \Big\{ \big\{ (i,\alpha), ((i+1) \ mod \ m, \alpha) \big\} \Big| i \in \{0,...,m-1\}, \alpha \in \{0,1\}^m \Big\} \\ & \cup \Big\{ \big\{ (i,\alpha), (i,\alpha(i)) \big\} \Big| i \in \{0,...,m-1\}, \alpha \in \{0,1\}^m \Big\} \end{aligned}$$

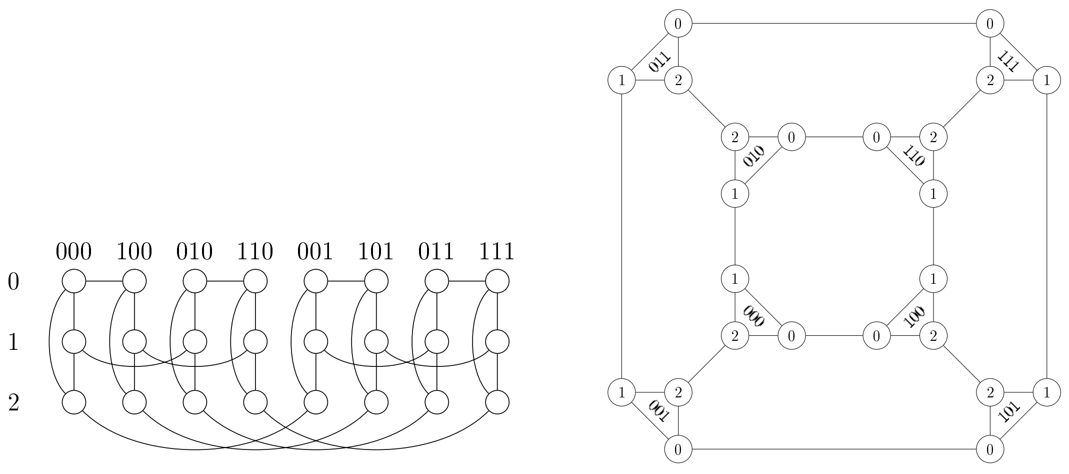
where for $\alpha = \alpha_0\alpha_1...\alpha_{m-1} \in \{0,1\}^m, i \in \{0,...,m-1\}, \alpha(i) = \alpha_0\alpha_1...\overline{\alpha_i}...\alpha_{m-1}$ (the *ith* bit of $\alpha$ is switched).

For some examples, see Figure 3.1 for the Cube Connected Cycles of dimension 3, and Figures 3.2 and 3.3 for the Cube Connected Cycles of dimension 4.

Since any $CCC_m$-graph is cubic, following Theorem 2.1, we have the following proposition:

**Proposition 3.1** *Let $m \in \mathbb{N}$ and $C$ be an identifying code of $CCC_m$. Then* $|C| \geq \dfrac{2|V(CCC_m)|}{5}$.

We approach this bound with the following result:



(a) (b)

Figure 3.1: Two representations of the graph $CCC_3$

**Theorem 3.2** *Let $m \geq 4$. Then $M_1(CCC_m) \leq m \cdot 2^{m-1} = \dfrac{|V(CCC_m)|}{2}$.*

**Proof** Let $C = \left\{ \left(i, \alpha\right) \in V(CCC_m) \middle| i \in \{0, ..., m-1\}, \alpha \in \{0,1\}^m, \sum_{i=0}^{m-1} \alpha_i = 0 \; mod \; 2 \right\}$.
We claim that $C$ is an identifying code of $CCC_m$. Indeed, $C$ is such that half of the cycles of $CCC_m$ are completely in $C$, and the cycles are at distance two from each other. Thus, every vertex of a cycle of $C$ is identified by itself and its two neighbours (since $m \geq 4$ the cycles $C_m$ are twin-free), whereas a vertex of the other cycles is uniquely identified by its neighbour on a cycle of $C$.
Thus both separation and domination are achieved and $C$ is an identifying code of size $\dfrac{|V(CCC_m)|}{2}$. □

An example of a code constructed in the proof of Theorem 3.2 for the graph $CCC_4$ is given in Figure 3.2. The code vertices are colored in gray, and arrows indicate wrap-around edges.

The ratio between the upper bound given by the previous theorem and the lower bound given by Proposition 3.1 is $\frac{5}{4}$. This upper bound is probably not tight; indeed, for the graphs $CCC_4$ and $CCC_5$, we found identifying codes of smaller cardinality. A better identifying code for $CCC_4$ is pictured in Figure 3.3 (the code vertices are colored in gray, and arrows indicate wrap-around edges). Since $CCC_4$ has $n_4 = 64$ vertices, any identifying code of $CCC_4$ has cardinality at least 26 according to Proposition 3.1. The found identifying code has cardinality $28 = \frac{7n_4}{16}$. In

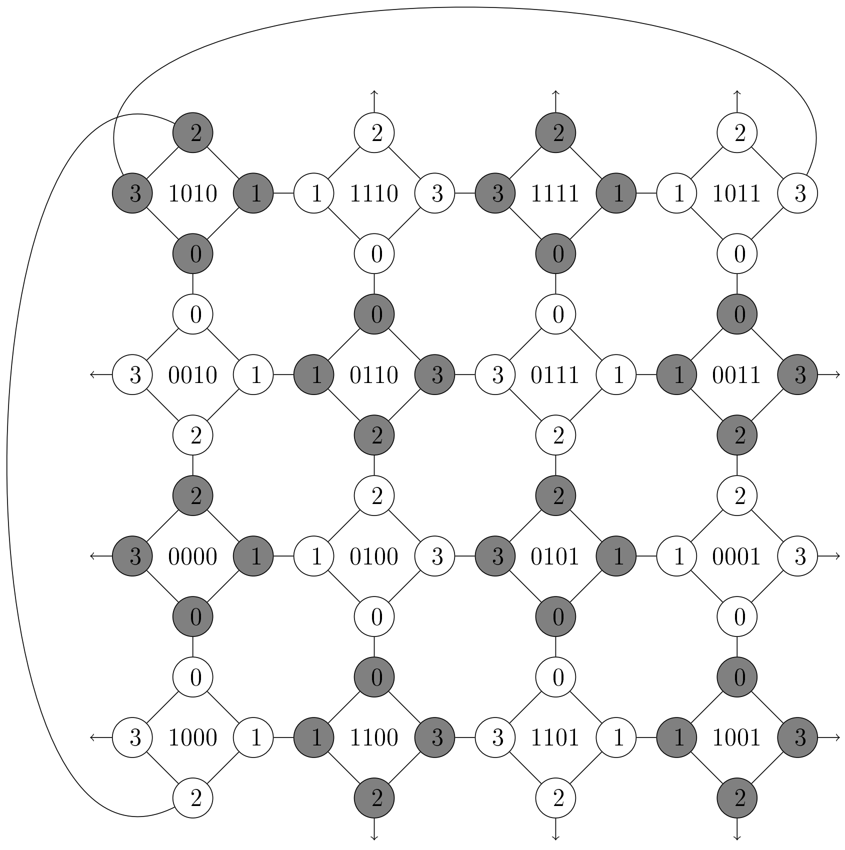


Figure 3.2: An identifying code of the graph $CCC_4$ containing half of its vertices

$CCC_5$, which has $n_5 = 160$ vertices, any identifying code has cardinality at least 64 according to Proposition 3.1. We found an identifying code of cardinality $72 = \frac{9n_5}{20}$. However, we were not able to generalize these codes to cube-connected cycles of any dimension.

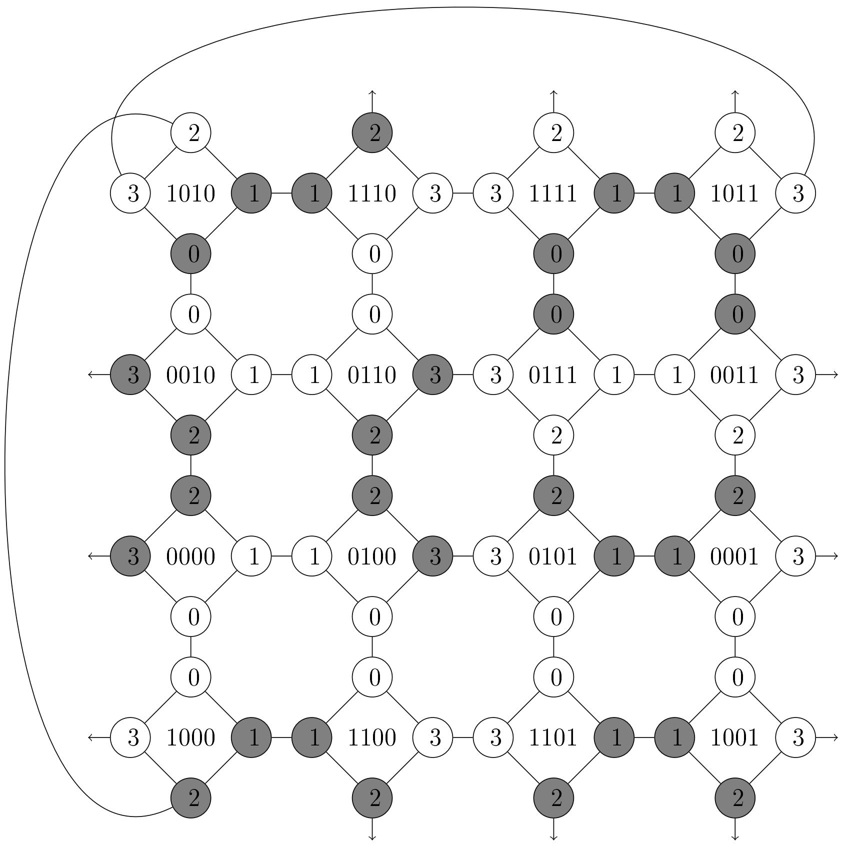


Figure 3.3: A better identifying code in $CCC_4$

## 3.2 Butterfly graph

The *butterfly graph* of dimension $m$, denoted as $BF_m$, is a variant of the hypercube graph of dimension $m$. It has been introduced for use in parallel computing. For more details, consult [dR94] or [HKP$^+$05].

$$\begin{aligned} V(BF_m) = \;& \{0, ..., m-1\} \times \{0,1\}^m \\ E(BF_m) = \;& \left\{ \{ (i, \alpha), ((i+1) \ mod \ m, \alpha) \} \middle| i \in \{0, ..., m-1\}, \alpha \in \{0,1\}^m \right\} \\ & \cup \left\{ \{ (i, \alpha), ((i+1) \ mod \ m, \alpha(i)) \} \middle| i \in \{0, ..., m-1\}, \alpha \in \{0,1\}^m \right\} \end{aligned}$$

where for $\alpha = \alpha_0 \alpha_1 ... \alpha_{m-1} \in \{0,1\}^m, i \in \{0, ..., m-1\}, \alpha(i) = \alpha_0 \alpha_1 ... \overline{\alpha_i} ... \alpha_{m-1}$ (the *ith* bit of $\alpha$ is switched).

For some examples, see Figure 3.4 for the Butterfly graph of dimension 3, and Figure 3.5 for the Butterfly graph of dimension 4 (cross edges between levels 0 and 3 are not represented).

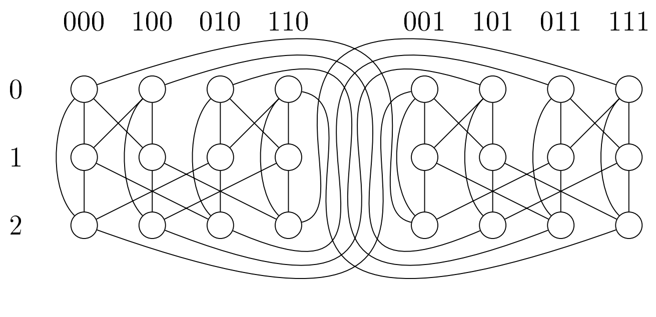


Figure 3.4: The graph $BF_3$

Since any $BF_m$-graph is 4−regular, following Theorem 2.1, we have the following proposition:

**Proposition 3.3** *Let $m \in \mathbb{N}$ and $C$ be an identifying code of $BF_m$.*
*Then* $|C| \geq \dfrac{|V(BF_m)|}{3}$.

We approach this bound with the following result:

**Theorem 3.4** *Let $m \geq 4$. Then* $M_1(BF_m) \leq \left\lceil \dfrac{3m}{4} \right\rceil \cdot 2^{m-1} \simeq \dfrac{3}{8} \cdot |V(BF_m)|$.

**Proof** Consider the functions $f, g : \{0,1\}^m \to \{0,1\}$, where

$f(\alpha) = \sum_{i=0}^{m-1} \alpha_i \; mod \; 2$ and $g(\alpha) = \sum_{\{i \mid i=0 \; mod \; 4\}} \alpha_i \; mod \; 2$

Now, let us construct the code $C$ as follows:
$C = \{(i, \alpha) \mid f(\alpha) = 0 \; and \; i \neq g(\alpha) \; mod \; 4\}$
In other words, only vertices of every second cycle are taken into $C$. On those cycles, every fourth vertex is not in the code; in half of them, the vertices which are out are those numbered 0 *mod* 4, in the others its those numbered 1 *mod* 4. An example of that code on the graph $CCC_4$ is shown in Figure 3.5. The cross edges between levels 0 and 3 are omitted, and the code vertices are gray.

Clearly, we have $\left\lceil \dfrac{3m}{4} \right\rceil$ vertices of every second cycle in $C$, so in total $|C| =$

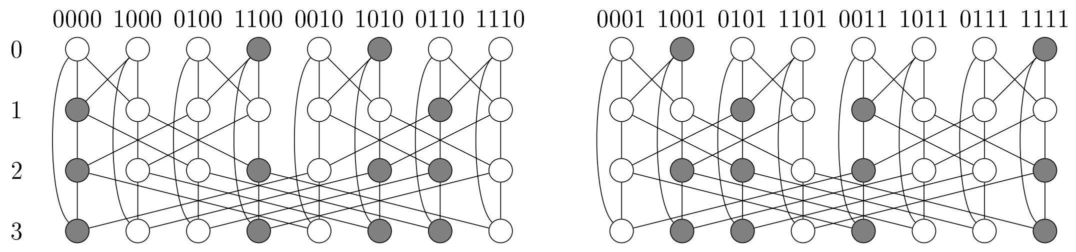


Figure 3.5: The graph $BF_4$ with the code given in the proof of Thm. 3.4

$\left\lceil \frac{3m}{4} \right\rceil \cdot 2^{m-1}$.

Let us show that $C$ is an identifying code. We first check that it is a dominating set. Let $x = (i, \alpha) \in V(BF_m)$.

1. if $f(\alpha) = 0$, then either $x$ or its two neighbours on the same cycle are in $C$, so it is dominated.
2. if $f(\alpha) = 1$, $x$ has two neighbours in other cycles: $y = (i-1, \alpha(i-1))$ and $z = (i+1, \alpha(i))$. Note that $f(\alpha(i-1)) = f(\alpha(i)) = 0$.
   (a) if $y \in C$, $x$ is dominated.
   (b) if $y \notin C$ because $g(\alpha(i-1)) = 0$ and $i = 0 \ mod\ 4$, then $i + 1 = 2 \ mod\ 4$ and thus $z \in C$, so $x$ is dominated.
   (c) if $y \notin C$ because $g(\alpha(i-1)) = 1$ and $i = 1 \ mod\ 4$, then $i + 1 = 3 \ mod\ 4$ and thus $z \in C$, so $x$ is dominated.

Let us now show that $C$ is separating all vertices. Let $x = (i, \alpha), y = (i', \alpha') \in V(BF_m)$, and distinguish between the following cases:

1. $f(\alpha) = f(\alpha') = 0$
   No vertex of the cycle $\alpha$ can be connected to a vertex of the cycle $\alpha'$. Thus, the vertices whichdominate $x$ and $y$ on their respective cycles, also separate them.
2. $f(\alpha) = f(\alpha') = 1$
   We know that both $x$ and $y$ have at least one of their two neighbours on different cycles which are in $C$. Let $n_x$ and $n_y$ be these two neighbours. If $n_x$ and $n_y$ are distinct, $x$ and $y$ are separated by them. So, suppose $n_x = n_y$. Without the loss of generality, suppose that $n_x = (i+1, \alpha(i))$ and $y = (i+2, \alpha(i)(i+1))$. Then, the other neighbours of $x$ and $y$ on other cycles, are the

following: $n'_x = (i-1, \alpha(i-1))$ and $n'_y = (i+3, \alpha(i)(i+1)(i+2))$.
Since $n_x = (i+1, \alpha(i)) \in C$, we have either that $g(\alpha(i)) = 0$ and $i+1 \neq 0$ mod 4, or $g(\alpha(i)) = 1$ and $i+1 \neq 1$ mod 4. If $i+1 = 0$ mod 4 or 1 mod 4, then $i-1 = i+3 = 2$ or 3 mod 4, and thus $n'_x, n'_y \in C$, and we're done.
Otherwise, $i+3 = i-1 = 0$ or 1 mod 4. If $n'_x \in C$, we're done.
So, suppose not. This means that $i-1 \neq g(\alpha(i-1))$ *mod* 4.
But then, by definition of $g$, if we switch four bits in a row in $\alpha(i-1)$ to obtain $\alpha(i-1)(i-1)(i)(i+1)(i+2) = \alpha(i)(i+1)(i+2)$, we have $i-1 = g(\alpha(i)(i+1)(i+2))$ *mod* 4. Since $i-1 = i+3$, this means that $n'_y \in C$, and thus $x$ and $y$ are separated.

3. $f(\alpha) \neq f(\alpha')$
Without the loss of generality, suppose $f(\alpha) = 0$. By construction of $C$, $x$ has at least two neighbours in $C$. Since $y$ can only be connected to one of them, $x$ and $y$ are separated by the other one.

So, $C$ is an identifying code of $BF_m$. □

Note that the code constructed in the previous proof is minimal: no code vertex can be put out of the code. However, there exists a gap between the lower bound and the size of this code. For the graph $BF_4$, the constructed code has 24 vertices, and according to Proposition 3.3, any identifying code of $BF_4$ has at least 22 vertices. Moreover, we ran a computer program on the graph $BF_4$ to check wether there exists an identifying code of smaller cardinality in it. Since this graph has 64 vertices, it is not possible to test the $2^{64}$ possible subsets of vertices in a reasonable time. So, we tested subsets containing only vertices of every second cycle, as the code constructed above. The result of this test was negative, so either this is an optimal code, or one has to construct a code containing vertices of more than half of the cycles of $BF_4$. It is likely that this remark also holds for butterfly graphs of higher dimensions. So, we gave an upper bound for $M_1(BF_m)$ which is quite close to the lower bound, since for all $m = 0$ mod 4, the ratio between the cardinality of the constructed code and the value of the lower bound is precisely equal to $\frac{9}{8}$; for other values of $m$, the ratio is close. However, there still exists a gap which can be filled either by proving a higher lower bound, or by constructing a smaller identifying code of $BF_m$.

# Chapter 4

# Upper bounds depending on the maximum degree

In a graph on $n$ vertices, the upper bound of an identifying code is known to be $n-1$. Moreover, it is tight and there exist an infinity of graphs attaining this bound (see Theorem 1.12 [CHL07, GM07]). However, all such graphs exhibited in [CHL07, GM07] have a very high maximum degree ($n-2$ or $n-1$). This leaded us to make the assumption that graphs of maximum degree $\Delta$ have an identifying code of cardinality at most $n - \frac{n}{f(\Delta)}$, where $f$ is a polynomial in $\Delta$. Intuitively, if we start in a graph $G$ with $C = V(G)$ as an identifying code of $G$, removing a vertex $v$ from $C$ only affects the identification of vertices which are not too distant from $v$, allowing us to take out another vertex which is sufficiently distant from $v$. In fact, there are no known graphs admitting a minimum identifying code bigger than $n - \frac{n}{\Delta}$, leading to the conjecture that $f(\Delta) = \Delta$:

**Conjecture 4.1** *Let $G = (V, E)$ be a connected, twin-free graph on $n$ vertices, with maximum degree $\Delta$. Then there exists an identifying code $C$ of $G$ such that $|C| \leq \left\lceil n - \frac{n}{\Delta} \right\rceil$.*

We were not able to prove this conjecture. However, we show the existence of an upper bound of the form $n - \frac{n}{\Delta}$ where $f$ is a polynomial in $\Delta$ in arbitrary graphs, and quadratic in regular graphs. We then give a better upper bound for triangle-free graphs where $f$ is linear in $\Delta$. We finally show that in graphs of girth at least 5 and minimum degree 2, the influence of the maximum degree of a graph $G$ on the value of $M_1(G)$ is weaker. Finally, we investigate the case of subcubic graphs.

The results of this section are summarized in Table 4.1. The entries for every class of graphs are in form of a pair $\langle b_1, b_2 \rangle$ of functions of $n$ and $\Delta$.

- $b_1(n, \Delta)$ denotes the highest known value for a minimum identifying code in graphs of the correspondant class: for any $\Delta$, there exist arbitrarily large values of $n$, for which some graph of this class, having $n$ vertices and maximum degree $\Delta$, admits no identifying code smaller than $b_1(n, \Delta)$.

- $b_2(n, \Delta)$ denotes the new theoretical upper bound for this class: for any graph of this class having $n$ veryices and maximum degree $\Delta$, we are able to construct an identifying code of cardinality at most $b_2(n, \Delta)$.

| | arbitrary graphs | $\Delta$-regular graphs |
|---|---|---|
| arbitrary graphs | $\left\langle n - \frac{n}{\Delta},\ n - \frac{n}{\Theta(\Delta^4)} \right\rangle$ ⟨ Prop. 4.2, Thm. 4.3 ⟩ | $\left\langle n - \frac{n}{\Delta},\ n - \frac{n-1}{\Delta^2} \right\rangle$ ⟨ Prop. 4.2, Thm. 4.6 ⟩ |
| triangle-free graphs | $\left\langle n - \frac{n}{\Delta - 1 + \frac{1}{\Delta}},\ n - \frac{n}{3\Delta+3} \right\rangle$ ⟨ Prop. 4.7, Thm. 4.10 ⟩ | $\left\langle n - \frac{n}{\frac{2\Delta}{3}},\ n - \frac{n}{2\Delta+2} \right\rangle$ ⟨ Prop. 4.12, Thm. 4.13 ⟩ |
| graphs of girth at least 5 | $\left\langle n - \frac{n}{\Delta - 1 + \frac{1}{\Delta}},\ n - \frac{n}{3\Delta+3} \right\rangle$ ⟨ Prop. 4.7, Thm. 4.10 ⟩ | $\left\langle \frac{2n}{\Delta+2},\ n - \frac{n-8}{8} \right\rangle$ ⟨ Thm. 2.1, Cor. 4.15 ⟩ |

Table 4.1: Summary of the new upper bounds for identifying codes in graphs of maximum degree $\Delta \geq 3$ and $n$ vertices

## 4.1 Arbitrary graphs

We now discuss the upper bound of identifying codes in arbitrary graphs of maximum degree $\Delta$.

First, consider the example of the following graph, denoted as $CK_m(G_{l,m})$, where $G_{l,m}$ is an $m$-regular graph of $l$ vertices. $CK_m(G_{l,m})$ consists of the graph $G_{l,m}$ in which every vertex is replaced by a clique of size $m$. So, $CK_m(G_{l,m})$ consists of $l$ connected cliques of size $m$. It has $l \cdot m$ vertices, and it is $m$-regular. See Figure 4.1 for an example with $l = 4$ and $l = 3$ where $G_{4,3}$ is the complete graph $K_4$.

Consider two vertices $u, v$ of the same clique. In order to separate them, at least one of their two respective neighbours in the other cliques must be in the code. Now, repeat this argument for all pairs of vertices of the same clique: in order to separate the $m$ vertices of the clique, $m - 1$ vertices outside of it have to be in the code. Since every vertex has a unique neighbour in another clique, in total at least $m - 1$ vertices per clique must be in the code.

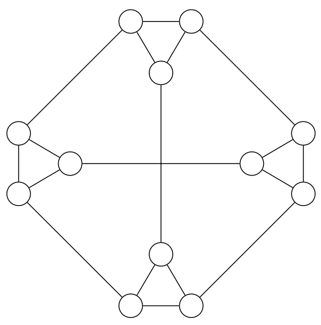


Figure 4.1: The graph $CK_3(K_4)$

So, an identifying code of $CK_{G_{l,m},m}$ has a size of at least
$l(m-1) = lm - \frac{lm}{m} = n - \frac{n}{m}$.

This leads to the following proposition:

**Proposition 4.2** *For any value of $\Delta \geq 3$, there exist arbitrarily large values of $n$, such that there exists a $\Delta$-regular graph on $n$ vertices admitting no identifying code of a smaller size than $n - \frac{n}{\Delta}$.*

Of course, the bound of this proposition also holds for graphs of maximum degree $\Delta$ since a $\Delta$-regular graph has maximum degree $\Delta$. Note that it is the same bound as the one of Conjecture 4.1; we don't know graphs of maximum degree $\Delta$ having a minimum identifying code of larger cardinality.

### 4.1.1 General case

Let us consider the case of arbitrary graphs. We prove the following result:

**Theorem 4.3** *Let $G = (V, E)$ be a connected, twin-free graph with $n$ vertices, and maximum degree $\Delta$. Then there exists an identifying code $C$ of $G$ such that*
$|C| \leq n - \frac{n}{\Delta(\Delta^3 + \Delta^2 - \Delta + 1)} = n - \frac{n}{\Theta(\Delta^4)}$.

**Proof** We first show the following lemma:

**Lemma 4.4** *Let $G = (V, E)$ be a connected twin-free graph. Let $a \in V$ be a vertex of $V$. Then there exists a vertex $b \in V$, $d(a, b) \leq 2$, such that $V \backslash \{b\}$ is an identifying code, and:*

- *$b = a$, or*

- *there exist two vertices $u, v \in V$ such that $N[u] = N[v] \cup \{a\}$ and:*
  - *$N[a] \subseteq N[u]$ and $b = u$, or*
  - *$d(a, b) = 1$ and $B_2(b) \subseteq B_2(a)$, or*
  - *$d(a, b) = 2$ and $B_1(b) \subseteq B_2(a)$*

**Proof of Lemma 4.4** Let $a \in V$. If there are no two vertices $u, v \in V$ such that $N[u] = N[v] \cup \{a\}$, then clearly $V \backslash \{a\}$ is an identifying code.
So, suppose there are two vertices $u, v \in V$ such that $N[u] = N[v] \cup \{a\}$ (See Figure 4.2).

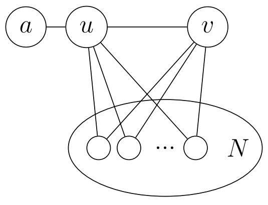


Figure 4.2: A vertex $a$ such that $V \backslash \{a\}$ is not an identifying code

Let us distinguish the following cases:

- $N[a] \subseteq N[u]$: $a$ has no neighbours other than $u$ or vertices of $N$
  Then, take $V \backslash \{u\}$ as a code.
  Let $x, y \in V$ such that $u \in N[x] \oplus N[y]$. Let us check that $x$ and $y$ are separated. Without loss of gnerality, $u \in N[x]$ and $u \notin N[y]$.
  1. $x = u$ or $x = v$
     Then $y$ is not connected to $v$ (else $xy \in E$ and $u \in N[y]$), so $x$ and $y$ are separated by $v$.
  2. $x \in N$
     Then $y \notin N$(else $u \in N[y]$), hence $x$ and $y$ are separated by $v$.
  3. $x = a$
     Then $y = u$. Since $a$ is not connected to $v$, and all its neighbours are connected to $v$, $a$ and $u$ are separated by $v$.
- $a$ has at least one neighbour $z$ other than $u$
  - $\forall m \in N, N[m] \oplus N[a] \neq \{v\}$
    Then, take $V \backslash \{v\}$ as a code.
    Let $x, y \in V$ such that $v \in N[x] \oplus N[y]$. Without loss of generality, $v \in N[x]$ and $v \notin N[y]$.

1. $x = v$
   Then $xy \notin E$, otherwise $v \in N[y]$. If $y$ not connected to $u$, $x$ and $y$ are separated by $u$. Else, $y = a$; but since $a$ is in the code, $x$ and $y$ are separated by $a$.
2. $x = u$
   If $xy \notin E$, $x$ and $y$ are separated by $x = u$ which is in the code. Otherwise, $y = a$ since $v \notin N[y]$. Since $a$ has a neighbour $z$ which is not in $N$, $x$ and $y$ are separated by $z$.
3. $x \in N$
   If $xy \notin E$, $x$ and $y$ are separated by $x$ which is in the code. Otherwise, if $yu \notin E$, they are separated by $u$ which is connected to $x$ and is in the code. If $y$ is connected to $u$, $y = a$ since $v \notin N[y]$. Since by hypothesis, $N[x] \oplus N[a] \neq \{v\}$, $x$ and $y$ are separated by either a neighbour of $y = a$, which is in the code, or a neighbour of $x$. Since this neighbour is at distance 2 of $a$, it is in the code as well, except if we did take it out with a similar move. This would mean that there is a vertex $a'$ at distance 4 of $a$, and a similar configuration where the neighbour of $x$ is some $v'$. But then $u'$ would be connected to $x$ and therefore $a'$ would be at distance 3 of $a$, which is not possible.

Notice that $d(a, v) = 2$. Since $N[u] = N[v] \cup \{a\}$, all neighbours of $v$ are neighbours of $u$. Since $u$ and $a$ are connected, this means that $B_1(v) \subseteq B_2(a)$.

– $\exists m \in N, N[m] \oplus N[a] = \{v\}$
Let us denote the common neighbourhood of $a$ and $m$ as $N_a$.
If there are no vertices $x, y \in V$ such that $N[x] \oplus N[y] = \{m\}$, clearly $V \backslash \{m\}$ is an identifying code. We have $d(a, m) = 1$. Since $N[m] = N[a] \cup \{v\}$, $B_2(m) = B_2(a) \cup B_1(v)$. But $B_1(v) \subseteq B_2(a)$, so $B_2(m) \subseteq B_2(a)$.
Let $x, y \in V$ such that $\{m\} = N[x] \oplus N[y]$. So, $xy \in E$. Without loss of generality, $m \in N(x)$ and $m \notin N(y)$. Then:

* $x \notin N_a$: Since $y$ is not connected to $m$, it is not connected to $a$. Thus $a \in N[x] \oplus N[y]$.
* $x \neq a$ since all the neighbours of $a$ are connected to $m$.
* $x \neq u$: Since $y$ is not connected to $m$, $y \in N$. Then, $y$ is not connected to $a$, otherwise it would be connected to $m$. So $a \in N[x] \oplus N[y]$.

So $x = v$, $y \in N$, and $y$ is connected to all vertices of $N$ except $m$; it has no other neighbours.
Now, let us partition $N$ into $N_1$, being the vertices of $N$ having no neighbours out of $N \cup \{u, v\} \cup N[a]$, and $N_2 = N \backslash N_1$. If a vertex of $N_1$ can be out of the code, we are done.

Suppose not. Notice that $y$ cannot be out of the code if and only if there is a vertex $p$ of $N_1$ such that there exists a second vertex $q \in N$ and $N[p] = N[q] \cup \{y\}$ (let us call $p$ a blocking vertex for $y$). Similarly, $p$ can also only be removed if there exist such vertices in $N$. Notice that the blocking vertices can themselves only be blocked by a vertex in $N_1$, and two vertices can only be blocked by two distinct vertices. So, we can define an order on the vertices of $N_1 = \{p_0, ..., p_l\}$ where $p_0 = m$, $p_1 = y$ and $p_{i+1}$ blocks $p_i$. Notice that the last vertex $p_l$ can not be blocked; thus, it can be removed from the code.
So there exists a vertex $p \in N$ such that $p$ can be out of the code, and $p$ has all its neighbours within $N \cup \{u, v\} \cup N[a]$: $d(a, p) \leq 2$ and $B_1(p) \subseteq B_2(a)$.

(End of the proof of Lemma 4.4) □

Let us now construct a 4-independent set $S$ of $G$ of cardinality at least $\frac{n}{B_4}$ (where $B_4 = \Delta(\Delta^3 + \Delta^2 - \Delta + 1)$ is the maximal size of a ball of radius four in $G$), using the greedy Algorithm 4.1.

**Algorithm 4.1** Greedy construction of a 4-independent set $S$

```
Require: a connected graph $G = (V, E)$
1: $X \leftarrow V$
2: $S \leftarrow \emptyset$
3: while $X \neq \emptyset$ do
4:     arbitrarily choose $s \in X$
5:     $S \leftarrow S \cup \{s\}$
6:     $X \leftarrow X \backslash B_4(s)$
7: end while
8: return $S$
```

Then we take $C = V \backslash S$, modify it slightly and show that it is an identifying code of $G$.

We first check the separation condition. It has to be verified that for every pair of vertices $u, v \in V$, at least one vertex of $N[u] \oplus N[v]$ belongs to the code.

Let $u, v \in V$ be two arbitrary vertices.
Since $S$ is a 4-independent set (so two vertices of $S$ are at least at distance 5 of each other), it is clear that in all cases except configurations 4.9(h) and 4.10(b), $u$ and $v$ are separated.
Now, suppose $u$ and $v$ are in one of these cases (see Figure 4.3) : $u$ and $v$ are

connected and have the same set of neighbours $N$ (possibly empty), except $a$ which is connected to $u$, and $a \in S$. $v$ has no neighbours out of $N$.

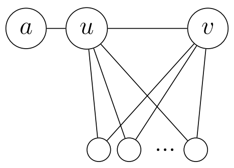


Figure 4.3: Vertices $u$ and $v$ are not separated

Suppose $a \in S$ : $u$ and $v$ are not separated. Using Lemma 4.4, we take $a$ out from $S$, and put another vertex $b$ into $S$ instead, thus not changing its cardinality.

However, $b$ is at distance at most 2 from $a$, thus $S$ might not be a 4-independent set anymore, and there could be pairs of vertices now becoming unseparated. Let $a'$ be another vertex originally placed in $S$. We know that $d(a, a') \geq 5$.
By Lemma 4.4, we know that $B_1(b) \subseteq B_2(a)$. This implies that $b$ is at least at distance 4 from $a'$. (Indeed, suppose not: $d(b, a') \leq 3$. Consider a shortest path $b$-$x$-$y$-$a'$ between $b$ and $a'$. Since $B_1(b) \subseteq B_2(a)$, $d(a, x) \leq 2$ and thus $d(a, a') \leq 4$, which is a contradiction.)
So, the only case where there could be two unseparated vertices is if both $a$ and $a'$ had been replaced by vertices $b, b'$, with $d(b, b') = 3$; so $d(a, b) \geq 1$ and $d(a', b') \geq 1$. Let us distinguish the following cases:

- $d(a, b) = 2$ or $d(a', b') = 2$ (Figure 4.4)

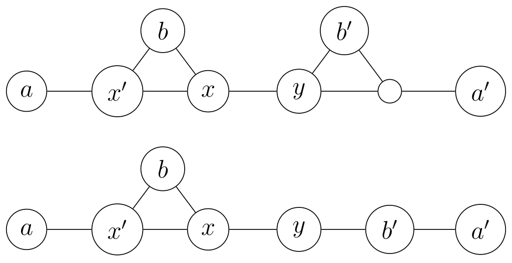


Figure 4.4: Unseparated vertices (the two variants of subcase 1)

Without loss of generality, $d(a, b) = 2$.
Suppose there exist two vertices $x, y$ such that $\{b, b'\} \subseteq N[x] \oplus N[y]$. Since $d(b, b') = 3$, none of $x$ and $y$ is connected to both $b$ and $b'$. Since $G$ is connected, $x$ and $y$ are neighbours. Without loss of generality, $x$ is a neighbour of $b$ and $y$ is a neighbour of $b'$. Since by Lemma 4.4, we know that $B_1(b) \subseteq B_2(a)$,

$x$ is connected to a vertex $x'$ at distance less than one from $a$ so $d(a,x) \leq 2$. Similarly, $d(a',y) \leq 2$: if $d(a',b') = 2$, apply the same argument, and if $d(a',b') = 1$ it is obvious.
Clearly, $x'$ is not in $S$, and $y$ is not connected to it (else $d(a,a') = 4$ which is a contradiction). So $x$ and $y$ are well separated by $x'$.

- $d(a,b) = 1$ and $d(a',b') = 1$ (Figure 4.5)

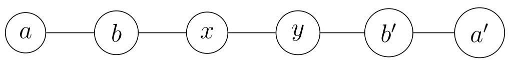


Figure 4.5: Unseparated vertices (subcase 2)

Then, following Lemma 4.4, the following cases are possible:

– There exist two vertices $u, v$ such that $N[u] = N[v] \cup \{a\}$ and $b = u$.
Suppose there exist two vertices $x, y$ such that $\{b, b'\} \subseteq N[x] \oplus N[y]$. Since $d(b,b') = 3$, none of $x$ and $y$ is connected to both $b$ and $b'$. Since $G$ is connected, $x$ and $y$ are neighbours. Without loss of generality, $x$ is a neighbour of $b$ and $y$ is a neighbour of $b'$.
If $x = v$, since $N[u] = N[v] \cup \{a\}$, $u$ is connected to $x$; $a$ and $a'$ are at distance 4 from each other, which is a contradiction.
So, $x$ is another neighbour of $u$ (obviously not $a$). But then, $x$ is connected to $v$ since $N[u] = N[v] \cup \{a\}$. And if $y$ was connected to $v$, it would be connected to $u$ and then $d(a,a') = 4$ which is a contradiction. Thus, $y$ is not connected to $v$, which is clearly not in $S$, and thus $x$ and $y$ are separated by $v$.

– $B_2(b) \subseteq B_2(a)$
This case cannot happen. Otherwise, since $d(a,a') = 5$ and $B_2(b) \subseteq B_2(a)$, $d(b,b') \geq 4$ which is a contradiction.

Finally, it is clear that the set $C = V \backslash S$ is an independent set; thus, since $G$ is connected, every vertex has at most one vertex of its closed neighbourhood outside of the code, and it is a dominating set.

So, $C$ is an identifying code. □

### 4.1.2 Regular graphs

Let us now study the particular case of regular graphs. We first make the following observation:

**Observation 4.5** *Let $G = (V, E)$ be a $\Delta-$regular graph. Let $u, v \in V$. Then $|N[x] \backslash N[y]| = |N[y] \backslash N[x]|$ and $|N[x] \oplus N[y]|$ is even.*

**Proof** Assume $|N[x] \cap N[y]| = I$. Since $G$ is $\Delta-$regular, $|N[x]| = |N[y]| = \Delta + 1$. Thus $|N[x] \backslash N[y]| = |N[y] \backslash N[x]| = \Delta + 1 - I$, and $|N[x] \oplus N[y]| = 2 \cdot (\Delta + 1 - I)$.□

We are now able to show the following theorem:

**Theorem 4.6** *Let $G = (V, E)$ be a $\Delta$-regular ($\Delta \geq 3$), connected, twin-free graph of $n$ vertices. Then there exists an identifying code $C$ of $G$ such that $|C| \leq n - \dfrac{n-1}{\Delta^2}$.*

**Proof** We first construct a 2-independent set $S$ of $G$, using the greedy Algorithm 4.2.

**Algorithm 4.2** Greedy construction of a 2-independent set $S$

```
Require: a connected graph $G = (V, E)$
1: $X \leftarrow V$
2: $S \leftarrow \emptyset$
3: while $X \neq \emptyset$ do
4:    if $S \neq \emptyset$ then
5:       choose $s \in X$ such that $\exists\ x \in S, d(x, s) \leq 2$
6:    else
7:       arbitrarily choose $s \in X$
8:    end if
9:    $S \leftarrow S \cup \{s\}$
10:   $X \leftarrow X \backslash B_2(s)$
11: end while
12: return $S$
```

We observe that at the first step of the while loop, at most $B_2max$ vertices are removed from $X$, where $B_2max = 1 + \Delta + \Delta \cdot (\Delta - 1) = \Delta^2 + 1$ is the maximum size of a ball of radius 2 in $G$.

In the other steps, since we choose $s$ as a vertex which has at least one neighbour which has already been removed from $X$, at most $B_2max - 1 = \Delta^2$ are removed. Thus, the set $S$ returned by the algorithm has a size of at least $\dfrac{n-1}{\Delta^2}$.

We now modify $S$ in order to guarantee that $V \backslash S$ becomes an identifying code.

We first check the separation condition. It has to be verified that for every pair of vertices $u, v \in V$, at least one vertex of $N[u] \oplus N[v]$ belongs to the code.

Let $u, v \in V$.We now distinguish different cases depending on the set $N[u] \oplus N[v]$. These cases are illustrated in Figure 4.6.

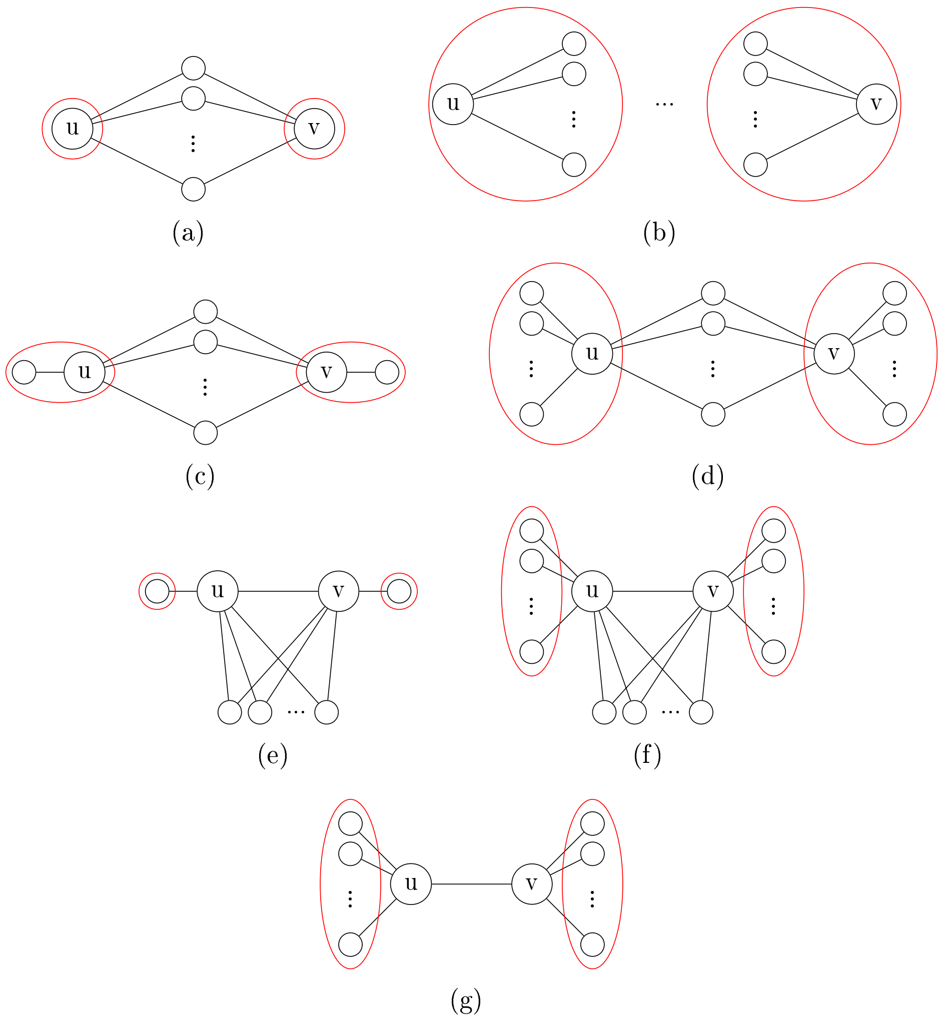


Figure 4.6: Configurations for symmetric differences between the closed neighbourhoods of two vertices $u$ and $v$ in regular graphs. The symmetric differences are circled in red.

First notice that for all cases, since $S$ is a 2-independent set, two elements of $S$ are at least at distance 3 from each other. Thus, in all cases listed in Figure 4.6 (except configuration 4.6(e)), at least one vertex of $N[u] \oplus N[v]$ belongs to the code $V \backslash S$, thus separating $u$ from $v$.

Now, suppose $u$ and $v$ are in the case 4.6(e): $N[u] \oplus N[v] = \{a, b\}$ and both $a$ and $b \in S$ (Figure 4.7). Then $u$ and $v$ are not separated by $V \backslash S$. Let us denote as $N$, the set of common neighbours of $u$ and $v$.

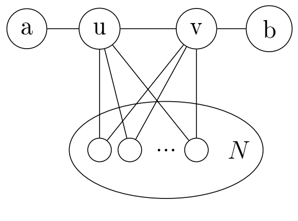


Figure 4.7: Configuration in which $u$ and $v$ are not separated

Then, let us put $v$ into $S$ and $b$ out of $S$. Vertices $u$ and $v$ are now separated. Let us show that if there exist $x, y \in V$ such that $v \in N[x] \oplus N[y]$, then $x$ and $y$ will be separated by another vertex of $V$ than $v$, showing that $v$ can be put into $S$ safely. (Note that the modified set $S$ may not be a 2-independent set anymore, but it is still a 1-independent set.

So, let $x, y \in V$ and $v \in N[x] \oplus N[y]$. Without loss of generality, let us assume that $x \in N[v]$ and $y \notin N[v]$. Let us distinguish the following cases:

1. $x = v$

   We know that $N[v] \cap V \backslash S = N \cup \{u, b\}$.

   If $y$ is not connected to all vertices of $N \cup \{u, b\}$, $x$ and $y$ are thus separated by those vertices, and we are done.

   Now assume that $y$ is connected to all these vertices. Since $y$ is connected to $u$, but not to $v$, $y \in (N[u] \oplus N[v]) \backslash \{b\} = \{a\}$. So $y = a$, but since $y$ is connected to $b$, $d(a, b) = 2$ which is a contradiction since $a$ and $b$ are at distance 3 from each other.

2. $x = u$

   If $xy = uy \notin E$, $x$ and $y$ are separated by $y = u$ since $u \notin S$ and we are done.

   So, assume $xy \in E$. Since $y \notin N[v]$, $y = a$. Since $u$ is connected to all vertices of $N$, which are all in $V \backslash S$, if there exists one vertex $n$ of $N$ such that $yn \notin E$, $n$ separates $x$ from $y$ and we are done.

   Suppose $y$ is connected to all vertices of $N$. There are exactly $\Delta - 2$ vertices in $N$, so there exists a vertex $z \in V$ such that $zy = za \in E$, but $zx = zu \notin E$. Since $d(z, a) = 1$, $z \notin S$ and $x$ and $y$ are separated by $z$.

3. $x \in N$

   Since $y \notin N[v]$, $y \notin N$. If $x$ and $y$ are not connected, they are separated by $x$ since $x \in N$ and all vertices of $N$ are in $V \backslash S$.

So, assume $xy \in E$. If $y$ is not connected to $u$, then $x$ and $y$ are separated by $u$ since $u \notin S$.

Assume $yu \in E$ ; then $y = a$. But then, since $G$ is regular, there exists at least one vertex $z$ such that $y$ is connected to $z$, but $x$ is not. Since $d(z, y) = d(z, a) = 1$, $z \notin S$ and therefore $x$ and $y$ are separated by $z$.

4. $x = b$
If $xy \notin E$, since $x = b \notin S$, $x$ and $y$ are separated by $x$ and we are done.

So, suppose $xy \in E$. Since $G$ is regular, $x$ is connected to $v$ but $y$ is not, following Observation 4.5, there exists at least one vertex $z$ such that $yz \in E$ and $xz \notin E$. Then, $d(x, z) = 2$ and thus $z \notin S$: $z$ separates $x$ from $y$.

However, it is possible that after another conflict such as this one concerning another pair of vertices $u'$ and $z = v'$, the vertex $z$ is put into $S$ following our transformation of $S$; then the previous argument is not valid anymore.

We show that if this occurs, in any case there is a vertex $z'$ separating $y$ from $b$. So, suppose there exist $u'$ and $z = v' \in V$, such that $a'$ and $b' \in S$ and $N[u'] \oplus N[v'] = \{a', b'\}$, leading to put $z = v'$ into $S$ (Figure 4.8).

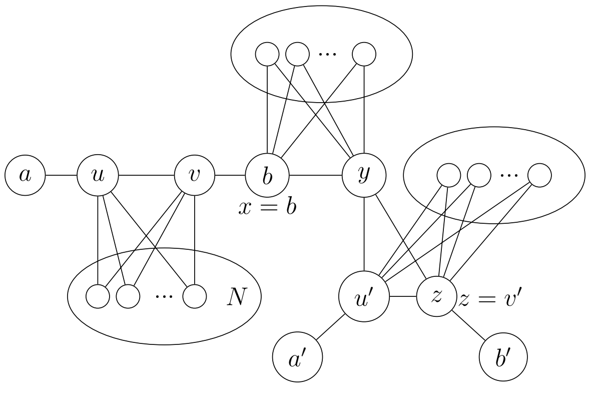


Figure 4.8: Modifying $S$ in order to separate $u$ and $v$

First, note that $bu' \notin E$: if yes, then $bz = bv' \in E$, which is a contradiction since we assumed that $z$ is not connected to $b$.

Second, suppose $u' = y$. Then $a' = b$: otherwise, since $by = bu' \in E$, we would have $bv' = bz \in E$ which is a contradiction. But then, $u'$ and $v'$ are separated by $b$, which is now out of $S$, and there is no need to put $z = v'$ into $S$, and we are done.

Finally, suppose $u' \neq b$. Then, since $yv' = yz \in E$, we have $yu' \in E$. With our construction, we know that $u'$ is not in $S$; thus, $y$ and $b$ are not separated by $z$, but by $u'$ and we are done.

Note that in the cases 1, 2 and 3, the identifying vertex $z$ is at distance 1 of a vertex which stays in $S$. Since $S$, as observed before, even after modification, is at least a 1-independent-set, these vertices are never put into $S$ and the construction remains stable, as opposed to case 4.

It has now to be checked that $V \backslash S$ is a dominating set for $G$. Since $S$ is an independent set, every vertex of $V$ has at least one neighbour which is not in $S$ and thus it is dominated.

So $C = V \backslash S$ is an identifying code of $G$ of size at most $n - \dfrac{n-1}{\Delta^2}$. □

## 4.2 Triangle-free graphs

We now discuss the upper bound for the cardinality of an identifying code in triangle-free graphs having a maximum degree of $\Delta$. Some of the results of this section will be submitted soon for publication [FKKR09].

### 4.2.1 General case

Note that in the complete $k$-ary tree ($k \geq 3$) with $h$ levels and $n = \dfrac{k^h - 1}{k-1}$ vertices, which is triangle-free and has a maximum degree of $\Delta = k+1$, the optimal identifying code has a size of $M = \left\lceil \dfrac{k^2 n}{k^2+k+1} \right\rceil$ [BCHL05]. This gives us:

$$\begin{aligned} M = &\left\lceil \frac{k^2 n}{k^2+k+1} \right\rceil = \left\lceil \frac{(\Delta-1)^2 n}{(\Delta-1)^2+\Delta} \right\rceil \\ M = &\left\lceil n - \frac{\Delta n}{(\Delta-1)^2+\Delta} \right\rceil \\ M = &\left\lceil n - \frac{n}{\Delta - 1 + \frac{1}{\Delta}} \right\rceil \geq n - \frac{n}{\Delta - 1 + \frac{1}{\Delta}} \end{aligned}$$

This leads to the following proposition:

**Proposition 4.7** *For any value of $\Delta \geq 3$, there exist arbitrarily large values of $n$, such that there exists a triangle-free graph on $n$ vertices and maximum degree $\Delta$ admitting no identifying code of a smaller size than $n - \dfrac{n}{\Delta - 1 + \frac{1}{\Delta}}$.*

Note that this bound is actually weaker than the bound of Proposition 4.2 for arbitrary graphs of maximum degree $\Delta$. So, an upper bound for the size of an identifying code in triangle-free graphs depending on parameters $n$ and $\Delta$ cannot be smaller than the bound of Proposition 4.7.

Moreover, in the complete bipartite graph $K_{a,b}$ on $a+b$ vertices ($a \geq 2$, $b \geq 3$) (which has a maximum degree of $\max(a,b)$), the smallest identifying code has a size of $n-2$. Indeed, for every pair of vertices of the same bipartition, since they have the same neighbours, it is only possible for one of them to be out of the code. So, only one vertex of each bipartition can be out of the code. In addition, since $\frac{n}{\Delta} = \frac{a+b}{\max(a,b)} \leq \frac{2\max(a,b)}{\max(a,b)} = 2$, $M_1(K_{a,b}) \leq n-2 \leq n - \frac{n}{\Delta}$. Let us hold that in the following Observation:

**Observation 4.8** *Let $a \geq 2$, $b \geq 3$ and $K_{a,b}$ be the complete bipartite graph with bipartitions of sizes $a$ and $b$ and maximum degree $\Delta = \max(a,b)$. Then $M_1(K_{a,b}) \leq n - \frac{n}{\Delta}$.*

Notice that the inequality of the previous observation turns into an equality exactly when $a=b$ and $K_{a,b}$ is the complete bipartite graph $K_{\Delta,\Delta}$ on $n = 2\Delta$ vertices. This bound is higher than the one of Proposition 4.7, but given a value of $\Delta$, it is only valid for one particular graph, namely the complete bipartite graph $K_{\Delta,\Delta}$. It leads to the following proposition:

**Proposition 4.9** *For any value of $\Delta \geq 3$, there exists a triangle-free graph $G$ on $n$ vertices having maximum degree $\Delta$, which admits no identifying code with a size smaller than $n - \frac{n}{\Delta}$.*

We now approach this bound with the following theorem:

**Theorem 4.10 ([FKKR09])** *Let $G = (V,E)$ be a connected, twin-free graph of maximum degree $\Delta$ ($\Delta \geq 3$), having $n$ vertices and no triangles. Then there exists an identifying code $C$ of $G$ such that $|C| \leq n - \frac{n}{3\Delta+3}$.*

**Proof** In this proof, we first construct an independent set $S$ of $G$ having the property that two vertices of $S$ do not share the same neighbourhood, using the greedy Algorithm 4.3. $S$ is the union of two sets $S_1$ and $S_2$, with the property that for each vertex $s$ of $S_1$ having degree at least 2, there exists no other vertex $t$ such that $N(s) = N(t)$. We then modify $S$ so that the set $V \backslash S$ becomes a valid identifying code of $G$. There are two cases to handle. The first case concerns vertices of degree one: we locally modify the code without changing its cardinality. For the other case, we construct a subgraph of $G$ containing all vertices involved in that case. In this subgraph, we provide three different manners of constructing a set which, if added to the code of the rest of the graph, give a valid identifying code of $G$. We are then able to bound the cardinality of these sets.

**Algorithm 4.3** Greedy construction of an independent set $S$ in which no vertices share the same neighbourhood

```
Require: a connected graph $G = (V, E)$
 1: $X \leftarrow V$
 2: $S_1 \leftarrow \emptyset$
 3: $S_2 \leftarrow \emptyset$
 4: while there exists a vertex $s \in X$ with $deg_G(s) = 1$ do
 5:     $S_1 \leftarrow S \cup \{s\}$
 6:     $X \leftarrow X \backslash B_2(s)$
 7: end while
 8: while $X \neq \emptyset$ do
 9:     arbitrarily choose $s \in X$
10:     if there exists a vertex $t$ having the same set of neighbours as $s$ then
11:         $S_2 \leftarrow S_2 \cup \{s\}$
12:         $X \leftarrow X \backslash \Big(B_1(s) \cup \{t \in V \mid N(s) = N(t)\}\Big)$
13:     else
14:         $S_1 \leftarrow S_1 \cup \{s\}$
15:         $X \leftarrow X \backslash B_1(s)$
16:     end if
17: end while
18: return $S_1$ and $S_2$
```

The constructed set $S = S_1 \cup S_2$ is an independent set having the property that no two elements of $S$ have the same neighbourhood. Indeed, when a vertex is chosen to be in $S$, all vertices having the same neighbourhood are removed from the set of candidates (line 16 of the algorithm). Moreover, if a vertex of degree 1 is in $S$, then all vertices at distance two or less are not.

Assume $|S| = \alpha \cdot |S_1| + (1-\alpha) \cdot |S_2|$ for some $\alpha \in [0,1]$. We now make the following statements on the sizes of $S_1$ and $S_2$:
$|S_1| \geq \dfrac{\alpha \cdot n}{\Delta + 1}$ and $|S_2| \geq \dfrac{(1-\alpha) \cdot n}{2\Delta - 1}$.

Indeed, only vertices of degree 1 and vertices for which there is no other vertex sharing the same set of neighbours are put into $S_1$. Let $s$ be a chosen vertex of degree 1 and let $s'$ be its unique neighbour. When $s$ is put into $S_1$, all vertices of $B_2(s)$ are removed from the set $X$. Since $B_2(s) = B_1(s')$, then at most $B_1 = \Delta + 1$ vertices are removed from $X$ when adding $s$ to $S$, where $B_1$ is the maximum size of a ball of radius 1 in $G$ (line 6 of the algorithm). Let $s$ be a vertex of degree at least 2 chosen to be in $S_1$. When $s$ is put into $S_1$, at most $B_1(s)$ vertices are put outside of $X$. Thus, the set $S_2$ has a size of at least $\dfrac{\alpha \cdot n}{\Delta + 1}$.

Now let $s$ be a vertex being added to $S_2$. When $s$ is added to $S_2$, at most $B_1 + N_1$ vertices are removed from $X$, where $B_1 = \Delta + 1$ is the maximum size of a ball of radius 1 in $G$, and $N_1$ is the maximum number of vertices in $V$ having the same set of neighbours as $s$. We assume $N_1 \leq \Delta - 2$ since if there are $\Delta - 1$ such vertices, since $G$ is connected, the graph would consist only of those vertices, and $G$ would be the complete bipartite graph $K_{\Delta,d}$ where $d \leq \Delta$ is the number of neighbours of $s$. In that case, we know that the graph admits an identifying code with a size of at most $n - \dfrac{n}{\Delta}$ (Observation 4.8), so we would be done. Thus, the set $S_2$ has a size of at least $\dfrac{(1-\alpha)\cdot n}{2\Delta - 1}$.

We now modify $S$ in order to guarantee that $V \backslash S$ becomes an identifying code.

We first check the separation condition. It has to be verified that for every $u, v \in V$, at least one vertex of $N[u] \oplus N[v]$ belongs to the code.

Let $u, v \in V$ be two arbitrary vertices. We now distinguish different cases depending on the set $N[u] \oplus N[v]$. These cases are illustrated in Figure 4.9.

Note that since $G$ has no triangles, the configurations listed in Figure 4.10 cannot appear in $G$.

If $u$ and $v$ are in configuration of Figure 4.9(a), by construction of $S$ (removal of the vertices having the same neighbours at line 16 of Algorithm 4.3), either $u$ or $v$ are not in $S$ and thus $u$ and $v$ are separated.

For the configurations of Figures 4.9(b), 4.9(c), 4.9(d), 4.9(e) and 4.9(f), since $S$ is an independent set, at least one vertex of $N[u] \oplus N[v]$ is not in $S$, separating $u$ and $v$.

Now, suppose $u$ and $v$ are in one of the configurations of Figures 4.9(h) or 4.9(k): $u$ and $v$ are connected, and $u$ is of degree 1. Suppose all neighbours of $v$ other than $u$ belong to $S$ ($u$ and $v$ are not separated). By construction of $S$, we know that $u \notin S$, else all neighbours of $v$ would be in the code. For the same reason, no other neighbour of $v$ has degree 1. So both $u$ and $v$ are in the code and all other neighbours of $v$ have a degree greater than 2.
Let us locally modify $S$ without changing its cardinality. Let $m$ be a neighbour of $v$ which is in $S$. Remove $m$ from $S$ and put $u$ into $S$ instead: $u$ will be identified by $v$; $v$ by itself and $m$; and since $S$ is an independent set, all other neighbours of $m$ are in the code and identify $m$, and we are done.

Now, suppose $u$ and $v$ are in one of the configurations of Figures 4.9(g), 4.9(i) or 4.9(j): $u$ and $v$ are connected, and both $u$ and $v$ have at least one other neighbour. Suppose also that all the vertices of $N[u] \oplus N[v]$ are in $S$: $u$ and $v$ are not separated.

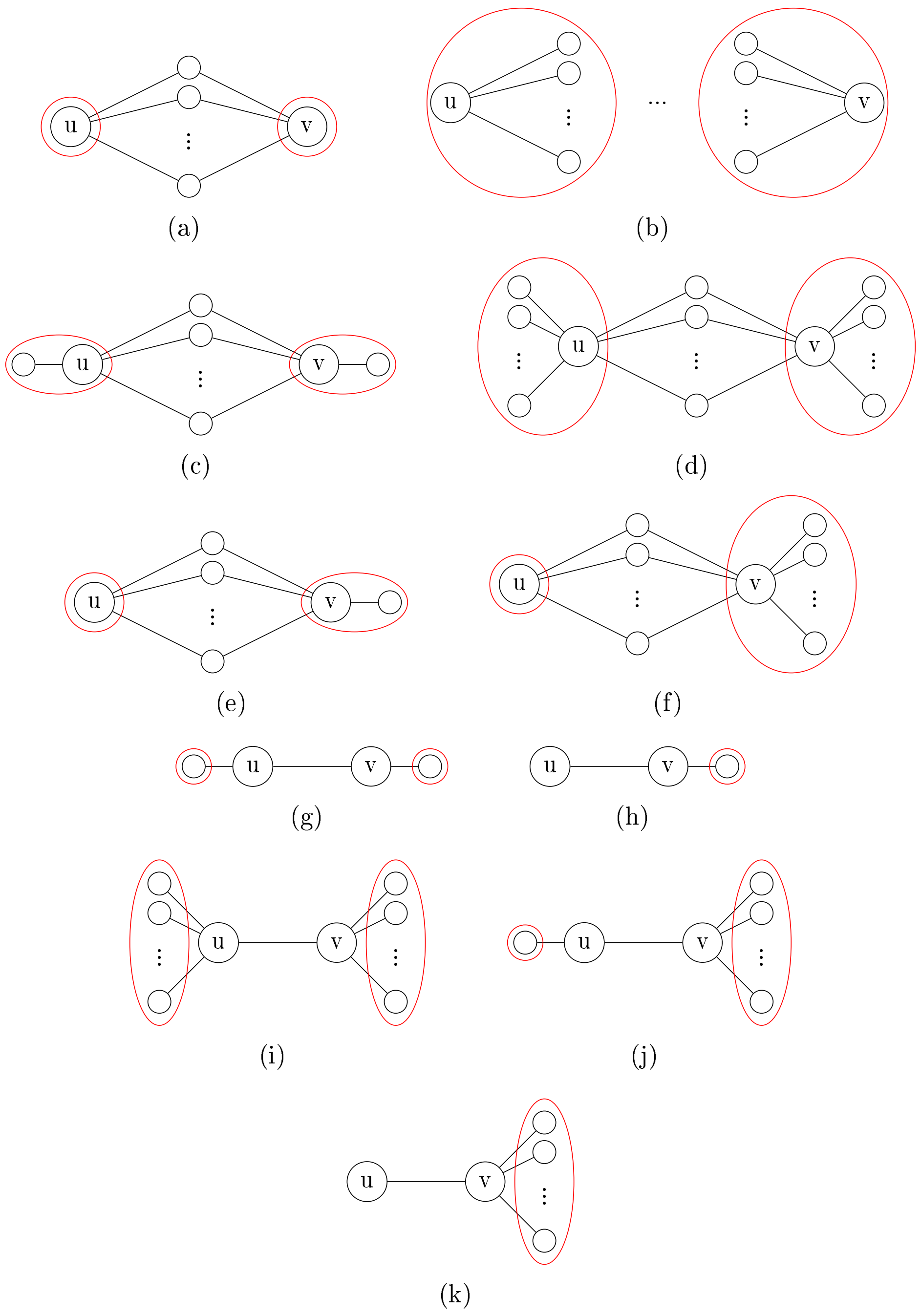


Figure 4.9: Triangle-free configurations for symmetric differences between the closed neighbourhoods of two vertices $u$ and $v$. The symmetric differences are circled in red.

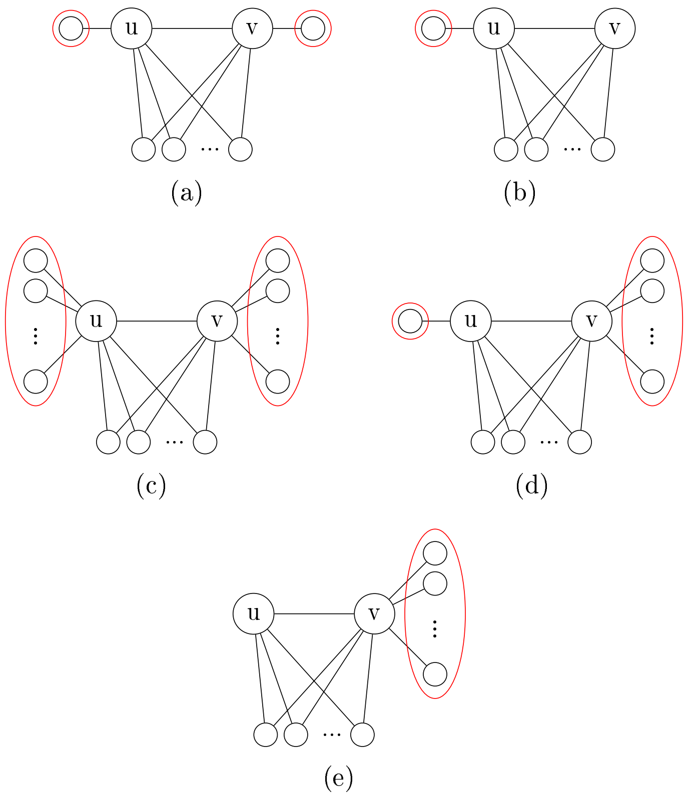


Figure 4.10: Configurations including triangles for symmetric differences between the closed neighbourhoods of two vertices $u$ and $v$. The symmetric differences are circled in red.

Consider the set $R$ of all the vertices belonging to a pair $\{u, v\}$ such that $u$ and $v$ are in such a configuration, and are not separated. Let us denote as $L$, the set of all neighbours of elements of $R$ which are not in $R$: $L = \left(\bigcup_{v \in R} N(v)\right) \backslash R$. Note that by definition, $L \subseteq S$, so no two vertices of $L$ are connected in $G$. In addition, each vertex of $R$ has exactly one neighbour in $R$, and at least one in $L$. Since $G$ is triangle-free, no two connected vertices of $R$ share neighbours.

Note that $L \subseteq S_1$. Indeed, suppose not. Then, there exists a vertex $l \in L$ such that $l \in S_2$. Then there exists another vertex $x$ such that $N(l) = N(x)$. Let $r$ be a neighbour of $l$ being in $R$. By construction of $S_1$ and $S_2$, $x$ is in the code, and since $N(l) = N(x)$, $r$ and $x$ are connected. Therefore $x$ is in $R$ since we assumed that vertices of $R$ have one unique neighbour in the code, this neighbour being in $R$. Then, by definition of $R$, $x$ has a neighbour $l'$ in $L$, $l'$ and $l$ being distinct (otherwise $r$, $x$ and $l$ form a triangle). But since $N(l) = N(x)$, $l$ is connected to $l'$ which is a contradiction since $L$ is an independent set. So $l \subseteq L$.

We will now show three different methods for replacing the set $L$ in the induced subgraph $G[L \cup R]$ by another set $L'$, not too small with respect to $L$, such that if $L'$ is taken out of the code instead of $L$, all vertices are still separated in $G$. This will allow us to take $S' = (S \backslash L) \cup L'$ out of the code, and $V \backslash S'$ as an identifying code. We will then show that in every case, the maximum cardinality of $(L \cup R) \backslash L'$ is bounded with respect to $|R|$, which is the original set supposed to be in the code.

First, we divide the sets $L$ and $R$ into the following subsets. Let $R_1 \subseteq R$ be such that $r \in R_1$ if both $r$ and its unique neighbour in $R$ have only one neighbour each in $L$. Let $L_1 \subseteq L$ be the set of all neighbours of vertices of $R_1$. Let $R_2 = R \backslash R_1$ and $L_2 = L \backslash L_1$. See Figure 4.11 for an illustration.

We now show the following Lemma:

**Lemma 4.11**

1. *$G[L_1 \cup R_1]$ admits an identifying code of size at most $|L_1| + \dfrac{|R_1|}{2}$.*
2. *$G[L_1 \cup R_1]$ admits an identifying code of size at most $|L_1| + \dfrac{|R_1|}{2} + \dfrac{|L_1|}{2}$ in which the whole set $L_1$ belongs to the code.*

**Proof of Lemma 4.11** We first construct a code $C_1$ of $G[L_1 \cup R_1]$ by putting $L_1$ into $C_1$, and aritrarily picking exactly one vertex of each pair of connected vertices of $R_1$ into it: $|C_1| = |L_1| + \dfrac{|R_1|}{2}$. It is clear that $C_1$ is a dominating set for $G[L_1 \cup R_1]$.

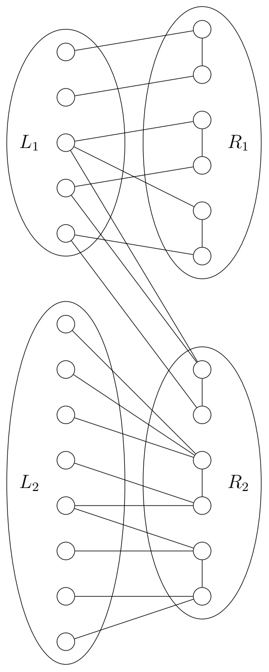


Figure 4.11: Illustration of the sets $L_1$, $L_2$, $R_1$, and $R_2$

However, there might be some vertices which are not separated. (See Figure 4.12 for an example; code vertices are gray).

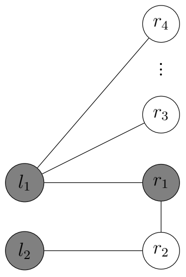


Figure 4.12: Vertices $l_1$ and $r_1$ are not separated

Let us construct the directed graph $\overrightarrow{G_1} = (L_1, A_1)$ where
$A_1 = \left\{ \overrightarrow{l_1 l_2} \mid l_1, l_2 \in L_1, \exists r_1, r_2 \in R_1, r_1 \notin C_1, r_2 \in C_1, \text{ and } l_1 r_1 r_2 l_2 \text{ is a path in } G[L_1 \cup R_1] \right\}$. An illustration is given in Figure 4.13.

So, the situation without separation in $G[L_1 \cup R_1]$ is equivalent to having a vertex in $\overrightarrow{G_1}$ with an in-degree of 1. We will now modify the orientation of $\overrightarrow{G_1}$ in order to avoid this, using Algorithm 4.4. This algorithm constructs an arbitrary spanning forest of $\overrightarrow{G_1}$. It goes over all vertices of each tree, level by level from the leaves up to

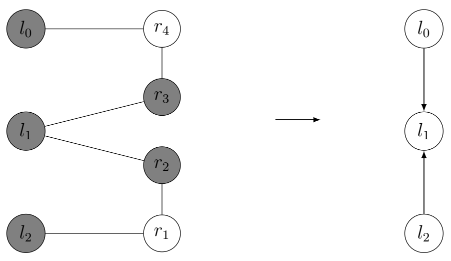


Figure 4.13: Construction of the graph $\overrightarrow{G_1}$

the root. It modifies the orientation of the arcs of the tree so that the in-degree of all vertices (except eventually for the root) is even (and therefore different from 1). If the in-degree of a vertex $v$ is odd, the arc pointing from $v$ to its parent is reversed, making the in-degree of $v$ even, and adding 1 to the in-degree of the parent. Since the algorithm considers the vertices level by level from the leaves up to the root, at the end of the process only the root can have an odd in-degree.

Now, we apply the reverse tranformation from $\overrightarrow{G_1}$ to $G[L_1 \cup R_1]$ and construct the code $C_2$ of $G[L_1 \cup R_1]$ based on the orientations of the arcs of $\overrightarrow{G_1}$. Note that the size of $C_2$ is still the size of $C_1$.
It remains to handle the case of the roots of the spanning trees, which might be in the situation depicted in Figure 4.12 if their in-degree is equal to one in $\overrightarrow{G_1}$, after the execution of the algorithm.
If we want the set $L_1$ to be in the code, it is sufficient to add one extra vertex per each such vertex into $C_2$ (see Figure 4.14; code vertices are gray and vertex $l_1$ is the root of the spanning tree).

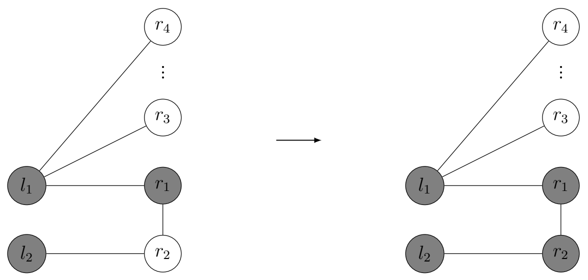


Figure 4.14: Local modification of the code by adding one vertex

Since there is at most one such vertex per spanning tree and one spanning tree per connected component, we get $|C_2| \leq |L_1| + \frac{|R_1|}{2} + cc(G[L_1 \cup R_1])$. Note that $cc(G[L_1 \cup R_1]) \leq \frac{|L_1|}{2}$ since every connected component contains at least two vertices

**Algorithm 4.4** Modification of the orientation of the arcs of $\overrightarrow{G_1}$

```
Require: the graph $\overrightarrow{G_1}$
1: for all connected components $\overrightarrow{H_1}$ of $\overrightarrow{G_1}$ do
2:   construct a spanning tree $\overrightarrow{T_1}$ of $\overrightarrow{H_1}$ (for example using a classic linear greedy algorithm, or Algorithm 4.5 used in section 4.2.3 to construct a DFS spanning tree)
3:   let $r$ be the root of $\overrightarrow{T_1}$, and $h$ its height
4:   let $\overrightarrow{H'_1} = (V(\overrightarrow{H_1}), A(\overrightarrow{H_1})\backslash A(\overrightarrow{T_1}))$ be the graph $\overrightarrow{H_1}$ without the edges of the spanning tree $\overrightarrow{T_1}$
5:   for all verticess $v$ of $\overrightarrow{T_1}$ do
6:     $label(v) \leftarrow indeg_{\overrightarrow{H'_1}}(v)$
7:   end for
8:   for all $i$ from $h$ to 1 do
9:     for all vertices $v$ of level $i$ of $\overrightarrow{T_1}$ do
10:      let $\overrightarrow{e} \in A(\overrightarrow{T_1})$ be the edge between $v$ and its parent vertex $p$
11:      if $label(v)$ is even then
12:        orient $e$ from $v$ to $p$
13:        $label(p) \leftarrow label(p) + 1$
14:      else
15:        orient $e$ from $p$ to $v$
16:        $label(v) \leftarrow label(v) + 1$
17:      end if
18:    end for
19:  end for
20:  //at the end of the process, the indegree of each vertex is precisely equal to its label
21: end for
```

of $L_1$. Hence, we get $|C_2| \leq |L_1| + \frac{|R_1|}{2} + \frac{|L_1|}{2}$.

If we do not need that $L_1$ is in the code, we can just switch one vertex in every connected component of $G[L_1 \cup R_1]$ as shown in Figure 4.15 (the gray vertices are in the code, and vertex $l_1$ is the root of the spanning tree), leading to a code of size $|C_2| \leq |L_1| + \frac{|R_1|}{2}$.

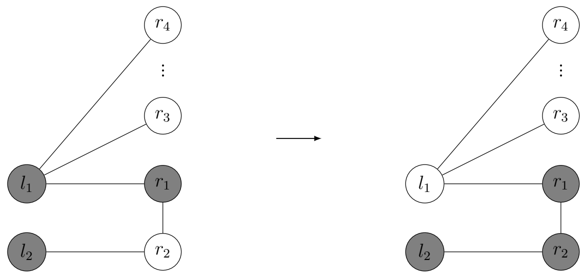


Figure 4.15: Local modification of the code by switching two vertices

So, in any case we get a code in which all vertices are dominated (a vertex is either dominated by itself or by its neighbour in $R_1$).
Let us show that all vertices are separated as well.
Let $u, v$ be two vertices of $L_1 \cup R_1$.

- if $u, v \in L_1$, if one of them is in the code, they are separated by this vertex. If not (due to a shifting like in Figure 4.15), they both have a neighbour in $R_1$ which is in the code, and are thus separated by this neighbour.
- if $u, v \in R_1$, for each of these two vertices, either itself or its neighbour in $R$ is in the code. If $u$ and $v$ are not connected, they are separated by these two vertices. If they are connected, at least one of them has a neighbour in $L_1$ which is in the code (at most one vertex of $L_1$ is not in the code per connected component) and they are separated by it since they do not share neighbours ($G$ is triangle-free).
- if $u \in L_1$ and $v \in R_1$, if $v$ is not in the code, then its neighbour in $R_1$ is, and $u$ and $v$ are separated by it. If $v$ is in the code, then either $u$ has another neighbour in $R_1$ which is in the code due to the application of Algorithm 4.4, or the neighbour of $v$ is in the code due to the modifications done after the application of the algorithm. Thus $u$ and $v$ are separated.

(End of the Proof of Lemma 4.11) $\square$

We now provide three distinct ways of replacing the set $R$ by another set, such that all vertices in $G[L \cup R]$ are separated. Those sets will fulfill the following properties:

**Property** $(*)$

- for every pair of connected vertices of $R$, at least one is in the code.
- for every pair of connected vertices of $R$, at least one which is in the code has at least one neighbour in $L$ which is in the code as well.

Note that these two properties are clearly fulfilled by the solutions for $G[L_1 \cup R_1]$ obtained in the proof of Lemma 4.11.

Let us now describe the three codes:

1. Construct a code $C$ by putting $L_2$ into it, a code for $G[L_1 \cup R_1]$ having the whole set $L_1$ in it (computed using the proof of Lemma 4.11), and half of the set $R_2$ such that for every pair of connected vertices of $R_2$, exactly one is in the code, and it has at least two neighbours in $L$ (such vertices exist by definition of $R_2$).
   This set has a size less than
   $|L_2| + \frac{|R_2|}{2} + |L_1| + \frac{|R_1|}{2} + \frac{|L_1|}{2} = |L_2| + \frac{3|L_1|}{2} + \frac{|R_1|}{2} + \frac{|R_2|}{2}$.
   Let us show that this is an identifying code.
   First, note that since the whole set $L$ is in $C$, it is clearly a dominating set.
   Let us now check it for separation: let $u, v \in G[L \cup R]$.
   - if both $u$ and $v \in L$, they are separated by $u$ and $v$ since $L$ is an independent set and both are in the code.
   - if $u, v \in R$ and they are not connected, since every vertex of $R$ is either in the code or has its neighbour in the code, $u$ and $v$ are separated. Else, they are separated by their respective neighbours in $L$, which they do not share since $G$ is triangle-free, and which are both in the code.
   - if $u \in L$ and $v \in R$, and they are not connected, they are separated by $u$, which is in the code. Else, if $v$ is not in the code, then its neighbour in $R$ is, and $u$ and $v$ are separated by it. Else, if $v \in R_2$, then $v$ has at least another neighbour in $L$ than $u$, and $u$ and $v$ are separated by it. If $v \in R_1$, then we already know that $u$ and $v$ are separated (Lemma 4.11).
2. Construct $C$ as follows: take $L_2$ and a solution for $G[L_1 \cup R_1]$ provided by the proof of Lemma 4.11 into $C$. Take also a subset of the set $R_2$ of cardinality $\frac{|R_2|}{2}$ such that for every pair of connected vertices of $R_2$, exactly one is in the code, and it has at least two neighbours in $L$. Finally, if there exist some code vertices of $R_2$ such that less than two of their neighbours are in $C$ (this can happen if these neighbours are in $L_1$), then put one of these neighbours into $C$. Note that there can be at most $\frac{|R_2|}{2}$ such vertices.
   So, the size of $C$ is at most $|L_2| + 2\frac{|R_2|}{2} + |L_1| + \frac{|R_1|}{2} = |L_2| + |R_2| + |L_1| + \frac{|R_1|}{2}$.
   Let us show that this is an identifying code.
   Note that $C$ is a dominating set: all vertices of $L_2$ are dominated by themselves, all vertices of $R$ are dominated either by themselves or by their neighbour in $R$, and all vertices of $L_1$ are dominated according to Lemma 4.11.
   Let us now check $C$ for separation: let $u, v \in G[L \cup R]$.

- if $u, v \in L_1 \cup R_1$, they are separated according to Lemma 4.11.
- if $u \in R_1$ and $v \in R_2$, then $u$ and $v$ are not connected; thus they are separated either by themselves or by their neighbours in $R$.
- if $u \in L_2$ and $v$ is not connected to $u$, then they are separated by $u$, which is in the code. In particular, if $v \in L_1 \cup L_2 \cup R_1$, they are separated since they cannot be connected.
- if $u \in L_2$ and $v \in R_2$ and $u$ and $v$ are connected, if $v \notin C$ they are separated by the neighbour of $v$ in $R$, which must be in the code. If not, then $v$ has at least one other neighbour in $L$ which is in the code (we made this clear in the last step of the construction of $C$), so $u$ and $v$ are separated by it.
- if $u \in L_1$ and $v \in R_2$, if $v \notin C$, then its neighbour in $R_2 \in C$, thus $u$ and $v$ are separated by it. Else, we know that $v$ has at least one neighbour in $L$ in the code. If there is one such neighbour other than $u$, then $u$ and $v$ are separated by this neighbour. If $u$ is the only one, this means it has been put into the code in the last step of the construction of $C$. Thus, it was not in the solution for $G[L_1 \cup R_1]$, which means that it has a neighbour in $R_1$ in the code. So $u$ and $v$ are separated by this neighbour.
- if $u, v \in R_2$ and are not connected, then they are separated by themselves or their neighbours in $R_2$. If they are connected, then the one which is in the code has at least one neighbour in $L$ in the code, so they are separated by this neighbour.

3. Construct $C$ as follows: take $R_2$ and a solution for $G[L_1 \cup R_1]$ provided by the proof of Lemma 4.11 into $C$. Put as well into $C$, for every pair of connected vertices of $R_2$, one neighbour in $L$ of either one of the vertices, into $C$.
   So, the size of $C$ is at most $|R_2| + \frac{|R_2|}{2} + |L_1| + \frac{|R_1|}{2} = 3\frac{|R_2|}{2} + |L_1| + \frac{|R_1|}{2}$.
   Let us show that $C$ is an identifying code.
   $C$ is a dominating set: all vertices of $L_2$ are dominated by their neighbours in $R_2$, all vertices of $R$ are dominated either by themselves or by their neighbour in $R$, and all vertices of $L_1$ are dominated according to Lemma 4.11.
   Let us now check $C$ for separation: let $u, v \in G[L \cup R]$.

   - if $u, v \in L_1 \cup R_1$, they are separated according to Lemma 4.11.
   - if $u, v \in R_2$ then $u$ and $v$ are separated by the neighbour of one of them which has been put into $C$ in the last step of the construction.
   - if $u \in R_2$ and $v \notin R_2$, then $u$ and $v$ are separated by the neighbour of $u$ which is in $R_2$.
   - if $u \in L_1$ and $v \in L_2$, then according to the solution of Lemma 4.11, either $u$ or a neighbour of $u$ in $R_1$ is in $C$, so since $L$ is an independent set, $u$ and $v$ are separated.

- if $u \in L_2$ and $v \in R_1$, then either $v$ or its neighbour in $R_1$ is in the code. Since no vertex of $L_2$ is connected to one of $R_1$, $u$ and $v$ are separated.
- if $u, v \in L_2$, and they have different neighbours in $R_2$, then they are separated by them. Else, note that since by construction of $S$ with Algorithm 4.3, no vertices of $S$, and therefore of $L$, share the same neighbourhood. So, at least one of $u, v$ has a neighbour in $V \backslash (L \cup R)$ which is not connected to the other vertex. Since $S$ is an independent set and $u, v \in L \subseteq S$, this vertex is in the code. So $u$ and $v$ are separated by it.

Let us determine the maximum number of vertices which are taken out in any case when choosing one of these three sets. The number of vertices which are not in the code are the following (using the fact that $|L_1| \leq |R_1|$):

- In construction 1: $\dfrac{|R_1|}{2} + \dfrac{|R_2|}{2} - \dfrac{|L_1|}{2} \geq \dfrac{|R_1|}{2} + \dfrac{|R_2|}{2} - \dfrac{|R_1|}{2} = \dfrac{|R_2|}{2}$
- In construction 2: $\dfrac{|R_1|}{2} \geq \dfrac{|L_1|}{2}$
- In construction 3: $|L_2| + \dfrac{|R_1|}{2} - \dfrac{|R_2|}{2} \geq |L_2| + \dfrac{|L_1|}{2} - \dfrac{|R_2|}{2}$

Let $m$ be the maximum of these three expressions:

$$\begin{aligned}
m &= \frac{1}{2} \cdot \max\Big\{|R_2|,\ |L_1|,\ 2|L_2| + |L_1| - |R_2|\Big\}\\
m &= \frac{|L|}{2} \cdot \max\left\{\frac{|R_2|}{|L|},\ \frac{|L_1|}{|L|},\ \frac{2|L_2| + |L_1| - |R_2|}{|L_1| + |L_2|}\right\}\\
m &\geq \frac{|L|}{2} \cdot \max\left\{\frac{\max\big\{|L_1|,\ |R_2|\big\}}{|L|},\ \frac{2|L_1| + 2|L_2| - |L_1| - |R_2|}{|L_1| + |L_2|}\right\}\\
m &\geq \frac{|L|}{2} \cdot \max\left\{\frac{\max\big\{|L_1|,\ |R_2|\big\}}{|L|},\ 2 - 2 \cdot \frac{\max\big\{|L_1|,\ |R_2|\big\}}{|L|}\right\}\\
&\text{Let } b = \frac{\max\big\{|L_1|,\ |R_2|\big\}}{|L|}. \text{ Then:}\\
m &\geq \frac{|L|}{2} \cdot \max\big\{b,\ 2 - 2b\big\}\\
m &\geq \frac{|L|}{2} \cdot \min_{b \geq 0}\Big\{\max\big\{b,\ 2 - 2b\big\}\Big\} \geq \frac{|L|}{2} \cdot \frac{2}{3} = \frac{|L|}{3}
\end{aligned}$$

Notice that this bound is achieved in the case where $|L_1| = |R_1| = |R_2|$ and $|L_1| = 2|L_2|$.

Let us now perform a small modification on the previously defined codes. Everytime there is a vertex $v$ of $L$ which is in the code, for which all neighbours in $R$ are not

in the code, we take $v$ out of the code and put one arbitrary neighbour $r \in R$ of $v$ into the code.
Following Property $(*)$, $v$ will still be separated from $r$ by the neighbour $r'$ of $r$, from $r'$ by one of the neighbours of $r'$ in $L$, and from all other vertices by $r$. All neighbours of $v$ are still dominated by their neighbours in $R$. Finally, $v$ is not needed for separation since all neighbours of $v$ are separated from their own neighbours, who are all in the code, by some vertex in $L$.
So, we get a code in which no vertex of $L$ is in the code without having a neighbour in $R$ in the code as well. Note that this implies that every vertex $v$ of $L \cup R$ has at least one vertex of $R$ being in the code and belonging to $N[v]$.

Now, it remains to check wether those solutions merged with the solution of $G[V \backslash (L \cup R)]$ separate all pairs of vertices. Let $u, v \in V$.

- if $u, v \in L \cup R$, they are separated following the previous discussion.
- if $u, v \in V \backslash (L \cup R)$, they are separated following our first observations.
- suppose $u \in L \cup R$ and $v \in V \backslash (L \cup R)$. Since we made it clear that $u$ has a vertex $r$ of $R$ being in the code and in $N[u]$, since $v$ is not connected to any vertex of $R$, $u$ and $v$ are separated by $r$.

It remains to show that the chosen code is a dominating set for $G$.
Following the previous discussion, is clear that in $G[L \cup R]$, in any case, the obtained set is a dominating set. In the rest of $G$, observe that since $S$ has been constructed as an independent set and $G$ is connected, every vertex has a neighbour in the code; thus, it is a dominating set.

So, we can replace the set $L$ by a set $L'$ which size is at least a third of the size of $L$. Let $S' = (S \backslash L) \cup L'$. By the preceding discussion, we have that $V \backslash S'$ is an identifying code of $G$.

Note that $S = S_1 \cup S_2$, $|S| = \alpha \cdot |S_1| + (1 - \alpha) \cdot |S_2|$ for some $\alpha \in [0, 1]$, $|S_1| \geq \dfrac{\alpha \cdot n}{\Delta + 1}$ and $|S_2| \geq \dfrac{(1 - \alpha) \cdot n}{2\Delta - 1}$. Moreover, $L \subseteq S_1$.

So, $|S'| \geq |S_2| + \dfrac{|S_1|}{3} \geq \dfrac{(1 - \alpha) \cdot n}{2\Delta - 1} + \dfrac{\alpha \cdot n}{3\Delta + 3} \geq \dfrac{n}{3\Delta + 3}$.

Hence, $C = V \backslash S'$ is an identifying code of size at most $n - \dfrac{n}{3\Delta + 3}$. □

### 4.2.2 Regular graphs

We now look at the particular case of regular graphs.

First, consider the example of the following graph, denoted as $CCK_{k,l}$, which consists of $l$ cycle-connected almost-complete bipartite graphs of $2k$ vertices defined as follows:
$V(CCK_{k,l}) = \{0, ..., l-1\} \times \{0,1\} \times \{0, ..., k-1\}$
$E(CCK_{k,l}) = \Big\{(x,0,y),(x,1,y') \mid x \in \{0,...,l-1\},\ y,y' \in \{0,...,k-1\},\ \{y,y'\} \neq \{0,0\}\Big\} \cup \Big\{(x,0,0),((x+1) \bmod l,1,0) \mid x \in \{0,...,l-1\}\Big\}$
Note that $CCK_{k,l}$ has $2 \cdot k \cdot l$ vertices and is $k$-regular. For an example, see Figure 4.16.

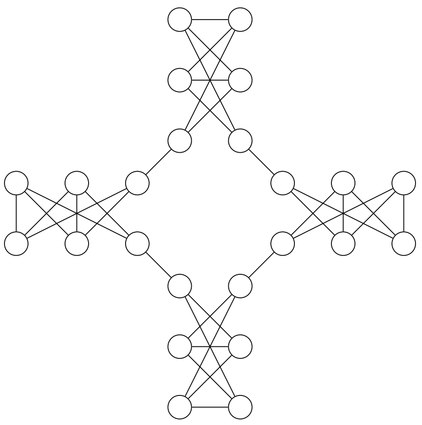


Figure 4.16: The graph $CCK_{3,4}$

It is easy to see that the graph $CCK_{k,l}$ has no identifying code smaller than $l(2k - 3) = 2kl - 3\dfrac{2kl}{2k} = n - 3\dfrac{n}{2k}$.
Indeed, consider all the vertices $(x, i, y)$ where $x \in \{0, ..., l-1\}$ and $i \in \{0,1\}$ are fixed, and $y \in \{1, ..., k-1\}$: this set of vertices is one bipartition of one of the complete bipartite subgraphs of $CCK_{k,l}$, minus the vertex which is connected to the next bipartite subgraph. Then, given two vertices of this set, one can see that they both have exactly the same set of neighbours. Thus, one can not remove two such vertices, otherwise those two would not be separated. So, for every complete bipartite subgraph of the graph (without considering the vertices which connect the complete bipartite subgraphs to each other), one can only leave two vertices out of the code.
Moreover, if we do remove two such vertices per bipartite subgraph, then at least one vertex $(x, i, 0)$ is needed per bipartite subgraph in order to separate it from all other vertices $(x, i, y)$, $y \neq 0$.

This leads to the following proposition:

**Proposition 4.12** *For any value of $\Delta \geq 3$, there exist arbitrarily large values of $n$, such that there exists a triangle-free, $\Delta$-regular graph on $n$ vertices admitting no identifying code of a smaller size than $n - \frac{n}{\frac{2\Delta}{3}}$.*

So, an upper bound for the size of an identifying code in a regular graph depending on its maximum degree, cannot be smaller than the bound of Proposition 4.12. Note that the bound of this proposition is weaker than the one of Proposition 4.7, but that bound is not valid for regular graphs, since the complete tree is not regular. The better bound of Proposition 4.9 still holds, since the complete bipartite graph is regular; however, this proposition is weaker in the sense that there exists only one such graph for a given value of $\Delta$.

We will now try to approach this bound. Note that the result of Theorem 4.10 is still valid in the case of $\Delta$-regular graphs. However, it can be improved using the same proof, together with a small remark.

**Theorem 4.13 ([FKKR09])** *Let $G = (V, E)$ be a $\Delta$-regular ($\Delta \geq 3$), connected, twin-free graph on $n$ vertices having no triangles. Then there exists an identifying code $C$ of $G$ such that $|C| \leq n - \frac{n}{2\Delta + 2}$.*

**Proof** Consider the proof of Theorem 4.10, and proceed exactly the same way, constructing a special independent set $S = S_1 \cup S_2$. When constructing the sets $R$ and $L$, since $G$ is $\Delta$-regular with $\Delta \geq 3$, it is easy to see that the sets $R_1$ and $L_1$ do not exist, since all vertices of $R_1$ have degree two.
So, as possible sets in $G[L \cup R]$ to be put into the identifying code, the solutions (1) of size $|L_2| + \frac{|R_2|}{2} = |L| + \frac{|R|}{2}$ and (3) of size $\frac{3|R_2|}{2} = \frac{3|R|}{2}$ can be used. (Solution (2) does not make any sense since it has a size of $|L_2| + |R_2| = |L| + |R|$).
So, we can always take out at least $\max\left\{\frac{|R|}{2}, |L| - \frac{|R|}{2}\right\}$ vertices from the code, within $G[L \cup R]$. This expression is at its minimum value when $|L| = |R|$; then it has a value of $\frac{|L|}{2}$.

So we can take out at least $\frac{|L|}{2}$ vertices instead of $|L|$.
The rest of the proof is exactly the same; in the worst case, we end with an identifying code of cardinality at most $n - \frac{1}{2} \cdot \frac{n}{\Delta + 1} \leq n - \frac{n}{2\Delta + 2}$. □

### 4.2.3 Graphs of larger girth

It is possible to improve the previous results for a smaller class of graphs. The previously considered triangle-free graphs are the graphs of girth at least 4; in this section, we will consider graphs of girth at least 5.

Note that the bound of Proposition 4.7 is still valid for this class of graphs, since trees contain no cycles, and in particular no cycles of length 4 or less.

Note that in graphs of girth at least 5, there do not exist two vertices $u, v$ of $V$ and of degree strictly greater than 1, for which $N(u) = N(v)$. Therefore, the proof of Theorem 4.10 could be simplified, since there would then be no need to compute the set $S_2$. However, this does not lead to an improved result.

Let us consider graphs of girth 5 and of minimum degree $\delta \geq 2$ (in particular, trees are not in this class of graphs since all leaves of a tree have degree 1). Then, we are able to show the following upper bound for the minimum cardinality of an identifying code of such a graph. Note that, as opposed to all previous results, this bound does not depend on the maximum degree of the graph.

**Theorem 4.14** *Let $G = (V, E)$ be a connected, twin-free graph with minimum degree $\delta \geq 2$ , having $n$ vertices and girth $g \geq 5$. Then there exists an identifying code $C$ of $G$ such that $|C| \leq \frac{7n}{8} + 1 = n - \frac{n-8}{8}$.*

**Proof** Let us first construct a *Depth-First-Search tree* (*DFS-tree*) $T$ of $G$, for example using the classic recursive Algorithm 4.5.

**Algorithm 4.5** Recursive construction of a DFS-tree of a graph

```
Require: a connected graph $G = (V, E)$, a vertex $v$ of $V$ and a tree $T = (V_T, E_T)$.
    The algorithm is initially called with an empty tree and an arbitrary vertex.
1: add $v$ to $V_T$
2: for all neighbours $w$ of $v$ do
3:   if $w$ is not in $V_T$ then
4:     add $w$ to $V_T$
5:     add the edge $vw$ to $E_T$
6:     recursively call the algorithm on $G$, $w$ and $T$
7:   end if
8: end for
9: return $T$
```

Let us define the levels of $T$ as integers such that the root of $T$ is at level 0, and the children of a vertex at level $i$ are at level $i + 1$. Let us denote the level of a vertex $v$ by $level(v)$.

A DFS-tree $T$ has the well-known property that for any edge $uv$ of $E(G)$, either $uv$ belongs to $T$, $u$ is an ancestor of $v$ in $T$, or $v$ is an ancestor of $u$ in $T$.

Moreover, since $G$ has girth at least 5, the previous property directly implies that a vertex $v$ such that $level(v) = i$, is not connected to any vertices of levels $i-3, i-2, i, i+2$ or $i+3$. In addition, it is connected to exactly one vertex of level $i-1$, its parent (except for the root, which has no parent) and all vertices except leaves are connected to exactly one vertex of level $i+1$. In particular, note that two leaves of $T$ cannot be connected since no leaf can be an ancestor of another leaf.

Also note that since $G$ has girth at least 5 and minimum degree 2, the level of a leaf $v$ of $T$ is at least 4: indeed, $v$ has necessarily a neighbour other than its parent but can only be connected to ancestors at levels at most $level(v) - 4$.

Let us now partition the vertices of $G$ into four sets $V_0, V_1, V_2$ and $V_3$ defined as follows: for $i \in \{0,1,2,3\}$, $V_i = \Big\{v \in V(G) \mid level(v) = i \bmod 4 \text{ in } T\Big\}$.

Consider one of these sets, $V_m$, $m \in \{0,1,2,3\}$, such that $|V_m| \geq \frac{n}{4}$. Such a set necessarily exists; otherwise there would be fewer than $n$ vertices in $G$. Let us take $C = V \backslash V_m$ as a code. Obviously, $|C| \leq \frac{3n}{4}$. An illustration is given in Figure 4.17 with $m = 1$. Black vertices are in the code.

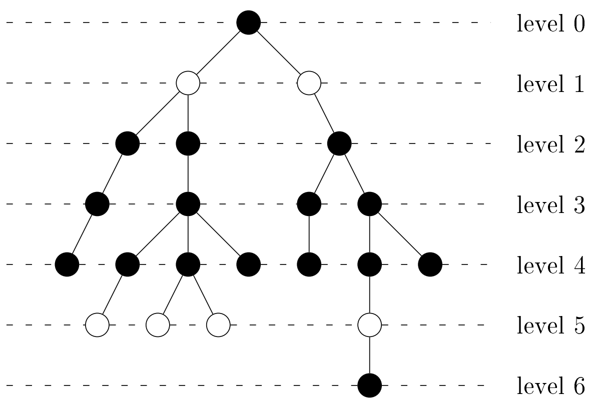


Figure 4.17: Example of the constructed code in a spanning tree

It is easy to see that $C$ is a dominating set. Indeed, every vertex $v$ has a neighbour $w$ in an adjacent level. Since no two adjacents levels are both out of $C$, at least one of the levels $level(v)$ and $level(w)$ is in $C$, and $v$ is dominated.

However, there may remain some unseparated vertices. Let $u$ and $v$ be to such vertices. Without loss of generality, suppose $level(u) \leq level(v)$ and let us distinguish the following case:

1. exactly one of $u$ and $v$ is in the code. Since $u$ and $v$ cannot share neighbours ($G$ has girth at least 5), all neighbours of $u$ and $v$ are not in the code. So one of $u$ and $v$ must be the root, and the other one must be a child of the root and a leaf at the same time (otherwise it would have children in the code). But this is not possible since we observed previously that there are no leaves at level 1 in $T$.

2. both $u$ and $v$ are not in the code. Then $u$ and $v$ are not connected. Indeed, otherwise they would be separated by their neighbours in $C$ since they would not share any neighbour. Moreover, $u$ and $v$ cannot be dominated by two distinct vertices $c_u$ and $c_v$ of $C$: if yes, since $u$ and $v$ are not separated, $u$ would be connected to $c_v$ and $v$ to $c_u$, and there would be a cycle of length 4 $u - c_u - v - c_v - u$ in $G$. So, $u$ and $v$ are two leaves having the same parent $p$. Since $G$ has minimum degree 2, both $u$ and $v$ are connected to at least one of their ancestors in $T$. But notice that they cannot be connected to the same ancestor $a$; indeed, since $u$ and $v$ are at distance 2, $G$ would contain the cycle of length 4 $u - p - v - a - u$. Hence, all neighbours of $u$ and $v$ other than $p$ are distinct ancestors which are not in the code (otherwise $u$ and $v$ would be separated by those ancestors). Let us just arbitrarily add one of these ancestors to $C$, and thus $u$ and $v$ become separated. If there are $N_l$ such unseparated leaves sharing the same parant, $N_l \geq 2$, we need to add $N_l - 1$ ancestors into the code. But since there are in total at least $2 \cdot N_l$ vertices out of the code, in the worst case we just divide the number of non-code vertices by two. See Figure 4.18 for an illustration.

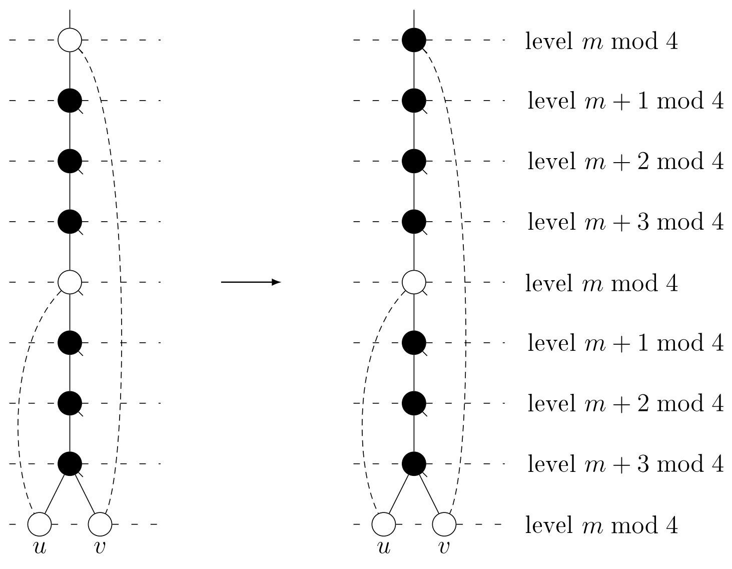


Figure 4.18: Two leaves $u$ and $v$ are unseparated by the code

3. both $u$ and $v$ are in $C$. Then $u$ and $v$ are connected (otherwise they would be

separated by themselves); since $G$ has girth at least 5, $u$ and $v$ do not share any neighbours. So, all neighbours of $u$ and $v$ are not in $C$. This can only happen in the follwing subcases:

- vertex $u$ is the root of $T$; $v$ is its only child in $T$, and $V_m = V_2$. Since this situation only happens once, we simply add one extra vertex $w$ to the code, arbitrarily chosen from the neighbours of $v$. This does not affect the separation of other vertices. See Figure 4.19 for an illustration.

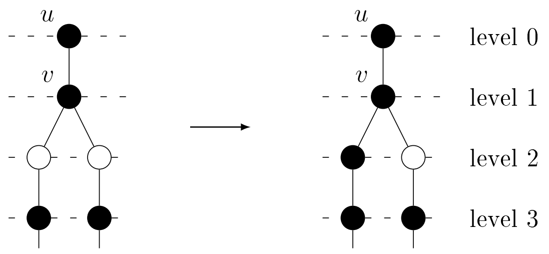


Figure 4.19: The root $u$ and its unique child $v$ are unseparated by the code

- vertex $u$ is the root of $T$; all its children are not in $C$ (therefore $V_m = V_1$), and $v$ is a leaf such that $level(v) = m + 1 \bmod 4$: its parent is out of the code. This case can only happen once since the root must be involved, and if the root is connected toanother such leaf, $u$ and $v$ would be separatedby it. Let us put an arbitrary child $w$ of the root into $C$ to separate $u$ and $v$. This does not harm other vertices' separation or domination. See Figure 4.20 for an illustration. Note that this case and the previous case cannot happen at the same time, so at most one vertex will be added to the code. Also note that these two cases must involve the root since two leaves of $T$ cannot be connected.

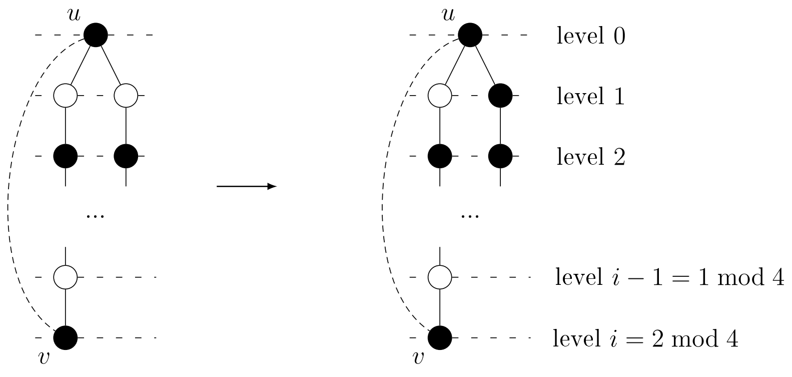


Figure 4.20: The root $u$ and a leaf $v$ are unseparated

- vertex $v$ is a leaf of $T$, $u$ is its parent and $level(u) = m+1 \bmod 4$. Vertex $u$ is not the root (otherwise, see the previous subcases). So, $u$ has a parent $w$ which is at level $m$ and therefore not in $C$. Let us simply take $w$ into $C$, and take $v$ out of $C$, without changing the cardinality of $C$. Then $u$ and $v$ are separated, all other vertices as well, and domination is maintained. See Figure 4.21 for an illustration.

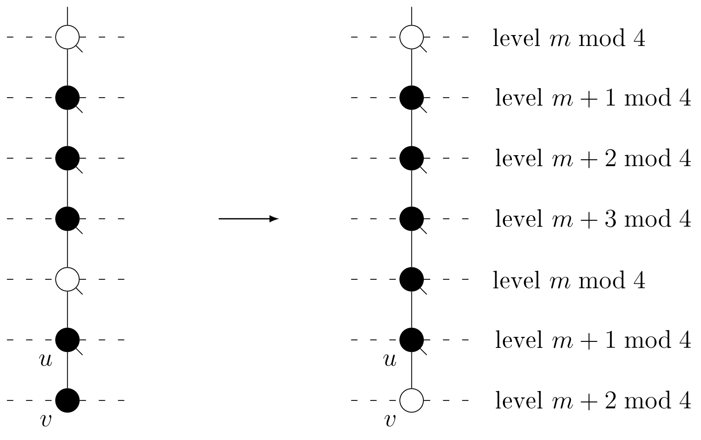


Figure 4.21: A leaf $v$ and its parent $u$ are not separated

So, because of case 2, we eventually end with a code of size at most $\frac{3n}{4} + \frac{n}{8} = \frac{7n}{8}$. Because of case 3, we eventually need to add one extra vertex to $C$.

So the size of the constructed code is at most $\frac{7n}{8} + 1$. □

Since any $\Delta$-reguler graph has minimum degree $\Delta$, the following corollary of Theorem 4.14 follows immediately:

**Corollary 4.15** *Let $G$ be a twin-free, $\Delta$-regular graph ($\Delta \geq 2$) on $n$ vertices. Then* $M_1(G) \leq \frac{7n}{8} + 1 = n - \frac{n-8}{8}$.

## 4.3 Subcubic and cubic graphs

We now discuss the question of an upper bound of identifying codes in subcubic graphs.

First, notice that there is no known subcubic graph for which a minimum identifying code has a size of more than $n - \dfrac{n}{\Delta} = \dfrac{2n}{3}$.
For an example of such a graph, see Figure 4.1 in section 4.1.

Let us apply the results of Theorems 4.3, 4.6, 4.10 and 4.13 to subcubic graphs:

**Corollary 4.16** *Let $G$ be a twin-free subcubic graph on $n$ vertices. Then:*

- $M_1(G) \leq \dfrac{101n}{102}$ (Theorem 4.3).
- *If $G$ is triangle-free, $M_1(G) \leq \dfrac{11n}{12}$* (Theorem 4.10).
- *If $G$ is cubic, $M_1(G) \leq \dfrac{8n}{9}$* (Theorem 4.6).
- *If $G$ is cubic and triangle-free, $M_1(G) \leq \dfrac{7n}{8}$* (Theorem 4.13).

However, we can improve these bounds in some subclasses of the class of subcubic graphs.

**Theorem 4.17** *Let $G$ be a Hamiltonian, twin-free, subcubic graph $n$ vertices. $G$ admits an identifying code $C$ such that $|C| \leq \left\lfloor \dfrac{3n}{4} \right\rfloor$.*

**Proof** Let $H = \{v_0, ..., v_{n-1}\}$ be an Hamiltonian cycle of $G$.

First, notice that there cannot be a vertex $v_i$ such that $v_i$ is connected to $v_{i+2}$, and $v_{i-1}$ is connected to $v_{i+1}$, since in this case, $v_i$ and $v_{i+1}$ would be twins (Figure 4.22).

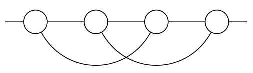


Figure 4.22: Forbidden situation in $G$: $v_i$ and $v_{i+1}$ are twins

Now, let us take as a code, the set $C = \{v_i \mid i \neq 0 \; mod \; 4\} \bigcup \{v_{n-3}, v_{n-2}, v_{n-1}\}$. We

have $|C| = \left\lceil \frac{3n}{4} \right\rceil$.
Obviously, $C$ is a dominating set, but there may be unseparated vertices. We will modify $C$ locally in order to achieve separation.
Let $v_i, v_j$ $(i < j)$ be two vertices of $G$ and suppose that they are not separated: $N[v_i] \oplus N[v_j] \subseteq V \backslash C$.

- $v_i$ and $v_j \in V \backslash C$. Then, since both of them have at least two neighbours in $C$, so in the worst case they are separated by two of them.
- $v_i$ and $v_j \in C$. Then $v_i$ and $v_j$ are connected (else they would be separated by themselves).
  - Suppose $v_j = v_{i+1}$. If $v_i$ and $v_j$ are of degree two, by construction of $C$ at least one of them has a neighbour in $C$ which separates them. So, either $v_i$ is connected to $v_{j+1}$ or $v_j$ is connected to $v_{i-1}$.
  - Then, $v_j = v_{i+2}$ because if not, their respective two neigbours on the Hamiltonian cycle would all not be in the code, which by construction of $C$ is not possible.
- $v_i \in C$ and $v_j \notin C$ or $v_i \notin C$ and $v_j \in C$. Then they are connected (otherwie they would be separated by $v_i$), and there exist at least two code vertices which dominate each of them. Since the configuration of Figure 4.22 is forbidden, they are separated by at least one code vertex.

So we see that there is one problematic situation which can appear several times in the graph, where $x, y, z$ are in $C$, $a, b$ are not, $xz \in E$ and $(v_i, v_j) \in \{(x, y), (x, z), (y, z)\}$. See Figure 4.23 for an illustration (code vertices are gray).

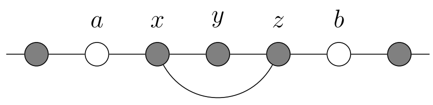


Figure 4.23: Situtation in which $C$ is not separating

We claim that in this case, it is always possible to modify locally the code to make it identifying without changing its cardinality. Let us present those modifications. First, notice that vertex $y$ can always be put out of $C$ safely.

1. suppose we deal with the configuration of Figure 4.23.
   (a) there is a connection between $a$ and its left neighbour at distance two. Then, we just put $a$ and $b$ into the code, and remove $y$ and the left neighbour of $a$ from it (Figure 4.24).

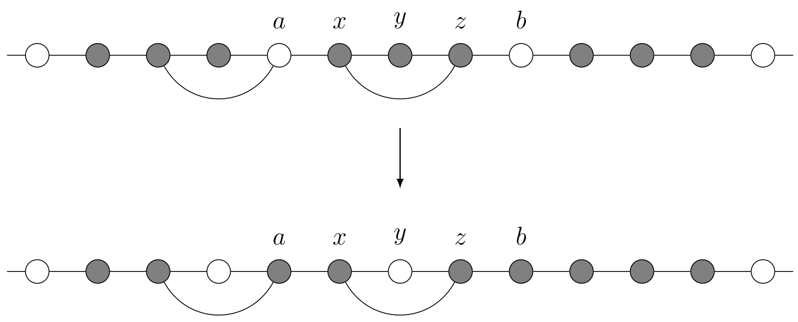


Figure 4.24: Local modification of the code (Case 1a)

(b) otherwise, we put $a$ and $b$ into $C$ but remove $x$ and $y$ from it (Figure 4.25).

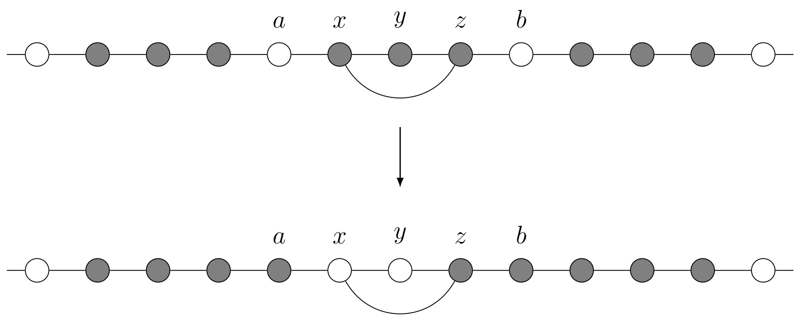


Figure 4.25: Local modification of the code (Case 1b)

2. suppose we are in the case where $a = v_0$ and $n = 4k + i$ for some $k \geq 2$ and $i = 1, 2$ or 3. Then we just put either $b$ or $a = v_0$ into the code, and put $y$ out of it (see Figure 4.26; in this example, $n = 4k + 1$ and $v_{n-1}$ is connected to $v_{n-3}$s).

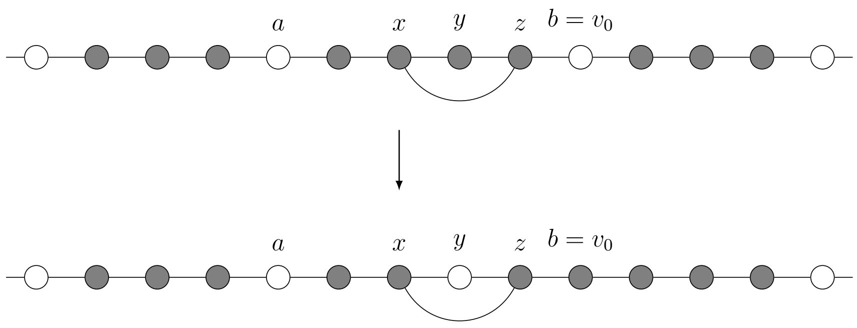


Figure 4.26: Local modification of the code (Case 2)

Notice that these local modifications do not affect nearby possible modifications.

One can see that the constructed code is a total dominating set, since all vertices (even code vertices) have a neighbour in the code. Moreover, almost all vertices have more than two code vertices in their closed neighbourhood. The only vertices for which this is not the case are not in the code, and their neighbour which is not in the code is connected to their other neighbour (construction of Figure 4.25).
So, if there is an edge between two vertices $v_i$ and $v_j$, $i > j + 2$, $v_i$ and $v_j$ are necessarily separated either by themselves or by one of their neighbours which is in $C$. So, $C$ is a valid identifying code.

We now claim that it is always possible to modify $C$ so that its size is at most $\left\lfloor \frac{3n}{4} \right\rfloor$.
To do so, it is sufficient to remove one single vertex from $C$. Consider two vertices $v_i$, $v_j = v_{i+4}$ which had initially been taken out of $C$, and such that they are not in the configuration depicted in Figure 4.23: $v_{i+1}$ is not connected to $v_{j-1}$. If there are no such vertices, by our modifications of the code notice that $C$ has a size of at most $\frac{2n}{3}$, and we are done.
Now, distinguish the following cases, by considering the edges $e_1 = \{v_{i-1}, v_{i+1}\}$ and $e_2 = \{v_{j-1}, v_{j+1}\}$.

1. If $e_1 \in E$ and $e_2 \notin E$, we can remove $v_{j-1}$ safely from $C$.
2. Similarly, if $e_1 \notin E$ and $e_2 \in E$, we can remove $v_{i+1}$ safely from $C$.
3. If either both or none of $e_1, e_2 \in E$, then $v_{i+2}$ can be removed from $C$.

Note that the property that $C$ is a total dominating set might not hold anymore if both $e_1$ and $e_2$ are not in $E$, and we took out vertex $v_{i+2}$ from $C$: then $v_{i+1}$ and $v_{j-1}$ are only dominated by themselves. But since we assumed that they are not connected to each other, this is not a problem.

So the constructed identifying code has a size of at most $\left\lfloor \frac{3n}{4} \right\rfloor$. □

Using the following Lemma of [KMOO02], the previous proof can be used to extend this result to another class of subcubic graphs.

**Lemma 4.18 ([KMOO02])** *Every 2-connected cubic graph has a 2-factor in which each component is a cycle of length at least four.*

**Corollary 4.19** *Every 2-connected, twin-free cubic graph $G$ on $n$ vertices has an identifying code $C$ such that $|C| \leq \frac{3n}{4}$.*

**Proof** Following Lemma 4.18, $G$ has a 2-factor in which each component is a cycle of length at least four. In other words, each component of the 2-factor is an Hamil-

tonian subcubic graph of at least 4 vertices.
Following Theorem 4.17, for every such component $C_i$, there is an identifying code of cardinality at most $\left\lfloor \frac{3|V(C_i)|}{4} \right\rfloor$. Since a vertex of a component can only be connected to one other vertex of another component, the union of the codes of all components is an identifying code of $G$ of size at most $\frac{3n}{4}$. $\square$

# Chapter 5

# Conclusion

As we saw in this thesis, there exists a strong relationship between the maximum degree $\Delta(G)$ of a graph $G$ and the minimum cardinality of an identifying code in this graph, $M_1(G)$. After having studied the lower bound of $M_1(G)$ depending on $\Delta(G)$ and investigated the case of two particular graph families, we were able to give a general upper bound of $M_1(G)$ depending on $\Delta(G)$. We could improve this bound for different subclasses of graphs such as triangle-free graphs, regular graphs and cubic graphs. Moreover, we saw that for a particular subclass of graphs (graphs of girth larger than 5 and minimum degree 2), the relationship between $M_1(G)$ and $\Delta(G)$ becomes less important. These results improve our knowledge on identifying codes, which are studied by a growing scientific community, but for which many problems remain open or unstudied. However, we were not able to prove or disprove our conjecture on the value of a general tight upper bound for $M_1(G)$ depending on the maximum degree. So, further work can be done in this direction, either by determining a tight upper bound, by restricting the problem to other graph classes such as planar graphs or bipartite graphs, or by using further graph parameters such as the average degree. Some techniques like probabilistic analysis of this problem, were not used in this thesis; it may be fruitful to use them in this context.

# Bibliography

[ACG$^+$99] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation.* Springer Verlag, 1999.

[Bak94] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41(1):153–180, 1994.

[BCHL04] N. Bertrand, I. Charon, O. Hudry, and A. Lobstein. Identifying and locating-dominating codes on chains and cycles. *European Journal of Combinatorics*, 25(7):969–987, 2004.

[BCHL05] N. Bertrand, I. Charon, O. Hudry, and A. Lobstein. 1-identifying codes on trees. *Australasian Journal of Combinatorics*, 31:21–35, 2005.

[BCM$^+$07] M. Blidia, M. Chellali, F. Maffray, J. Moncel, and A. Semri. Locating-domination and identifying codes in trees. *Australasian Journal of Combinatorics*, 39:219–232, 2007.

[Ber58] C. Berge. *Théorie des graphes et ses applications.* Collection Univesitaire des Mathématiques, Dunod, Paris, 1958.

[Ber84] A. A. Bertossi. Dominating sets for split and bipartite graphs. *Information Processing Letters*, 19(1):37–40, 1984.

[BHL05] Y. Ben-Haim and S. Litsyn. Exact minimum density of codes identifying vertices in the square grid. *SIAM Journal of Discrete Mathematics*, 19(1):69–82, 2005.

[BWLT06] T. Y. Berger-Wolf, M. Laifenfeld, and A. Trachtenberg. Identifying codes and the set cover problem. Proceedings of the 44th Annual Allerton Conference on Communication, Control and Computing, Monticello, USA, September 2006.

[CGH$^+$99] G. Cohen, S. Gravier, I. Honkala, A. Lobstein, M. Mollard, C. Payan, and G. Zémor. Improved identifying codes for the grid. *Electronical Journal of Combinatorics*, 1999. comment to 6(1):R19, 3 pp.

[CGH$^+$06] I. Charon, S. Gravier, O. Hudry, A. Lobstein, M. Mollard, and J. Moncel. A linear algorithm for minimum 1-identifying codes in oriented trees. *Discrete Applied Mathematics*, 154(8):1246–1253, 2006.

[CHHL07] I. Charon, I. S. Honkala, O. Hudry, and A. Lobstein. Structural properties of twin-free graphs. *Electronic Journal of Combinatorics*, 14(1), 2007.

[CHL02] I. Charon, O. Hudry, and A. Lobstein. Identifying and locating-dominating codes: NP-completeness results for directed graphs. *IEEE Transactions on Information Theory*, 48(8):2192, 2002.

[CHL03] 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*, 290(3):2109–2120, 2003.

[CHL05] I. Charon, O. Hudry, and A. Lobstein. Possible cardinalities for identifying codes in graphs. *Australasian Journal of Combinatorics*, 32:177–195, 2005.

[CHL07] I. Charon, O. Hudry, and A. Lobstein. Extremal cardinalities for identifying and locating-dominating codes in graphs. *Discrete Mathematics*, 307(3-5):356–366, 2007.

[CHLL97] G. Cohen, I. Honkala, S. Litsyn, and A. Lobstein. *Covering Codes.* Elsevier, Amsterdam, 1997.

[CHLZ01] G. Cohen, I. Honkala, A. Lobstein, and G. Zémor. On identifying codes. Vol. 56 of Proceedings of the DIMACS Workshop on Codes and Association Schemes '99, pages 97–109, 2001.

[CSS87] C. J. Colbourn, P. J. Slater, and L. K. Stewart. Locating–dominating sets in series-parallel networks. *Congressus Numerantium*, 56:135–162, 1987.

[dR94] J. de RUMEUR. *Communications dans les réseaux de processeurs.* Masson, Paris, 1994.

[FKKR09] F. Foucaud, R. Klasing, A. Kosowski, and A. Raspaud. Identifying codes and the maximum degree in triangle-free graphs. 2009. to be submitted.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences).* W. H. Freeman, 1979.

[GKM08] S. Gravier, R. Klasing, and J. Moncel. Hardness results and approximation algorithms for identifying codes and locating-dominating codes in graphs. *Algorithmic Operations Research*, 3(1):43–50, 2008.

[GM07] S. Gravier and J. Moncel. On graphs having a V$\setminus\{x\}$ set as an identifying code. *Discrete Mathematics*, 307(3-5):432 – 434, 2007.

[GMS06] S. Gravier, J. Moncel, and A. Semri. Identifying codes of cycles. *European Journal of Combinatorics*, 27(5):767–776, 2006.

[GVGN$^+$01] J. Gimbel, B. D. Van Gorden, M. Nicolescu, C. Umstead, and N. Vaiana. Location with dominating sets. *Congressus Numerantium*, 151:129–144, 2001.

[HHS98] T. W. Haynes, T. W. Hedetniemi, and P. J. Slater. *Fundamentals of Domination in Graphs.* Marcel Dekker, 1998.

[HIMR$^+$98] H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. NC-approximation schemes for NP-and PSPACE-hard problems for geometric graphs. *Journal of Algorithms*, 26:238–274, 1998.

[HKP$^+$05] J. Hromkovic, R. Klasing, A. Pelc, P. Ruzicka, and W. Unger. *Dissemination of Information in Communication Networks: Broadcasting, Gossiping, Leader Election, and Fault-Tolerance (Texts in Theoretical Computer Science. An EATCS Series)*. Springer, 1 edition, April 2005.

[Hro02] J. Hromkovic. *Algorithmics for Hard Problems*. Springer, 2nd edition, November 2002.

[Joh74] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[KCL98] 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*, 44:599–611, 1998.

[KKY80] Y. Kakuda, T. Kikuno, and N. Yoshida. The NP-completeness of the dominating set problem in cubic planar graphs. *Transactions of the Institute of Electronics and Communication Engineers of Japan*, E63(6):443–444, 1980.

[KMOO02] K.-i. Kawarabayashi, H. Matsuda, Y. Oda, and K. Ota. Path factors in cubic graphs. *Journal of Graph Theory*, 39(3):188–193, 2002.

[Lit09] S. Litsyn. A table of the best currently known lower and upper bounds on the smallest size of a covering code. `http://www.eng.tau.ac.il/~litsyn/tablecr/index.html`, 2009.

[Lob09] A. Lobstein. A bibliography on identifying, locating-dominating and discriminating codes in graphs. `http://www.infres.enst.fr/~lobstein/bibLOCDOMetID.html`, 2009.

[LY94] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5):960–981, 1994.

[Mon05a] J. Moncel. *Codes Identifiants dans les Graphes*. PhD thesis, Université Joseph Fourier – Grenoble I, 2005.

[Mon05b] J. Moncel. Optimal graphs for identification of vertices in networks. *Les cahiers Leibniz*, 138, 2005.

[Mon06a] J. Moncel. Monotonicity of the minimum cardinality of an identifying code in the hypercube. *Discrete Applied Mathematics*, 154(6):898–899, 2006.

[Mon06b] J. Moncel. On graphs on $n$ vertices having an identifying code of cardinality $\log_2(n+1)$. *Discrete Applied Mathematics*, 154(14):2032–2039, 2006.

[MS06] T. Müllmer and J. Sereni. Identifying and locating-dominating codes in (random) geometric networks. 2006. submitted.

[PV81] F. P. Preparata and J. Vuillemin. The cube-connected cycles: a versatile network for parallel computation. *Communications of the ACM*, 24(5):300–309, 1981.

[PY91] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425 – 440, 1991.

[RUDP$^+$03] S. Ray, R. Ungrangsi, F. De Pellegrini, A. Trachtenberg, and D. Starobinski. Robust location detection in emergency sensor networks. In *Proceedings of the IEEE INFOCOM*, pages 1044–1053, April 2003.

[Ska07] R. D. Skaggs. *Identifying vertices in graphs and digraphs.* PhD thesis, University of South Africa, 2007.

[Sla87] P. J. Slater. Domination and location in acyclic graphs. *Networks*, 17(1):55–64, 1987.

[SR84] P. J. Slater and D. F. Rall. On location–domination numbers for certain classes of graphs. *Congressus Numerantium*, 45:97–106, 1984.

[Suo07] J. Suomela. Approximability of identifying codes and locating–dominating codes. *Information Processing Letters*, 103(1):28–33, 2007.

[TXXH06] K. Thulasiraman, M. Xu, Y. Xiao, and X.-D. Hu. Vertex identifying codes for fault isolation in communication networks. Proceedings of the International Conference on Discrete Mathematics and Applications (ICDM 2006), Bangalore, 2006.

# Index of Definitions

# List of Symbols

| | |
|---|---|
| $(u,v)$ | An arc between two vertices $u$ and $v$ of a directed graph, page 5 |
| $\Delta(G)$ | The maximum degree of a graph $G$, page 5 |
| $\delta(G)$ | The minimum degree of a graph $G$, page 5 |
| $\gamma(G)$ | The minimum cardinality of a dominating set of a graph $G$, page 9 |
| $\gamma_L(G)$ | The minimum cardinality of a locating-dominating set of a graph $G$, page 9 |
| $\oplus$ | The symmetric difference between two sets, page 8 |
| $\overrightarrow{uv}$ | An arc between two vertices $u$ and $v$ of a directed graph, page 5 |
| $\{u,v\}$ | An edge between two vertices $u$ and $v$ in an undirected graph, page 5 |
| $A(G)$ | The arc set of a directed graph $G$, page 5 |
| $B_r(v)$ | The ball of radius $r$ of a vertex $v$, page 6 |
| $B_1^-(v)$ | The incoming ball of a vertex $v$ in a directed graph, page 12 |
| $BF_m$ | The butterly graph of dimension $m$, page 34 |
| $C_n$ | The cycle on $n$ vertices, page 13 |
| $CCC_m$ | The cube-connected cycles graph of dimension $m$, page 31 |
| $d_G(u,v)$ | The distance between two vertices $u$ and $v$ in a graph $G$, page 6 |
| $deg_G(v)$ | The degree of a vertex $v$ in a graph $G$, page 5 |
| $E(G)$ | The edge set of a graph $G$, page 5 |
| $g(G)$ | The girth of a graph $G$, page 7 |
| $G=(V,E)$ | A graph $G$ with vertex set $V$ and edge set $E$, page 5 |
| $G[S]$ | The subgraph of a graph $G$ induced by $S$, page 6 |
| $Gr_{\infty,\infty}$ | The infinite grid, page 16 |
| $H_m$ | The hypercube of dimension $m$, page 15 |
| $I_C(v)$ | The identifying set of a vertex $v$, page 8 |
| $I_C(X)$ | The identifying set of a set $X$ of vertices, page 11 |
| $K_n$ | The complete graph on $n$ vertices, page 6 |
| $K_{a,b}$ | The complete bipartite graph with bipartition sizes $a$ and $b$, page 52 |
| $M_1(G)$ | The minimum cardinality of an identifying code of a graph $G$, page 9 |
| $M_r(G)$ | The minimum cardinality of an $r$-identifying code of a graph $G$, page 11 |
| $n$ | The order of a graph, page 5 |
| $N(v)$ | The neighbourhood of a vertex $v$, page 6 |
| $N[v]$ | The closed neighbourhood of a vertex $v$, page 6 |
| $P_\infty$ | The infinite path, page 13 |
| $P_n$ | The path on $n$ vertices, page 13 |
| $T_k^h$ | The complete $k$-ary tree of $h$ levels, page 14 |
| $uv$ | An edge between two vertices $u$ and $v$ in an undirected graph, page 5 |
| $V(G)$ | The vertex set of a graph $G$, page 5 |