Universidad Complutense de Madrid

Facultad de Informática

Bachelor's Thesis in Computer Science

Double Degree in Mathematics and Computer Science

Analysis of Boolean functions using equanimity and entanglement

---

Análisis de funciones booleanas utilizando ecuanimidad y enredo

---

Authors:

Enrique Carro Garrido
José Ramón Cobián Fernández

Directors:

Ismael Rodríguez Laguna
Narciso Martí Oliet

Madrid, 29^{th} May 2023

Abstract

The study of Boolean functions has made limited progress in recent years. Despite significant efforts, fundamental questions regarding the complexity and behavior of Boolean functions remain unresolved. The lack of comprehensive mathematical models and techniques hinders advancements in this field, leaving many open questions and avenues for future research. In this work, we aim to expand our limited knowledge in this field by proposing and studying two notions that could help unravel when a Boolean function is computed by super-polynomial circuits.

The first notion is equanimity, which captures the condition where the output of a Boolean function is equally influenced by all input variables and/or subsets of the input. We will observe that this definition is not capable of classifying when a function cannot be computed by a polynomial circuit, although it remains a good indicator. Therefore, we need to introduce another concept, namely entanglement. This concept measures the amount of information needed from each subset of the input variables in order to obtain the output of the function.

Finally, we will provide empirical evidence that a highly entangled and highly equanimous function must necessarily be computed by large circuits, ideally super-polynomial ones. This could lead to a potential new approach to demonstrate that P $\neq$ NP.

Keywords

Boolean functions, circuit complexity, P vs. NP, $P_{/poly}$ class, lower bounds of circuits, equanimity, entanglement.

Resumen

El estudio de las funciones booleanas ha tenido un progreso limitado en los últimos años. A pesar de algunos resultados significativos, quedan todavía muchas preguntas fundamentales sobre la complejidad y el comportamiento de las funciones booleanas. La falta de modelos matemáticos completos y técnicas exhaustivas obstaculiza los avances en este campo, dejando muchas preguntas abiertas y vías para futuras investigaciones. En este trabajo, tenemos como objetivo ampliar nuestro conocimiento limitado en este campo al proponer y estudiar dos conceptos que podrían ayudar a desentrañar cuándo una función booleana es calculada por circuitos superpolinómicoss.

El primer concepto es el de ecuanimidad, que pretende capturar cuándo una función booleana es igualmente influenciada por todas las variables de entrada o subconjuntos de la entrada. Observaremos que esta definición no es capaz de clasificar cuándo una función no puede ser calculada por un circuito polinómico, aunque sigue siendo un buen indicador. Por lo tanto, necesitamos introducir el otro concepto, el de enredo. Este concepto mide la cantidad de información necesaria de cada subconjunto de las variables de entrada para obtener la salida de la función.

Finalmente, proporcionaremos evidencia empírica de que una función altamente enredada y ecuánime debe ser necesariamente computada por circuitos grandes, idealmente superpolinómicos. Esto podría conducir a un nuevo enfoque prometedor para demostrar que $P \neq NP$.

Palabras clave

Funciones booleanas, complejidad de circuitos, P vs NP, clase $P_{/poly}$, cotas inferiores de circuitos, ecuanimidad, enredo.

Contents

Chapter 1

Introduction

Boolean functions play a crucial role in the study of the P vs NP problem. However, in recent years, there has been a lack of significant advancements in the field of Boolean function research. Therefore, any contribution, no matter how small, would be valuable in our collective journey towards finding a solution to the millennium problem.

Some interesting progress was made by our former colleagues Jorge Villarrubia [VE21] and Celia Rubio [RM22] in their respective Bachelor's theses. The objective of our project is to continue expanding our knowledge on the topic via the study of two concepts that, to the best of our knowledge, are novel: equanimity and entanglement[1]. These properties will help us understand the inner working of functions and the circuits that compute them. In particular, we theorize that if a Boolean function is both equanimous and entangled, then it will inevitably be computed by a big (ideally superpolynomial-sized) circuit.

First of all, it is necessary to provide a formal definition of these abstract concepts to be able to operate with them. This process will involve brainstorming and identifying the key aspects of each concept, as well as evaluating their behavior within specific decision problems. This way, we will be able to adjust and perfect these definitions.

Then, we will test how they work by conducting tests on a set of functions computed by small circuits and another set of functions computed by larger circuits. This is motivated by our expectation that the metrics will yield distinct values depending on the circuit's size. In consequence, they will be useful to determine particular behaviors that prevent the circuits that compute a function from having polynomial size. Following that, we will analyze the results in order to determine if the metrics are indeed a good indicator. Moreover, additional experiments will be carried out to gather further information and insights.

Finally, the collection of all the results will help us determine the viability of our con-

[1]Our notion of entanglement will be somehow inspired by quantum entanglement from physics, although we are not aware of any applications of related ideas to the classification of Boolean functions.

jecture. Based on these findings, we will explain how this approach can be pursued in future works, ultimately bringing us closer to advancing our understanding of the P vs NP problem.

In the subsequent paragraphs, we will outline the structure of the report and provide an overview of the content covered in each chapter.

In Chapter 2, we provide a detailed overview of the background of this work. It formally describes the P vs NP problem and then outlines the main proof techniques that have been utilized since its conception in 1970. Finally, we focus on the technique that we consider the most promising but with very few advancements: Circuit complexity.

In Chapter 3, we present a developed framework of the methodology we propose and the specific objectives of this work, stating our conjecture regarding the relationship between circuit size and high values of equanimity and entanglement. We provide an "a priori" description of these abstract concepts. Then, we outline the decision problems of interest that we use as benchmarks for our metrics, and finally we list the various experiments we conducted to provide empirical support for the conjecture. In the following chapters, we focus on systematically developing this methodology step by step.

In Chapter 4, we study the concept of equanimity. The chapter is structured in such a way that we begin with an introduction to this abstract concept, followed by a formal definition of it. Within the same formalization, we explore two distinct approaches to the concept: one based on variable importance and another based on the survival of subsets of the input. Once the formal definition is provided, we proceed to discuss and analyze the cost of its different implementations. Finally, we present the limits of equanimity, setting the stage for the next chapter, which explains the concept of entanglement.

In Chapter 5, we follow the same structure as the previous chapter to present the concept of entanglement. We begin with an introduction to the concept, followed by its formalization. We then proceed to demonstrate certain properties that hold for this metric. Similar to equanimity, we provide the implementation of the metric and analyze its computational cost. The last section of this chapter is dedicated to arguing that entanglement alone may not function properly, thereby emphasizing the continued need for equanimity.

Chapter 6 presents the results of three experiments conducted to provide empirical support for our initial hypothesis. The first experiment involves testing the metric using the dataset provided by José Ramón Cobián [CF23]. In the second experiment, we explore the scalability of the metrics by examining how the metric values behave for the given benchmark problems as the input size increases. The final experiment focuses on testing the metrics using a specific type of circuits known as alternating circuits.

In Chapter 7, we will present our deductions and the results obtained from our studies.

We will be able to ultimately conclude whether the method we propose is suitable for our purpose in identifying complex Boolean functions. Finally, we will describe some future directions for our work with the ideal goal of providing useful insights for anyone attempting to ultimately solve the P vs NP problem.

Chapter 2

State of the art

The P vs NP question is central in theoretical computer science [Wig06]. The challenge of solving this problem, together with the need to understand the power and limits of computation, has led to the development of computational complexity theory. The discussion of progress includes diagonalization, algebrization, natural proofs barriers and other techniques [AS16]. This problem is considered one of the great problems of science.

Another interesting approach to face the P vs NP problem is obtaining circuit lower bounds, which has led to circuit complexity theory. It seems more than reasonable that problems should be analyzed by visualizing what is happening locally, which is the aim of circuit complexity theory. However, this area is usually overlooked and not given proper attention.

2.1. P vs NP Problem

With the aim of understanding this problem, we have to determine what P and NP classes are. First, we have to define the type problems that will be considered and what being efficiently computable means in this context.

Definition 2.1.1 (Decision problem)**.** *A decision problem is a problem that can be expressed as a yes-or-no answer of the input.*

A problem can be computed if there exists a set of rules that, followed in a particular way, may solve it. This fixed set of rules is called **algorithm**. In order to compute a problem, these rules could be applied multiple times, making the algorithm take more time. This is why we will measure computational efficiency as the number of basic operations an algorithm performs, which has to be a function of its input length.

In order to formalize this concept, not depending on the formalism chosen, Turing introduced Turing machines.

We're not going to go into detail about Turing machines, so we will introduce an informal definition to give an idea of what they are.

Definition 2.1.2 (Turing machines)**.** *[Tur36] A Turing machine is an abstract machine consisting of a one-dimensional infinite tape divided into cells, each capable of containing one symbol. It has a finite set of rules that determine how the machine should behave based on the current symbol being read and the machine's internal state. If there is only one possible transition from one state to another, we say the Turing machine is* ***deterministic****. Otherwise, we say it is* ***non-deterministic****.*

In order to group certain problems based on similarities on the way they are computed, complexity classes were created. There exists a lot of them, with a broad range of specificity, so we will introduce the most remarkable ones.

Definition 2.1.3 (The class P)**.** *The class P contains all the decision problems that can be computed in polynomial time by a deterministic Turing machine.*

Most known algorithms that solve problems in P run in $O(n^k)$ for a low value of k, so these problems are also called tractable.

Definition 2.1.4 (The class NP)**.** *The class NP contains all the decision problems that can be computed in polynomial time by a non-deterministic Turing machine.*

Alternatively, the class NP can be defined as the set of all the decision problems such that, if a certificate (a potential proof indicating a "yes" answer) of polynomial size is provided, then a deterministic Turing machine can verify in polynomial time whether the guess is correct or not.

For example, a certificate of the Hamiltonian path problem (find a path in a graph that visits each vertex exactly once) would be a sequence of vertices. In order to check if this solution is correct, we only need to check if those vertices actually form a Hamiltonian path.

Evidently, $P \subseteq NP$, since a solution can be verified in polynomial time by simply solving the problem. However, the proof or whether the other inclusion holds or not has not been found, and remains as the most important open problem in computer science.

The vast majority of complexity theorists believe $P \neq NP$, since the equality would imply that the class of the most difficult problems in NP, called NP-complete, would also be equal to P, i.e, solvable in polynomial time. To define this class, let's introduce some concepts first.

Definition 2.1.5 (Reduction, NP-Hard)**.** *A decision problem A is polynomial-time reducible to another decision problem B, noted $A \leqslant_p B$, if there exists a polynomial-time computable function γ such that $x \in A$ if and only if $\gamma(x) \in B$. A problem B is NP-hard if $A \leqslant_p B$ for every $A \in NP$, i.e, every problem in NP can be polynomial-time reducible to it.*

We are now in position to define NP-completeness.

Definition 2.1.6 (NP-complete)**.** *A decision problem is NP-complete if it is NP-hard and it belongs to NP.*

As we were mentioning, an NP-complete problem has a polynomial-time algorithm that solves it if and only if $P = NP$, and this would revolutionize the world as we know it. Not only in the field of Computational Complexity, but also any field that relies on the assumption that some problems are computationally hard, such as Cryptography, Artificial Intelligence or even Finance.

This is why an obvious way to tackle this problem is to show that a problem known to be NP-complete does not belong to P. This could be done in different ways, but in this text we will focus in one in particular: through the study of Boolean circuits.

2.2. Techniques to solve P vs NP

In the past, various approaches have been employed in attempts to solve the famous P vs NP problem. Here are some notable ones:

- Proof techniques: Researchers have explored different proof techniques such as diagonalization reduction, and contradiction to establish results related to P vs NP.
- Complexity classes: The study of various complexity classes, such as NP-complete problems, has played a crucial role in understanding the P vs NP problem. Researchers have focused on the structure and relationships between different complexity classes to gain insights into the nature of the problem.
- Oracle machines: Researchers have studied the power of hypothetical machines with access to an oracle for solving specific problems efficiently. By analyzing the limitations of such machines, they have attempted to shed light on the P vs NP problem.
- Approximation algorithms: Another approach has been to investigate the possibility of efficiently approximating NP-hard problems. Researchers have developed approximation algorithms that provide near-optimal solutions for certain problem instances, but not in a provably optimal manner.
- Structural properties: Exploring the structural properties of problems and the existence of efficient algorithms for specific problem classes has been another avenue of research.
- **Circuit complexity**: Techniques from circuit complexity theory have been employed to analyze the computational power of Boolean circuits and their relationship to P and NP.

The most common technique in the last 50 years has been diagonalization, which originates from Cantor's proof in 1891 and was later used by Turing to demonstrate the undecidability of the Halting problem. However, this proof technique alone is not sufficient to show that $P \neq NP$, as demonstrated by Baker, Gill, and Solovay [BGS75].

The problem with this technique is that it treats problem-solving machines as black boxes. The important aspect is to analyze what is happening at a local level. For this reason, we decided to focus on the study of Boolean circuits.

2.3. Circuit complexity

The purpose of the study of Circuit Complexity is to understand and measure the reasons why the minimum circuit that computes a Boolean function is either big or not in relation to the size of its input. To understand this statement, and explain why it is helpful to determine whether P is equal or not to NP, we have to introduce some concepts.

First of all, let's introduce and alternative way to represent decision problems, based on Boolean functions.

Definition 2.3.1 (Boolean function)**.** *A Boolean function is defined as* $f : \mathbb{Z}_2^* \longrightarrow \mathbb{Z}_2$*, i.e, a function that takes a binary input of any size and returns either 1 or 0. In particular, if the input size is n, we will denote it as* $f_n : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$*.*

Then, we can identify a Boolean function f with the set $\mathcal{L}_f = \{x : f(x) = 1\}$, which defines a decision problem. When there is no ambiguity with f we will just write $\mathcal{L}$. During this text, we will express such languages as $\mathcal{L} = \{x \in \mathbb{Z}_2^n : f_n(x) = 1\}_{n \geqslant 1}$ since we will study the decision problem for certain sizes n of the input.

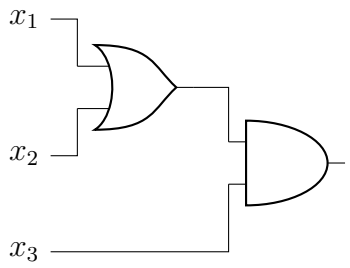
We need to establish a certain relation between decision problems and Circuit Complexity, but first we need to define the concept of Boolean circuit.

Definition 2.3.2 (Boolean circuit)**.** *A Boolean circuit* C *of* $n \in \mathbb{N}$ *inputs is a directed acyclic graph in which n nodes are sources and m are sinks, and the nodes in between are logical gates. The* ***size*** *of a circuit is the number of logical gates it contains. Given* $x \in Z_2^n$*, we denote the output of the circuit as* $C(x) \in \mathbb{Z}_2^m$*.*

From now on, we will consider Boolean circuits with $m = 1$ and fan-in 2, i.e, every gate has 2 inputs at most.

A circuit C of size n **computes** a Boolean function $f : \mathbb{Z}_2^n \longrightarrow Z_2$ if for every $x \in \mathbb{Z}_2^n$, $C(x) = f(x)$.

Example 2.3.1. *Let's consider the following circuit,*



which can be expressed as $C(x) = (x_1 \vee x_2) \wedge x_3$ *and has the associated truth table*

x1	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*
x2	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*0*
x3	*1*	*1*	*1*	*1*	*0*	*0*	*0*	*0*
$C(x)$	*1*	*1*	*1*	*0*	*0*	*0*	*0*	*0*

This circuit computes the function $f : \mathbb{Z}_2^3 \longrightarrow Z_2$ *such that f((1,1,1)) = 1, f((0,1,1)) = 1,..., f((0,0,0)) = 0.*

Observation 2.3.1. *A function can be computed by multiple circuits, so normally when we mention "the circuit that computes it" we will mean the minimum (one among the ones with the least size).*

We are now in position to understand how Boolean circuits and complexity of decision problems are connected. With this aim, a complexity class that contains the decision problems that can be computed by *small* circuits was created. Let's formalize this concept.

Definition 2.3.3 ($P_{/poly}$ class)**.** *A decision problem* $A \equiv \{x : f_n(x) = 1, f_n : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2\}_{n \geqslant 1}$ *belongs to* $P_{/poly}$ *if there exists a family of circuits* $\{C_n\}_{n \geqslant 1}$*, each of polynomial size with respect to n, such that* C_n *computes* f_n *for every n.*

Note that, for the same decision problem, the circuit that computes the function can be completely different for each size of input (the only requirement is for its size to be polynomial with respect to it).

As proved in [AB09], $P \subseteq P_{/poly}$. However, the other inclusion does not hold since $P_{/poly}$ even contains undecidable languages, such as recognizing the set of unary codes in $\{1\}^*$ representing a Turing machine not accepting its own code.

This is why by proving $NP \not\subseteq P_{/poly}$ we would prove $P \neq NP$. In order to show this, if is sufficient to find a decision problem A such that $A \in NP$ but $A \notin P_{/poly}$ (we will refer to this as finding superpolynomial lower bounds), since this would imply $A \notin P$. So, why is this approach reasonable?

First of all, let's introduce an interesting result.

Theorem 2.3.1. *[Sha49] For $n \geqslant 100$, almost all Boolean functions $f : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$ require a circuit of size at least $\frac{2^n}{10n}$ to be computed.*

Based on this result, the majority of functions require superpolynomial circuits. Finding just one of these functions and proving the corresponding decision problem belongs to NP would be sufficient to show $P \neq NP$.

Moreover, the study of circuits gives an insight of why things occur the way they do. Instead of treating it as a black box, we could understand and identify certain behaviors and characteristics that contribute to the size of a circuit.

There are some interesting results regarding circuit complexity, such as an exponential lower bound for monotone circuits — circuits with only OR and AND gates — computing the CLIQUE decision problem [RW93].

However, circuit complexity remains a highly open field. For instance, it is even an open problem to find a function in the complexity class NEXP — decision problems that can be solved by a non-deterministic Turing machine within exponential time — that it is not in $P_{/poly}$! Progress on general circuits has been almost nonexistent: a lower bound of n is trivial for any function that depends on all its input bits. The best circuit lower bound we can do after years is $5n - o(n)$ for any NP problem [AB09].

Chapter 3

Presentation of the new approach

In this chapter we will present the conjecture from which we start in order to tackle the problem P $\neq$ NP. The hypothesis is briefly explained in the first section. In the second section, we present some decision problems that are fundamental in complexity theory in order to test whether the conjecture seems true or not. Finally, in the last section we present the experiments we have carried out to give further support to the proposed conjecture.

3.1. Equanimity and entanglement

Our approach to face the $P \neq NP$ problem will be based on the general observation that functions requiring big circuits to be computed fulfill some properties, as well as some intuitive arguments explaining why small circuits should not be able to compute functions following these properties.

In particular, we will construct two concepts that together can empirically separate functions computed by big circuits from those with small ones. Our goal is to give a new angle to face this millennial quest supporting the conjecture with experimental results.

First, we start with the concept of equanimity, attempting to formalize the opposite of privileging some inputs over others, a property that is inherent in polynomial-sized circuits. Note that a polynomial-sized circuit cannot replicate what it locally does to some subset of the input bits to all the other subsets, because the number of gates would grow combinatorially. Hence, these circuits must privilege some combinations of bits over the others in the way gates treat them.

In order to measure this property on functions, we defined two different approaches that are different from each other but try to capture the same thing: when a function is equanimous. However, there is a counter example to this intuition, the PARITY problem, a problem in $P_{/poly}$ which is very equanimous, showing the need to introduce a new

concept, entanglement.

Secondly, we present the entanglement concept as a way to try to separate every NP problem which is equanimous from the factors that make PARITY equanimous. This is, although PARITY is very equanimous, all the inputs of the optimal polynomial-sized circuit that computes it are only *directly* affecting a small region of that circuit. We suspect this is exactly what lets PARITY be computed by a small circuit despite its high equanimity. However, if we demand the opposite, namely that the function entangles its inputs and still maintains equanimity, then we suspect this function will require a big, ideally super polynomial, circuit to be computed.

To sum up, our conjecture states that every function that is equanimous and entangled can only be computed by circuits of super polynomial size (i.e., it cannot belong to $P_{/poly}$). We believe that a super polynomial lower bound could be obtained for circuits that satisfy certain equanimity and entanglement criteria. In this scenario, P $\neq$ NP would be immediately proved by finding an NP-Complete problem that satisfies these conditions (since, as demonstrated earlier, $P \subset P_{/poly}$). In fact, we have found a problem that we will define later, which is NP-complete and seems to be very equanimous and entangled: CLIQUE′.

Observation 3.1.1. *A **natural proof** is one that shows the circuit size of any sequence of functions that satisfy certain natural property is superpolynomial. In order to be natural, a property must satisfy two conditions: it can be computable in time that is polynomial in the truth table of f_n* [1] *(constructivity) and it is satisfied by a sufficiently large fraction of the set of all Boolean function (largeness).*

As shown by Razborov [RR94], no strategy along these lines can ever succeed to prove $P \neq NP$. Such a proof would imply the existence of an algorithm capable of breaking a pseudorandom generator (essential in Cryptography), which is widely believed to be impossible.

As we will show during the following chapters, our metrics satisfy the constructivity condition. However, being both highly equanimous and entangled at the same time is feasible for a small amount of functions, so we do not expect our criterion to satisfy the largeness condition and the property would not be natural.

3.2. Decision problems of interest

In order to test the metrics described in the following sections and check if these definitions are able to capture the desired behavior, we have chosen some functions with instructive properties. These functions will be used to compare their metric values with those of a

[1] Note that the size of the truth table of a Boolean function is exponential with the number of input variables

larger set of functions at a later stage.

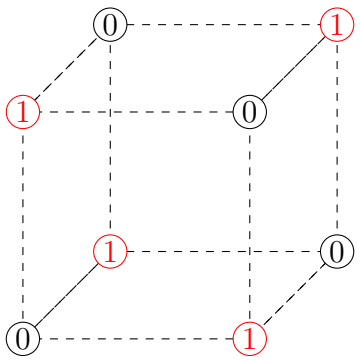
3.2.1. Problems known to be in $P_{/poly}$

A problem usually studied in complexity theory is the language PARITY because it has an unusual property: it has a small circuit although it has the most complex DNF (Disjunctive Normal Form) and CNF (Conjunctive Normal Form).

Definition 3.2.1 (PARITY Problem)**.** *Given a binary input, it determines whether the number of ones is even or not. It can be represented as the family of functions* $\{p_n : n \in \mathbb{N}\}$*, where*

$$p_n : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$$
$$(x_1, ..., x_n) \mapsto |\{i : x_i = 1, 1 \leqslant i \leqslant n\}| = 0 \pmod 2$$

The n-variable PARITY function and its negation are the only Boolean functions for which all disjunctive normal forms have the maximal number of 2^{n-1} monomials of length n and all conjunctive normal forms have the maximal number of 2^{n-1} clauses of length n. The reason why this is happening is for the hypercube representation in the Karnaugh procedure.



That is, each vertex is connected to others that have the opposite value, so that when applying the Karnaugh map algorithm, the most complex DNF and CNF possible is obtained.

In fact, from the Håstad's Switching Lemma [Hå14], PARITY $\notin AC^0$, which means it cannot by computed by a polynomial-sized, constant-depth circuit with AND, OR and NOT gates of unbounded fan-in. From this point of view, PARITY may seem very complex; however, PARITY is in P, and there is a simple circuit that computes it[2]. Because of this, if our metrics didn't reflect PARITY function's simplicity, they would not be adequate.

In order to prove this result, Håstad's introduced a certain form of expressing Boolean circuits, called Alternating Circuit (AC) Form. This form is useful since it is easier to

[2] $p_n(x) = x_1 \oplus x_2 \oplus ... \oplus x_n$, where $\oplus$ denotes the exclusive or (XOR).

visualize and to understand, and we will use it to explain some concepts and to carry out experiments.

Definition 3.2.2 (AC form)**.** *A Boolean circuit has AC form if:*

- *Level 0 is composed solely of the NOT gate and the identity*
- *Inputs of gates in level $i > 0$ come from level i-1*
- *The depth (d) — which is the number of levels — is odd*
- *Gates at level $i > 0$ function as AND gates when i is odd, and as OR gates otherwise.*

Observation 3.2.1. *If a Boolean function of n inputs can be computed by a circuit C with size s(n) and depth d(n), it can be demonstrated that the same function can also be computed by a circuit C' in AC form. The size and depth of C' can be obtained by polynomial transformations of s(n) and d(n).*

The problem of finding whether a number is prime or not has been around for a long time. It was suspected to be in the NP-Intermediate class of decision problems — neither in the class P nor NP-complete but still in NP. However, in 2002, it was proven that it's a problem in P by the AKS polynomial decision method, created from the Number Theory field in Mathematics. Let's enunciate this problem for a binary input:

Definition 3.2.3 (PRIMALITY Problem)**.** *Given a binary input, it determines if the corresponding number is prime or not. It can be represented as the family of functions $\{P_n : n \in \mathbb{N}\}$, where*

$$P_n : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$$
$$(x_1, ..., x_n) \mapsto y \text{ is prime, where } y = \sum_{i=1}^{n} x_i * 2^{i-1}$$

Since this problem is in P, our metrics should classify it as "simple." Note that our metrics consider the best known algorithm, so they could be helpful to determine if problems formerly thought to be NP-intermediate, but being in P, are properly classified by our metrics.

Another interesting problem that is in P is the MAJORITY decision problem. Let's define this problem.

Notation 3.2.1. *Given $\vec{x} \in \mathbb{Z}_2^n$, we denote $ones(\vec{x}) = |\{i : 1 \leqslant i \leqslant n, \vec{x}_i = 1\}|$.*

Definition 3.2.4 (MAJORITY Problem)**.** *Given a binary input, it determines if the number of ones is greater than or equal to the number of zeros. It can be represented as the family of functions $\{M_n : n \in \mathbb{N}\}$, where*

$$M_n : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$$
$$\vec{x} \mapsto ones(\vec{x}) \geqslant \left\lceil \frac{n}{2} \right\rceil$$

Intuitively, every circuit that computes this function must be somehow biased, given that when the number of ones is already greater than or equal to $\lceil \frac{n}{2} \rceil$, the information provided by the rest of the inputs is irrelevant. Hence, the function propensity to return 1 is heavily lowered (or raised) when it is conditioned to some known input bits.

An objective that aims to avoid our metrics is to make them capable of distinguishing when a problem depends on "what" rather than "how many." MAJORITY is a clear example of a problem that, similar to PARITY, depends solely on "how many."

3.2.2. CLIQUE Decision Problems

Once we have seen how the metrics behave with problems from the $P_{/poly}$ class, we want to find an NP-complete problem that exhibits high values of equanimity and entanglement, since, if the conjecture were true, this problem would not be in $P_{/poly}$.

If we want a problem whose value depends on "what" rather than "how many", then the CLIQUE problem is indeed a suitable example. In the CLIQUE problem, each input contributes valuable information to the rest, and it is virtually impossible to find a subset of input bits having a trivial dependence on the remaining bits. This characteristic suggests that the CLIQUE function is highly entangled. Besides, since CLIQUE is isomorphism-equivalent, its fairness (i.e., equanimity) should be relatively high.

A well-known NP-complete problem easy to implement as a binary function is the Half CLIQUE Problem. In order to define it, we need to see first what a CLIQUE of size $k \in \mathbb{N}$ is.

Definition 3.2.5 (k-CLIQUE)**.** *Let $G = (V, E)$ be an undirected graph, where V is the set of vertices and E is the set of edges between those vertices, and let $k \in \mathbb{N}$. A k-CLIQUE is a subgraph $S = (V', E')$ with $|V'| = k$ such that S is complete, i.e., each pair of vertices $u, v \in V'$ are connected by an edge $(u, v) \in E'$.*

As we know, to show that a problem is NP-hard, we need to see if an NP-hard problem can be reduced in polynomial time to it. Naturally, we will choose the CLIQUE Decision Problem, which we know is NP-complete.

Definition 3.2.6 (CLIQUE Decision Problem)**.** *Given an undirected graph $G = (V, E)$ and $k \in \mathbb{N}$, the problem consists in deciding whether G contains a k-CLIQUE or not.*

We are now in the position to define the Half CLIQUE Problem and to show it is an NP-complete problem.

Definition 3.2.7 (Half CLIQUE Problem)**.** *Particular case of CLIQUE Decision Problem where $k = \frac{|V|}{2}$.*

Lemma 3.2.1. *Half CLIQUE is NP-Complete.*

Proof: A problem is NP-Complete if and only if it is in NP and it is NP-Hard. Therefore, the proof for the statement consists of two parts:

Half CLIQUE is in NP. Given a subset V' of vertices as a possible solution of the problem, we can validate this solution by verifying if all vertices of V' share an edge with each other. This can be done in $O(V + E)$, thus polynomial.

Half CLIQUE is NP Hard. Let´s show the CLIQUE problem can be reduced to the Half CLIQUE problem. We consider an instance of the clique problem, being $G = (V, E)$ an undirected graph and $k \in \mathbb{N}$, and we convert it into $G' = (V', E')$ to be the instance of the Half clique problem. There are two scenarios to be considered:

- $k \geqslant \frac{|V|}{2}$. Let's take $V' = V \cup I$ and $E' = E$, where I is a set of vertices each of degree 0. We can take $|I| = 2k - |V|$ to ensure $k = \frac{|V'|}{2}$. Therefore, G has a clique of size k if and only if G' has a clique of size k.
- $k < \frac{|V|}{2}$. Let's take $V' = V \cup M$ and $E' = E \cup F$, where M is a set of vertices which are connected to every other vertex in V' by edges in F. We pick $|M|$ such that $|M| = |V| - 2k$, so $k' := k + |M| = \frac{|V'|}{2}$. Therefore, G has a clique of size k if and only if G' has a clique of size k'.

□

Since the Half CLIQUE problem is computationally easier, we will use it for our experiments. From now on, when we mention the CLIQUE in the context of our experiments, we will be referring to the Half CLIQUE problem. Let's define it given a binary input.

Definition 3.2.8 (CLIQUE Problem)**.** *Given a binary input representing the edges of an undirected acyclic graph, it returns 1 if that graph contains a Half Clique. We will denote the family of functions as* $\{C_n : n \in \mathbb{N}\}$.

However, through the experiments we will introduce in Chapter 6, we observed that the previous problem was not very unbiased (i.e., equanimous). This does not contradict our conjecture, as a function requiring super-polynomial circuit does not need to be equanimous and entangled (the implication of the conjecture does not go this way). The goal is to find a NP-complete problem which has high values for both metrics, and therefore, if the conjecture were really true, it would necessarily require a super polynomial circuit to be computed.

What is wrong about the Half CLIQUE Problem is its equanimity. Therefore, since PARITY is highly even, we decided to define a new problem CLIQUE′, which combines both Half CLIQUE and PARITY decision problems.

Definition 3.2.9 (CLIQUE′ Decision Problem)**.**

- *Input: Undirected graph* $G = (V, E)$.

- *Output: Reply yes if and only if:*
  1. *G contains a CLIQUE of size* $\frac{|V|}{2}$ *and* $|E|$ *is even.*
  2. *G doesn't contain a CLIQUE of size* $\frac{|V|}{2}$ *and* $|E|$ *is odd.*

Similar to the definition of the CLIQUE Problem, we identify as $\{C'_n : n \in \mathbb{N}\}$ the family of functions that receive a binary input of size n and return the output of the CLIQUE′ decision problem.

As we will see later, very high values are obtained in both metrics by this problem. Let's see it is a NP-complete problem.

Theorem 3.2.1. *CLIQUE′ is NP-Complete*

Proof: We check both conditions for NP-Completeness.

CLIQUE′ is in NP

Certificate: Let $S = (V', E')$ be a subgraph of $G = (V, E)$. Note that the construction of a certificate S only needs to define a subset of V, V', which can be done in non-deterministic polynomial time.

Validate: To verify the correctness of the certificate, we need to validate the following predicates first:

1. $C := S$ is a complete subgraph with $|V'| = \frac{|V|}{2} \iff \forall (u,v) \in V', (u,v) \in E' \wedge |V'| = \frac{|V|}{2} \iff (u,v) \in V', G[u][v] = 1$ (if G is implemented with an adjacency matrix) $\wedge |V'| = \frac{|V|}{2}$. This can be done in time $O(|V'|^2) = O((\frac{|V|}{2})^2)$.
2. $P := |E|$ is even $\iff P = 0 \; mod \; 2$. This can be tested in constant time.

The certificate is correct if and only if $(P \wedge C) \vee (\neg P \vee \neg C)$. As a summary, we have obtained a non-deterministic polynomial algorithm that solves the problem and therefore CLIQUE′ is in NP.

CLIQUE′ is NP-Hard if every algorithm in NP can be reduced in polynomial time to CLIQUE′. As it was seen in Lemma 3.2.1, the Half CLIQUE Problem is NP-Hard, therefore every problem in NP can be reduced to the Half CLIQUE Problem in polynomial time. Thus, if the Half CLIQUE Problem is reducible in polynomial time to CLIQUE′, then every NP Problem can be reduced to CLIQUE′ in polynomial time. It can be shown that the latter is true.

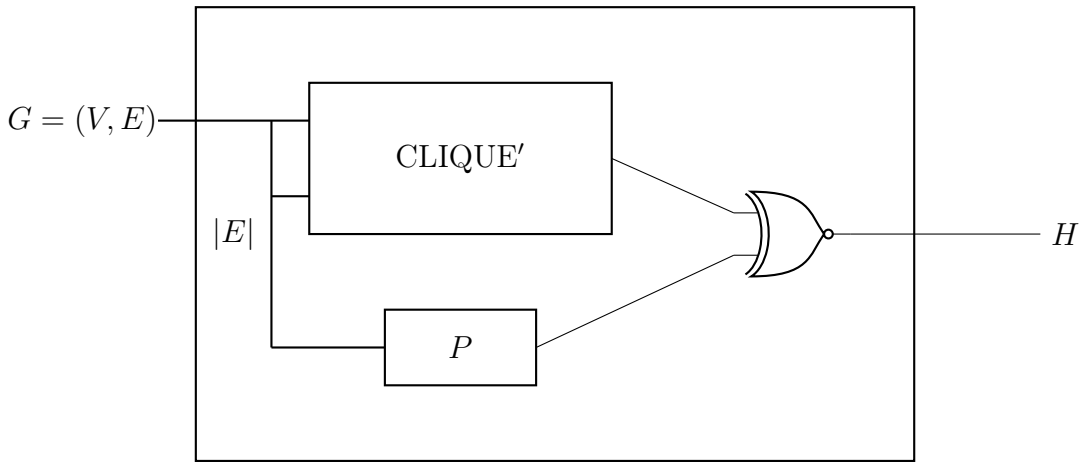
We will present a polynomial-time Turing reduction between these problems (alternatively, a many-one polynomial-time reduction could also be constructed, in particular by adding

two nodes isolated from the rest but connected with each other with an edge when |E| is odd).

Using the same notation for P as before, let H be the true/false answer of the Half CLIQUE Problem for $G = (V, E)$, it can be shown with first-order logic that P NXOR $CLIQUE' \equiv H$:

$$
\begin{aligned}
&\neg(P \oplus CLIQUE') \equiv \\
&(P \wedge CLIQUE') \vee (\neg P \wedge \neg CLIQUE') \equiv \\
&(P \wedge ((H \wedge P) \vee (\neg H \wedge \neg P))) \vee (\neg P \wedge \neg((H \wedge P) \vee (\neg H \wedge \neg P))) \equiv \\
&(P \wedge H) \vee (\neg P \wedge H) \equiv \\
&(P \vee \neg P) \wedge H \equiv \\
&\top \wedge H \equiv H
\end{aligned}
$$

To sum up, this can be represented with the following scheme:



That is, the Half CLIQUE Problem can be reduced to CLIQUE′ by testing if the number of vertices is even and with an NXOR gate, therefore, in polynomial time. □

3.3. Experiments to support the conjecture

The purpose of this section is to simply present all the different experiments that have been carried out with the aim of providing empirical support for the concept. The development and presentation of the results will be explained later on in Chapter 6.

The first experiment consists of drawing conclusions about the **dataset** of José Ramón Cobián [CF23]. This dataset contains exactly the Boolean functions from five bits to one bit which can be computed by circuits of 10 or less NAND gates. We will compare this dataset with a large sample of 5-bit Boolean functions randomly selected, and see

if it is possible to predict whether they will belong to this dataset or not (i.e., whether they can be computed by particularly small circuits). We are interested in obtaining percentiles that completely separate both sets, as these percentiles could provide us with more information about the aforementioned lower bound sought earlier.

Secondly, we want to provide more information about these percentiles, since working only with 5-bit Boolean functions, the information obtained from the previous experiment will obviously not be asymptotic, that is, it will not reflect the general tendency as the number of input bits grows. In other words, we want to observe the **scalability of the metrics**. This experiment consists of taking a small sample of N-bit Boolean functions, which, according to Theorem 2.3.1, are unlikely to belong to $P_{/poly}$, in order to observe the values of the metrics they possess. Moreover, with this experiment, a more detailed comparison could be provided regarding where the mentioned problems of interest, mentioned in the previous section, are encountered.

The last experiment is divided into two parts, studying a particular case of a circuit called **"Alternate circuits"** (defined in 3.2.2) by requiring that $x_1 \wedge x_2$ always appears at the first level, while $x_3 \wedge x_4$ never appears, and all circuits have a size limit of 50 gates.

- In the first part, we observe what happens when we enforce that $x_1 \wedge x_2$ only affects one branch of the circuit. The goal is to find functions that are able to be equanimous at least between $\{x_1, x_2\}$ and $\{x_3, x_4\}$, i.e., to find functions that have a similar behavior to PARITY.
- On the other hand, if we "poison" the entire second level with the output of gate $x_1 \wedge x_2$, then it is impossible for the function not to be biased in favor of $x_1 \wedge x_2$. This supports the conjecture that a function computed by a small and tangled circuit cannot be equanimous.

With these experiments, we aim to convince researchers in this field that the conjecture seems to be true, with the hope that someone may use it to finally solve the Millennium Problem. Both the code and the data can be found in the repository [CC23].

Chapter 4

Equanimity

4.1. Introduction to the concept of equanimity

It is more than reasonable to think that a Boolean function of polynomial size, by the mere fact of being it, inevitably is biased. This resulted in us having to measure when a Boolean circuit is unbiased, what we called equanimity. Our conjecture states that a problem that is in $P_{/poly}$ should have low values of equanimity for every Boolean function of its family, as every circuit that computes these functions should be biased.

A good metaphor of this unbiased situation on a circuit could be to think of a tightrope walker crossing a thin line while holding a pole. All inputs in a circuit should have the same effect on the output of an equanimous circuit, otherwise the output falls into one region of the inputs. A similar thing happens with the tightrope walker: if he does not calibrate the weights on each side of the pole he would fall into one side of the line.

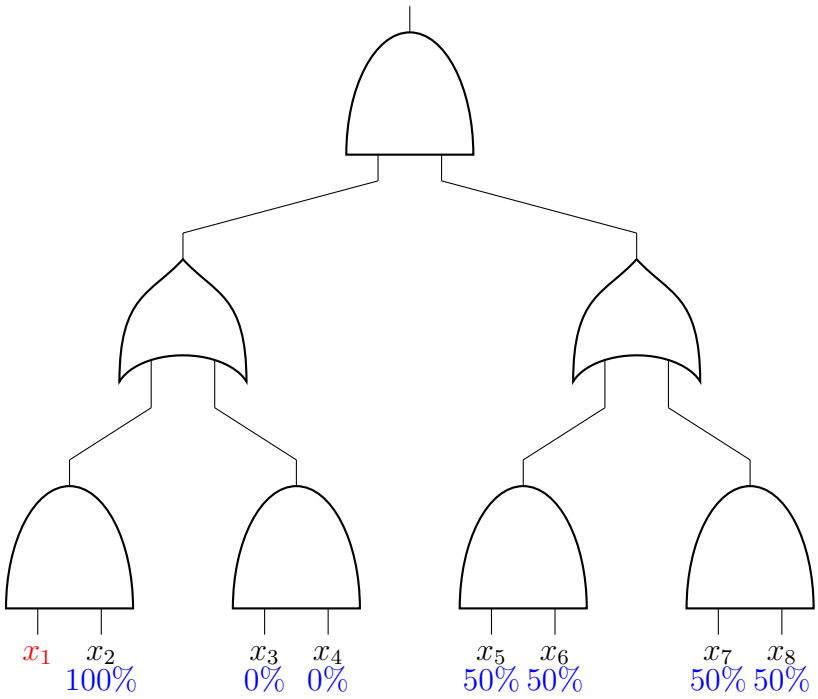
In order to visualize this assumption that polynomial circuits bias unavoidably, let's see the following example.

Example 4.1.1. *The variable x_1 in an unbiased circuit should have an effect on every other variable equally. However, if the input x_1 affects only a small region of the circuit, then the effect on the variables that are near it through the evaluation of the circuit is more substantial than the one in further variables. Here we are understanding the concept of distance as the level where both inputs intersect.*

Note that this phenomenon also depends on the type of gates where the inputs intersect. The goal of an unbiased circuit is to have the ability of balancing the inputs in a way where all of them have the same impact on its final value (that is, whether the output of the circuit is 0 or 1).

A simple way to comprehend this abstract concept is by comparing how x_1 is "accompanied" along the circuit by each different variable as it is shown in the figure below, which

represents a circuit in AC form (defined in 3.2.2).



For each wire in a circuit, we can consider the Boolean function it computes. If we see these functions in DNF, then the wires at the bottom compute the trivial DNFs x_1, x_2, etc. Firstly, the set of conjunctive clauses computed by an AND gate is the Cartesian product of the set of clauses of its inputs (removing all clauses with contradictions, i.e., $x_i \wedge \neg x_i$). Secondly, the set of clauses computed by an OR gate is the union of the sets of its inputs.

Since the AND gate in the left is the only gate where x_1 AND x_2 intersect, this forces x_1 to always be accompanied by x_2. On the other hand, the OR gate just above has a different behavior, as it forces x_1 to never be accompanied by x_3 or x_4 (the set of clauses in this case is $\{x_1 \wedge x_2, x_3 \wedge x_4\}$). The other percentages below x_5, x_6, x_7, x_8 are reasoned by a similar argument (we just have to see how the Cartesian product applies to the set of clauses received at the top OR gate). As we can see, this behavior, inherent in every polynomial circuit, produces a bias in the function computed by the circuit.

For instance, a simple way to balance the difference between x_1 and (x_3, x_4) is by including new AND gates to join the latter inputs, making the number of gates increase. Fixing the bias this way is gates-consuming, and a polynomial-sized circuit cannot afford to include a subcircuit specially dedicated to deal with each possible combination of 2 inputs, 3 inputs, and so on (since the number of subcircuits would not be polynomial). Intuitively, all circuits of polynomial size are condemned to privilege some subsets of inputs over others during their computation. This is why it seems reasonable to see equanimity as a metric that identifies super-polynomial circuits.

Since the property of equanimity must be maintained in all circuits that compute the same function, we should analyze when a Boolean function is unbiased.

A Boolean function is biased if its output is strongly determined by a certain subset of the input rather than the others. For example, the Boolean function $f_n(x) = x_2$ is biased towards x_2, since only those subsets of the input that contain x_2 are relevant to the output.

On the contrary, a function is unbiased if every combination of the input is relevant to the output. Intuitively, if the number of connections between inputs increases, the number of gates of the Boolean circuit that computes the functions will increase (since "connecting inputs" means linking them with a gate). This is why we theorize the following: "Being unbiased is only within reach of bigger circuits."

During this section, we will formalize this concept in order to measure bias in functions and we will analyze if the previous statement holds or not. There are two different ways of approaching the formalization of this concept, so we will introduce them and explain why both are interesting.

4.2. Equanimity based on the importance of each variable

In order to be unbiased, all variables in a Boolean function should be considered equally, i.e, every variable is as "important" as the others. Let's formalize what we mean by important.

Definition 4.2.1 (Important variable)**.** *Let $f : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$ be a Boolean function. The $i-th$ variable is important to f if and only if*

$$\exists\, \vec{x},\ \vec{y} \in \mathbb{Z}_2^n :\ f(\vec{x}) \neq f(\vec{y}),\ \textit{with}\ x_i = 1 \wedge y_i = 0 \wedge\ \forall j \neq i,\ x_j = y_j$$

Note that we don't only have to know whether a variable is important or not, but also in how many combinations it is important.

Definition 4.2.2 (Degree of importance)**.** *Let $f : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$ be a Boolean function. The degree of importance of the i-th variable, $I(i)$, is the number of different combinations in which the output of f depends on whether $x_i = 0$ or $x_i = 1$. Formally,*

$$I(i) := |\{(\vec{x}, \vec{y}) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n : (x_i, y_i) = (1, 0) \wedge\ \forall j \neq i,\ x_j = y_j \wedge f(\vec{x}) \neq f(\vec{y})\}|$$

Note that if $I(i) > 0$ then the i-th variable is important.

Let's see an example to understand this concept better and to show how to calculate it.

Example 4.2.1. *Let $f_3(x) = x_1 \vee x_2$. This function has the following truth table associated:*

x_1	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*
x_2	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*0*
x_3	*1*	*1*	*1*	*1*	*0*	*0*	*0*	*0*
$x_1 \wedge x_2$	*1*	*1*	*1*	*0*	*1*	*1*	*1*	*0*

Let's calculate how many times is each variable important. In order to do this, we have to consider every possible combination of the other inputs, and see how changing the value of the variable affects the output.

Let's begin with the variable x_1. If $x_2 = 1$ and $x_3 = 1$, we consider $\vec{x} = (0, 1, 1)$ and $\vec{y} = (1, 1, 1)$. Since $f(\vec{x}) = f(\vec{y})$, the variable x_1 is not important in this context.

By the same method, we obtain this variable is only important when $x_2 = 0$ and $x_3 = 1$ and when $x_2 = 0$ and $x_3 = 0$, so $I(1) = 2$. Symmetrically, $I(2) = 2$.

It is easy to see that the variable x_3 is never important, so $I(3) = 0$.

It could be thought it is enough to see if $I(i)$ is similar for every variable i to determine whether a function is biased or not. However, this is not entirely accurate.

Example 4.2.2. *Let's consider the Boolean function*

$$f_n(x) = \bigwedge_{i=0}^{n-1} x_i$$

In this case, $I(i) = 1\ \forall i \in \{1, ..., n\}$. However, this function is totally biased towards the combinations of the input that don't have a 0 in them, since having a 0 immediately implies the output will be 0. This is why we also have to consider the value of $I(i)$ itself.

In short, a function is totally unbiased if every variable is important for every combination, i.e, the sum of the degrees of importance of each variable. Note that our initial conjecture states that all circuits computing functions that have high values of equanimity should be big. If a substantial amount of variables have low degrees of importance, the equanimity will be greatly reduced. All of these reflections lead us to the following definition of the metric.

Definition 4.2.3 (Equanimity based on the importance of each variable)**.** *Given $f : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$, we define its equanimity as*

$$\mathcal{Q}_I(f) = \frac{\sum_{x=1}^{n} I(i)}{n * 2^{n-1}}$$

The denominator in the expression above is used to normalize the values so $Q(f) \in [0, 1]$, where 0 means totally biased and 1 means totally unbiased.

4.2.1. Implementation for $\mathcal{Q}_I$

The following function takes a Boolean function of n input bits f_n as an input and returns $\mathcal{Q}_I(f_n)$

```
double equanimity_importance(const vector<int>& f, const int N) {
    int I = 0;
    // calculate I(i) for every 1<=i<=n and add it into I
    for (int i = 1; i <= N; i++)
            for (int j = 0; j < pow(2, N); j += pow(2, i))
                    for (int k = 0; k < pow(2, i - 1); k++)
                            if (f[k + j] != f[(k + j) + pow(2, i - 1)])
                                    I++;
    // return the value of Q_I(f)
    return (I + 0.0) / (N * pow(2, N - 1));
}
```

The outer loop iterates through every variable, and the inner loops iterate through the truth table associated to f_n (which is not explicitly constructed) in order to calculate the number of times the variable is important in determining the function's output. Since for every variable we have to iterate through the entire truth table (which contains 2^n entries), the cost of this algorithm is in $O(n \cdot 2^n)$. If we express this in the size of the truth table, the cost is $O(\log_2(N) \cdot N)$, where $N = 2^n$.

4.3. Equanimity based on the survival of subsets

In the last approach, we described the bias as the difference of importance between variables. Another reasonable consideration would be to define bias as the difference of importance between subsets of the input, but to measure this concept we need to state first what we mean by importance in subsets of the input. Note that the value of each variable could be 0 or 1, therefore subsets of the input are subsets of variables in any possible combination of positive and negative values.

Intuitively, a subset of the input is important if its value is reflected in the output of the Boolean function, i.e., its value survives the circuit. Let's formalize this idea, but first we need to fix some notation.

Notation 4.3.1. *Let $\vec{x} := (x_1, ..., x_n) \in \mathbb{Z}_2^n$ be a binary input, and $S \subset \{1, ..., n\}$ be a set of indices that represent a subset of variables, with $|S| = k$. Then, we denote $\vec{x}_S := (x_{i_1}, ..., x_{i_k})$, with $\{i_1, ..., i_k\} = S$.*

We are now in position to define the survival of subsets.

Definition 4.3.1 (Survival of subsets)**.** *Given a Boolean function $f : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$ and S be a subset of variables, with $|S| = k$. Then, we say $\vec{x}_S \in \mathbb{Z}_2^k$ survives if*

$$\exists \vec{z} \in \mathbb{Z}_2^{n-k} \ : g^{\vec{x}_S}(\vec{z}) = 1$$

where

$$g^{\vec{x}_S} : \mathbb{Z}_2^{n-k} \longrightarrow \mathbb{Z}_2$$
$$g^{\vec{x}_S}(\vec{x}_{\overline{S}}) = f(\vec{x})$$

Similarly to the first approach, it would be more informative to know the amount of survival a subset has.

Definition 4.3.2 (Degree of survival)**.** *Given a Boolean function $f : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$, the amount of times a subset of the input $\vec{x}_S$ survives is denoted by*

$$c_{\vec{x}_S} = |\{g^{\vec{x}_S}(\vec{z}) = 1 : z \in \mathbb{Z}_2^{n-k}\}|$$

This concept may seem a bit confusing, so let's clarify it with an example.

Example 4.3.1. *Let $f_4(x) = (\bar{x_1} \wedge x_2) \vee x_4$. Its associated truth table is*

x_1	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*
x_2	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*0*
x_3	*1*	*1*	*1*	*1*	*0*	*0*	*0*	*0*	*1*	*1*	*1*	*1*	*0*	*0*	*0*	*0*
x_4	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
$f_4(x)$	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*0*	*1*	*0*	*0*	*0*	*1*	*0*	*0*

Let's consider $S = \{1, 3\}$ and $\vec{x}_S = (0, 0)$, i.e., any input in which $x_1 = 0$ and $x_3 = 0$. This corresponds to the 6th, 8th, 14th and 16th columns of the truth table.

As we can observe, $g^{\vec{x}_S}((1, 1)) = f((0, 1, 0, 1)) = 1$ (6th column), $g^{\vec{x}_S}((0, 1)) = f((0, 0, 0, 1)) = 1$ (8th column), $g^{\vec{x}_S}((1, 0)) = f((0, 1, 0, 0)) = 1$ (14th column) and $g^{\vec{x}_S}((0, 0)) = f((0, 0, 0, 0)) = 0$ (16th column). Therefore, $c_{\vec{x}_S} = 3$.

An equanimous function should treat all subsets equally, therefore the degree of survival should be similar for every subset of the input. However, the amount of survival for a subset is dependent on the size of this subset. For this reason, it is desirable to seek equanimity in subsets of the same size. This led us to use variance to see if all degrees are the same for subsets of the same size.

Definition 4.3.3. *Let $f : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$ be a Boolean function. We define the average in degree of survival of a given size k as*

$$\mu_k = \frac{1}{\omega_k} \sum_{\substack{\vec{x}_S \in \mathbb{Z}_2^k \\ S \subset \{1,\dots,n\} \\ |S|=k}} c_{\vec{x}_S}$$

Then, the variance is defined as

$$\sigma_k = \frac{1}{\omega_k} \sum_{\substack{\vec{x}_S \in \mathbb{Z}_2^k \\ S \subset \{1,\dots,n\} \\ |S|=k}} (c_{\vec{x}_S} - \mu_k)^2$$

where ω_k is the number of subsets of size k of the input.

Note that the input contains negative and non negative variables, but in order to calculate the number of different subsets we need to take into account that a variable and its negation cannot be in the same subset. The following proposition gives a formula to calculate the value of ω_k.

Proposition 4.3.1. *Let $k \in \mathbb{N}$, then $\omega_k = 2^k \cdot \binom{n}{k}$.*

Proof: Note that ω_k can be expressed as the cardinal of the different binary strings for each subset of the input of size k, i.e.,

$$\omega_k := |\{\vec{x}_S : S \subset \{1, ..., n\}, |S| = k\}| = |\mathbb{Z}_2^k \times \{S \subset \{1, ..., n\}, |S| = k\}|$$

Since the cardinal of the Cartesian product is the product of the cardinals,

$$\omega_k := |\mathbb{Z}_2^k| \cdot |\{S \subset \{1, ..., n\}, |S| = k\}| = 2^k \cdot \binom{n}{k}$$

□

Since we are considering high values of equanimity should imply low bias, this approach should multiply by (-1) the sum of all σ_k, because high values of σ_k capture high bias. With all this, we can now formally define this approach of equanimity.

Definition 4.3.4 (Equanimity based on the survival of subsets)**.** *Given a Boolean function $f : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$, we define its equanimity as*

$$\mathcal{Q}_S(f) = -\sum_{k=1}^{n} \sigma_k$$

Note that high values of equanimity, i.e, values close to 0, imply f is unbiased.

4.3.1. Alternative to the definition

Let us take into account the following result regarding the range of ω_k.

Proposition 4.3.1.1. *Let $k \in \mathbb{N}$, then $0 \leqslant \omega_k \leqslant 2^{2(n-k-1)}$.*

Proof. Let $S \subset \{1, ..., n\}, |S| = k$, then $\forall\ \vec{x}_S \in \mathbb{Z}_2^{n-k}$, $0 \leqslant c_{\vec{x}_S} \leqslant |Z_2^{n-k}| = 2^{n-k}$. If we denote

$$m = 0 \qquad M = 2^{n-k}$$

By Popoviciu's inequality on variances [Po35],

$$\omega \leqslant \left(\frac{M-m}{2}\right)^2 = \left(\frac{2^{n-k}}{2}\right)^2 = 2^{2(n-k-1)}$$

□

We can see that if $k < k'$ then the value of equanimity depends more on σ_k than $\sigma_{k'}$. To solve this issue we could resize the range of values in every σ_k so that they are in the same scale of $[0, 1]$.

Once every σ_k is resized into values of $[0, 1]$, the sum of all these σ_k will be in $[0, n]$, where n is the size of the binary input. If we multiply this value by $\frac{1}{n}$ then the value will be in $[0, 1]$. Arguing in the same way so that high values of the metric lead to high values of equanimity, the last thing we have left to do is subtract 1 minus this. This will result in the following metric of equanimity.

Definition 4.3.5. *(Equanimity based on the survival of subsets normalized) Given a Boolean function $f : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$, we define its equanimity as*

$$\mathcal{Q}'_S(f) = 1 - \frac{1}{n} \sum_{k=1}^{n} \frac{\sigma_k}{2^{2(n-k-1)}} \in [0, 1]$$

where $\mathcal{Q}'_S(f) = 0$ means totally biased and $\mathcal{Q}_S(f) = 1$ means totally unbiased.

Although this is an alternative, it does not mean it is a better approach, we will compare all of them later.

4.3.2. Implementation for $\mathcal{Q}_S$

The principal part of the algorithm that receives f_n as an input and returns $Q_S(f_n)$ is the one that calculates $c_{\vec{x}_S}$. In order to do so, we create a data structure called *counters* which links each $\vec{x}_S$ with its associated $c_{\vec{x}_S}$. The algorithm will consist on iterating through the truth table of f_n and, if the output is 1, the counter value of every subset of that entry will be updated. The following function (in which x represents the entry of the truth table) is the one responsible for this:

```
using counters = unordered_map<vector<int>, int, hash_fn>;

void update_counters(vector<counters>& count, vector<int> & subset,
    vector<int>& x, int index) {
    // Update the counter of the corresponding subset
    count[subset.size()][subset]++;

    // Loop to create every subset of x
    for (int i = index; i < x.size(); i++) {
            // include x[i] in subset.
            subset.push_back(x[i]);

            // move onto the next element.
            update_counters(count, subset, x, i + 1);
```

```
                // exclude x[i] from subset and triggers
                // backtracking.
                subset.pop_back();
        }
}
```

As the reader may have noticed, the counters are organized by size to make it simpler to calculate each σ_k. In the worst case scenario ($f_n(x) = 1\ \forall x \in \mathbb{Z}_2^n$), the function update_counters is called 2^n times, once for each entry of the truth table. Since that function goes through every subset of a set of size n, the total cost of the algorithm is in $O(2^{2n})$ (note that calculating the average and the variance does not add additional cost). This time complexity is quadratic in the size of the truth table of f_n.

4.4. Limits of equanimity

In this chapter, we have built an intuition on why the circuit that computes an unbiased function must be big. This intuition will be confirmed in Chapter 6. However, although this is true most of the time, there exists a major counter example: the PARITY decision problem.

This function fits with the definition of unbiased because every bit in the binary input always has to be taken into consideration. This implies that the function does not bias any region of the input over another. However, as we mentioned in Chapter 3 PARITY is in $P_{/poly}$. This intuition should be captured in both approaches defined in the previous sections.

The following result proves that if we take the equanimity based on the importance of variables, PARITY is completely equanimous. This means that PARITY gives the highest degree of importance to every variable.

Theorem 4.4.1. $\forall n \in \mathbb{N},\ \mathcal{Q}_I(p_n) = 1$.

Proof. Let $n \in \mathbb{N}$, it is sufficient to show that for all variables i, $I(i) = 2^{n-1}$. In this case,

$$\mathcal{Q}_I = \frac{\sum_{i=0}^{n} I(i)}{n \cdot 2^{n-1}} = \frac{n \cdot 2^{n-1}}{n \cdot 2^{n-1}} = 1$$

For simplicity, we denote

$$\mathcal{Y} := \{(\vec{x}, \vec{y}) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n : (x_i, y_i) = (1,0) \wedge\ \forall j \neq i,\ x_j = y_j \wedge p_n(\vec{x}) \neq p_n(\vec{y})\}$$

Since $|\mathcal{Y}| = I(i)$, the statement is equivalent to say that the application

$$\begin{gathered}\Gamma : \mathbb{Z}_2^{n-1} \longrightarrow \mathcal{Y} \\ \vec{z} \mapsto (\vec{x}, \vec{y}) :\ (x_i, y_i) = (1,0) \wedge\ \forall j \neq i,\ x_j = y_j = z_j\end{gathered}$$

is bijective. Note that $\Gamma(\vec{z}) \in \mathcal{Y},\ \forall \vec{z} \in \mathbb{Z}_2^{n-1}$ due to the good properties of the PARITY function. Let's prove this. Let $\Gamma(\vec{z}) = (\vec{x}, \vec{y})$, then

$$p_n(\vec{x}) = x_i \oplus p_{n-1}(\vec{z}) = \neg p_{n-1}(\vec{z}) \neq p_{n-1}(\vec{z}) = y_i \oplus p_{n-1}(\vec{z}) = p_n(\vec{y})$$

The only thing left to prove is the bijectivity of gamma.

- Injective: Let $\vec{z_1}, \vec{z_2} \in \mathbb{Z}_2^{n-1}$ such that $(\vec{x_1}, \vec{y_1}) = \Gamma(\vec{z_1}) = \Gamma(\vec{z_2}) = (\vec{x_2}, \vec{y_2})$. Then,

  $$\vec{z_1} = \vec{x_1}_J = \vec{x_2}_J = \vec{z_1}$$

  where $\vec{x}_J = (x_1, ..., x_{i-1}, x_{i+1}, ..., x_n)$.

- Surjective: Let $(\vec{x}, \vec{y}) \in \mathcal{Y}$, if we take $\vec{z} = \vec{x}_J$, then by definition, $\Gamma(\vec{z}) = (\vec{x}, \vec{y})$.

- Well-defined: Let $\vec{z_1}, \vec{z_2} \in \mathbb{Z}_2^{n-1}$ such that $(\vec{x_1}, \vec{y_1}) = \Gamma(\vec{z_1}) \neq \Gamma(\vec{z_2}) = (\vec{x_2}, \vec{y_2})$, then

  $$\vec{x_1} \neq \vec{x_2} \implies \vec{z_1} = \vec{x_1}_J \neq \vec{x_2}_J = \vec{z_1}$$

By all this, $I(i) = |\mathcal{Y}| = |Z_2^{n-1}| = 2^{n-1}$. □

The other approach also takes into account this consideration. Every subset of the input is treated equally except the ones of maximum size. This is shown in the following result.

Theorem 4.4.2. $\forall n \in \mathbb{N},\ \mathcal{Q}_S(p_n) = -\frac{1}{4}$

Proof. It is sufficient to show that

$$\sigma_k = \begin{cases} 0, & 1 \leqslant k \leqslant n-1 \\ \frac{1}{4}, & k = n \end{cases}$$

since this implies

$$\mathcal{Q}_S(p_n) = -\sum_{k=1}^{n} \sigma_k = 0 - \sigma_n = -\frac{1}{4}$$

Therefore, the statement is divided into two parts:

- $\sigma_k = 0,\ \forall\ 1 \leqslant k \leqslant n-1$. Let $\vec{x_S} \in \mathbb{Z}_2^k$, with $S \in \{1, ..., n\}$, then

  $$\begin{aligned} c_{\vec{x_S}} &= |\{g^{\vec{x}_S}(\vec{z}) = 1 : z \in \mathbb{Z}_2^{n-k}\}| \\ &= |\{p_{\lfloor \frac{n}{2} \rfloor}(\vec{x}_S) \oplus p_{\lceil \frac{n}{2} \rceil}(\vec{z}) = 1 : z \in \mathbb{Z}_2^{n-k}\}| \\ &= |\{p_{\lceil \frac{n}{2} \rceil}(\vec{z}) = 1 : z \in \mathbb{Z}_2^{n-k}\}| \\ &= \frac{\omega_k}{2} \end{aligned}$$

  Since every subset of the input has the same degree of survival, then the variance is 0.

- $\sigma_n = \frac{1}{4}$. In this case, $S = \{1, ..., n\}$, so for all $\vec{x} \in \mathbb{Z}_2^n$,

$$c_{\vec{x}} = |\{p_n(\vec{x}) = 1\}| \leqslant 1$$

Therefore,

$$\mu_n = \frac{1}{\omega_n} \sum_{\substack{\vec{x}_S \in \mathbb{Z}_2^n \\ S=\{1,...,n\}}} c_{\vec{x}} = \frac{1}{\omega_n} \sum_{\vec{x} \in \mathbb{Z}_2^n} c_{\vec{x}} = \frac{1}{\omega_n} \sum_{\substack{\vec{x} \in \mathbb{Z}_2^n \\ p_n(\vec{x})=1}} 1 = \frac{1}{\omega_n} \cdot \frac{\omega_n}{2} = \frac{1}{2}$$

Then, the variance results in

$$\sigma_n = \frac{1}{\omega_n} \sum_{\vec{x} \in \mathbb{Z}_2^n} (c_{\vec{x}} - \mu_n)^2 = \frac{1}{\omega_n} \sum_{\vec{x} \in \mathbb{Z}_2^n} (\frac{1}{2})^2 = \frac{\omega_n \cdot \frac{1}{4}}{\omega_n} = \frac{1}{4}$$

Although equanimity is a good indicator to capture the reasons why circuits must be big, it has some troubles with functions that (despite being small) use a certain *trick* to be equanimous. For this reason, we need a metric that can ignore that trick and, in addition to equanimity, can provide us a circuit lower bound (ideally superpolynomial) for NP-complete problems. This new metric is what we will call entanglement.

□

Chapter 5

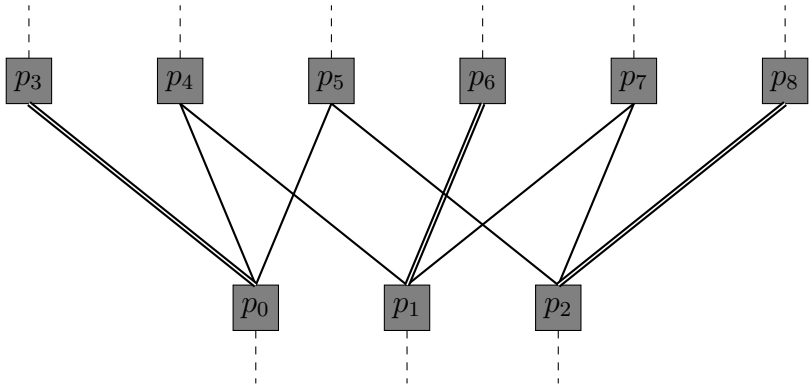
Entanglement

5.1. Introduction to the concept of entanglement

In physics, quantum entanglement refers to a fundamental phenomenon in quantum mechanics where two or more particles become correlated in such a way that their individual properties cannot be described independently [Ho09]. Similarly, in our case, it is intuitive to think that the messier a Boolean function is (i.e., if the effect of every input on the output of the function strongly depends on the other inputs) the bigger the circuit that computes it will be since the connection between variables will increase.

Note that the entanglement is a property of Boolean functions, not circuits that compute them. However, this metric should capture the behavior of all the circuits that compute a given Boolean function.

For instance, a function should be entangled if all levels of all the circuits that compute it look similar to the following structure.



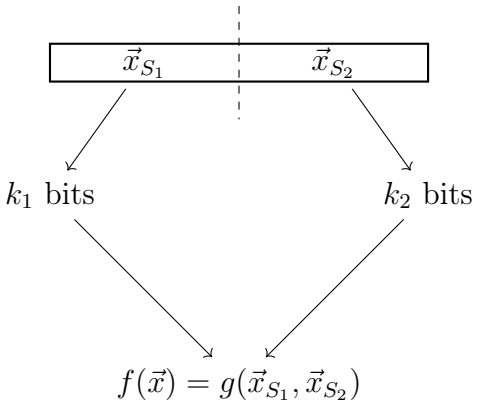
That is, in order to compute its function, the circuit needs to directly combine the outputs of the lower level in all possible combinations. Thus, the upper level cannot be split into

parts depending on (almost) disjoint subsets of outputs of the lower level. If this level cannot be untangled, the number of gates will increase inevitably in quadratic order to the number of outputs from the level below [1]. If this keeps happening for k levels, then the size of the circuits will be in the order of $O(n^{2^k})$.

Intuitively, the entanglement of the function will be higher if none of its circuits are capable of untangling their levels. If the number of levels needs to be high enough, then, as we have shown, the size of all the circuits that compute the function will be super-polynomial.

Our goal is to define a metric that is capable of capturing this behavior, but how can we quantify this phenomenon? An entangled function forces its variables to interact with each other, as it happens with the quantum entanglement we mentioned earlier. Therefore, it is easier to think that a function is entangled if each subset of variables must provide to its complementary a great amount of information in order to compute the function.

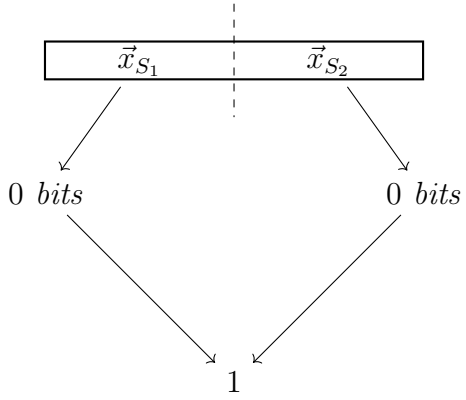
To understand the last idea, it is shown in the image below a representation of the amount of information partitions of the inputs are transmitting. For instance, the partition in the left, $\vec{x}_{S_1}$, needs to communicate k_1 bits of information and the partition in the right, $\vec{x}_{S_2}$, needs to send k_2 bits to compute the value of the function in $\vec{x}$, $f(\vec{x})$. A function is entangled if k_1 and k_2 are high. Note that the partition could be any subset of the set of variables, not only by dividing it in half as the diagram may suggest.



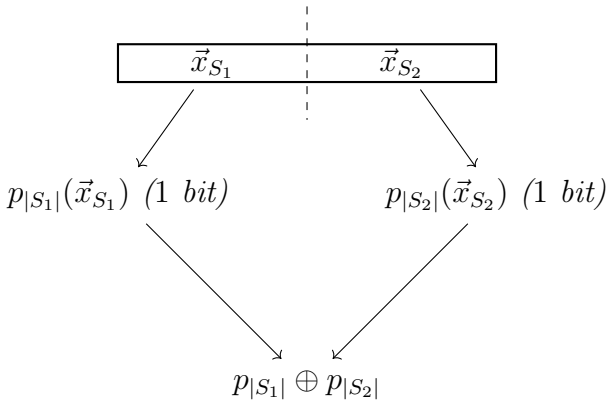
The concept is still pretty abstract and some examples can make it easier to comprehend it.

Example 5.1.1. *The least entangled functions are $f_n \equiv 1$ and $f_n \equiv 0$ because the partition does not need to communicate any kind of information. This is represented in the following figure.*

[1] For n inputs, the next level will need at least $\binom{n}{2}$ gates to connect the inputs with each other, and n more gates to connect the inputs with themselves. Note that $\binom{n}{2} + n \in O(n^2)$.



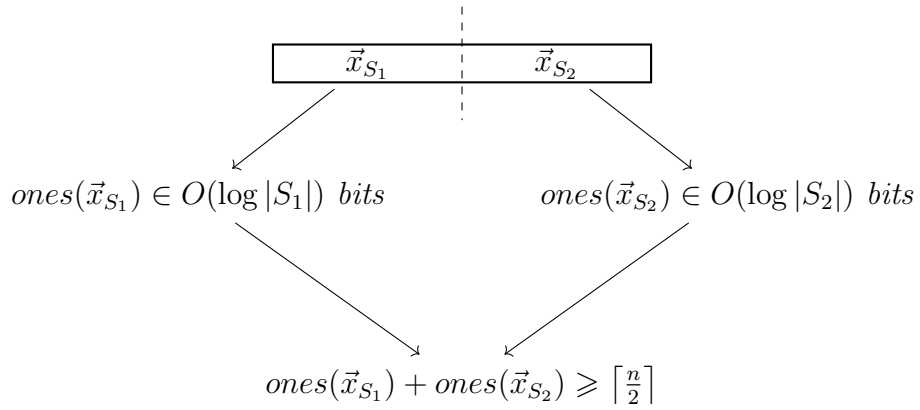
Example 5.1.2. *The PARITY function is one of the least entangled functions after the constant function, as the information needed to be shared between partitions is only one bit, which represents if the number of ones is either even or odd. In the figure,* p_n *denotes the PARITY function for inputs consisting of* n *bits.*



Example 5.1.3. *The MAJORITY function,* M_n*, has a similar behavior as PARITY, the amount of needed shared information is small.*

In this case, although every input is determinant for the output, the information they provide can be simplified. For instance, may $n = 6$ *and* $x = (1, 0, 1, 1, 1, 0)$*. We can divide* x *in half, obtaining* $x^1 = (1, 0, 1)$ *and* $x^2 = (1, 1, 0)$*. The information needed from each* x^i *is just the number of ones, so only* $\log m$ *bits are required, being* m *the size of* x^i*. If this simplification can be made, then the function will not be very entangled, since the dependence between variables is not that tight.*

As it is shown in the figure below, each partition needs to transmit the number of ones it has, which is in the worst case $\log |S|$*, with* S *being the partition of variables. When the total number of ones is gathered,* $ones(\vec{x})$*, the value of the function is* 1 *if and only if* $ones(\vec{x}) \geqslant \left\lceil \frac{n}{2} \right\rceil$*.*



In the examples above, the number of bits required to transmit the information is independent of the partition chosen. However, in same cases a function may seem entangled when observed through certain partitions and low-entangled when observed through others. Let's see an example of this.

Example 5.1.4. *Let's consider the language $\mathcal{L} = \{ww\} \subset \mathbb{Z}_2^*$, which can be expressed equivalently as $f : \mathbb{Z}_2^* \longrightarrow \mathbb{Z}_2$ such that $f(x) = 1 \iff x \in \mathcal{L}$.*

We will consider the functions of a certain size n, and more specifically, those in which n is even (note that if n is odd, $\mathcal{L} = \emptyset$, so f_n would be the constant 0 function).

If we consider the partition obtained by dividing the set of variables in half in the specific order in which they appear in the input, i.e., $S_1 = \{1, .., \frac{n}{2}\}$ and $S_2 = \{\frac{n}{2} + 1, ..., n\}$, then every bit is important and has to be transmitted. This happens because the first half of the input has to be exactly the same as the second one, which of course means every bit has to be the same, so no simplification can be made.

However, we can make the partition in a more clever way. Let us see our input $x \in Z_2^n$ as $x = w_1 w_2$, with $w_i \in Z_2^{\frac{n}{2}}$ and $w_i = w_i^1 w_i^2$, where $w_i^j \in \mathbb{Z}_2^{\frac{n}{4}}$. Let's take the subset S_1 such that w_1^1 and w_2^1 belong to it, and S_2 its complementary. Since $w_1 = w_2 \iff w_1^i = w_2^i$ for $i \in \{1, 2\}$, S_1 only needs to transmit one bit to S_2 which indicates whether $w_1^1 = w_2^1$ or not (and the same happens from S_2 to S_1).

Following these intuitions, we can understand the entanglement of a Boolean circuit as its internal ability to simplify information in order to obtain the desired output. In our experiments we will show how this data simplification is directly related to a reduction in the number of gates needed to compute the function with a circuit.

5.2. Entanglement formal definition

From here on, this abstract concept will be formalized in order to create a metric. For this purpose, we will once again use Notation 4.3.1. We can now define what is a partition of input variables.

Definition 5.2.1 (Partition of variables)**.** *Let $\mathcal{V} := \{1, ..., n\}$ be the set of variables and $S \subset \mathcal{V}$, then $\{S, V \setminus S\}$ represents a partition of input variables. For simplicity of notation, we refer to $\mathcal{V} \setminus S$ as $\overline{S}$ unless otherwise mentioned.*

We need to measure the amount of information needed to be transmitted in the partition before defining the metric. A simple manner to do so is counting the number of different functions obtained by fixing the values of one of the subsets in the partition. This can be formally defined as follows.

Definition 5.2.2 (Information shared by a subset)**.** *Let $f : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$ be a Boolean function and $\{S, \overline{S}\}$ be a partition of variables, where $|S| = k$. The information that the partition S gives to its complementary is*

$$i(S) = |\{g^{\vec{x}_S} : \vec{x}_S \in \mathbb{Z}_2^k\}|$$

where

$$g^{\vec{x}_S} : \mathbb{Z}_2^{n-k} \longrightarrow \mathbb{Z}_2$$
$$g^{\vec{x}_S}(\vec{x}_{\overline{S}}) = f(\vec{x})$$

Then, an early definition of entanglement could be the following.

Definition 5.2.3 (First definition of entanglement)**.** *Given a Boolean function $f : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$, we define the entanglement of f as the minimum of the information shared in the different partitions, i.e.,*

$$\mathcal{T}(f) = \min\{i(S) + i(\overline{S}) : S \subset \{1, ..., n\}\}$$

However, we need to restrict the size of the partition S since otherwise we would get a short range of entanglement values, as it can be seen in the following result.

Lemma 5.2.1. *Given $f : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$, for $S \subset \{1, ..., n\}$ of any size, $i(S) \leqslant min(2^{|S|}, 2^{2^{|\overline{S}|}})$.*

Proof: For each possible assignment of values to all variables in S, at most one additional g function is included in the count, so we can obtain no more than $2^{|S|}$ different g functions. Also, since g functions depend on $|\overline{S}|$ bits, there can be no more than $2^{2^{|\overline{S}|}}$ of them. □

A problem arises when S is too small (or too big), because the maximum value of $i(S) + i(\overline{S})$ would be small compared to other partitions. Since the entanglement is calculated as the minimum of the information shared, these partitions would lower its value just because of their size, making this metric less useful.

We are finally in the position to give a proper definition of entanglement.

Definition 5.2.4 (Entanglement)**.** *Given a Boolean function $f : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$, we define the entanglement of f as the minimum of the information shared in the different half-sized partitions, i.e.,*

$$\mathcal{T}(f) = \min\{i(S) + i(\overline{S}) : S \subset \{1, ..., n\}, |S| = \lfloor \tfrac{n}{2} \rfloor\}$$

With this formal definition we are now in the position to show that PARITY has a low entanglement value.

Theorem 5.2.1. $\forall n \in \mathbb{N}, \mathcal{T}(p_n) = 4$.

Proof: Let's show $i(S) = 2$ for any S we choose. In order to do that, we need to prove the following statements first:

I $\exists z_0 \in \mathbb{Z}_2^{\lceil \frac{n}{2} \rceil} : g^{\vec{x}_S}(z_0) = g^{\vec{y}_S}(z_0) \Rightarrow \forall z \in \mathbb{Z}_2^{\lceil \frac{n}{2} \rceil}, g^{\vec{x}_S}(z) = g^{\vec{y}_S}(z)$

II $\exists z_0 \in \mathbb{Z}_2^{\lceil \frac{n}{2} \rceil} : g^{\vec{x}_S}(z_0) = \neg g^{\vec{y}_S}(z_0) \Rightarrow \forall z \in \mathbb{Z}_2^{\lceil \frac{n}{2} \rceil}, g^{\vec{x}_S}(z) = \neg g^{\vec{y}}_S(z)$

We will prove the first statement (the second one is analogous). We can express $g^{\vec{x}_S}$ as

$$g^{\vec{x}_S}(z) = p_n(\vec{x}) = x_1 \oplus x_2 \oplus ... \oplus x_n = p_{\lfloor \frac{n}{2} \rfloor}(\vec{x}_S) \oplus p_{\lceil \frac{n}{2} \rceil}(z)$$

and since $g^{\vec{x}_S}(z_0) = g^{\vec{y}_S}(z_0)$, we have

$$p_{\lfloor \frac{n}{2} \rfloor}(\vec{x}_S) \oplus p_{\lceil \frac{n}{2} \rceil}(z_0) = p_{\lfloor \frac{n}{2} \rfloor}(\vec{y}_S) \oplus p_{\lceil \frac{n}{2} \rceil}(z_0)$$

so [2]

$$p_{\lfloor \frac{n}{2} \rfloor}(\vec{x}_S) = p_{\lfloor \frac{n}{2} \rfloor}(\vec{y}_S)$$

We can conclude, $\forall z \in \mathbb{Z}_2^{\lceil \frac{n}{2} \rceil}$,

$$g^{\vec{x}_S}(z) = p_{\lfloor \frac{n}{2} \rfloor}(\vec{x}_S) \oplus p_{\lceil \frac{n}{2} \rceil}(z) = p_{\lfloor \frac{n}{2} \rfloor}(\vec{y}_S) \oplus p_{\lceil \frac{n}{2} \rceil}(z) = g^{\vec{y}_S}(z)$$

Therefore, for each partition there are only two different functions, i.e., $\forall S, i(S) = 2$. □

Similarly, it can be shown that the MAJORITY function has low entanglement values.

Theorem 5.2.2. $\forall n \in \mathbb{N}, \mathcal{T}(M_n) = \lceil \frac{n}{2} \rceil + 1 + \lfloor \frac{n}{2} \rfloor + 1$

Proof: Let $S \subset \{1, ..., n\}$. By the definition of the majority problem, we know

$$g^{\vec{x}_S}(z) = M_n(\vec{x}) = 1 \Leftrightarrow ones(\vec{x}) \geqslant \left\lceil \frac{n}{2} \right\rceil \Leftrightarrow ones(\vec{x}_S) + ones(z) \geqslant \left\lceil \frac{n}{2} \right\rceil$$

In order to obtain the value of the entanglement, we will prove first the following statements:

[2] Note that the XOR function satisfies $a \oplus c = b \oplus c \iff a = b$, $\forall a, b, c \in \mathbb{Z}_2$.

- $ones(\vec{x}_S) = ones(\vec{y}_S) \Rightarrow \forall z \in \mathbb{Z}_2^{\lceil \frac{n}{2} \rceil}, g^{\vec{x}_S}(z) = g^{\vec{y}_S}(z)$
- $ones(\vec{x}_S) \neq ones(\vec{y}_S) \Rightarrow \exists z_0 \in \mathbb{Z}_2^{\lceil \frac{n}{2} \rceil}, g^{\vec{x}_S}(z_0) \neq g^{\vec{y}_S}(z_0)$

The first one is immediate. Let $\vec{x}_S, \vec{y}_S \in Z_2^{\lfloor \frac{n}{2} \rfloor}$ such that $ones(\vec{x}_S) = ones(\vec{y}_S)$. Then, for $z \in \mathbb{Z}_2^{\lceil \frac{n}{2} \rceil}$,

$$g^{\vec{x}_S}(z) = 1 \Leftrightarrow ones(\vec{x}_S) + ones(z) \geqslant \left\lceil \frac{n}{2} \right\rceil \Leftrightarrow ones(\vec{y}_S) + ones(z) \geqslant \left\lceil \frac{n}{2} \right\rceil \Leftrightarrow g^{\vec{y}_S}(z) = 1$$

Let's prove the second one.

We can take, w.l.o.g., $\vec{x}_S, \vec{y}_S \in Z_2^{\lfloor \frac{n}{2} \rfloor}$ such that $ones(\vec{x}_S) > ones(\vec{y}_S)$. Let $z_0 \in \mathbb{Z}_2^{\lceil \frac{n}{2} \rceil}$ such that $ones(\vec{x}_S) + ones(z_0) = \lceil \frac{n}{2} \rceil$. Then,

$$ones(\vec{y}_S) + ones(z_0) < ones(\vec{x}_S) + ones(z_0) = \left\lceil \frac{n}{2} \right\rceil$$

so

$$g^{\vec{x}_S}(z_0) = 1 \neq 0 = g^{\vec{y}_S}(z_0)$$

We have proven the only cases where the $g^{\vec{x}_S}$ functions are different happen when $ones(\vec{x}_S)$ varies. Since $|S| = \lfloor \frac{n}{2} \rfloor$, $ones(\vec{x}_S) \in \{0, 1, ..., \lfloor \frac{n}{2} \rfloor\}$, so $i(S) = \lfloor \frac{n}{2} \rfloor + 1$. By the same argument, $i(\bar{S}) = \lceil \frac{n}{2} \rceil + 1$.

□

5.3. Properties of entanglement metric

In this section, we will discuss two properties of entanglement that are interesting to consider when working with this metric.

As a consequence of Lemma 5.2.1, we can obtain a range of values for the entanglement metric of any arbitrary function $f : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$. To do this, we need to prove the following result.

Lemma 5.3.1. *Given $f : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$, for $S \subset \{1, ..., n\}$ such that $|S| = \lfloor \frac{n}{2} \rfloor$, $i(S) \leqslant 2^{\lfloor \frac{n}{2} \rfloor}$ and $i(\overline{S}) \leqslant 2^{\lceil \frac{n}{2} \rceil}$.*

Proof: From Lemma 5.2.1, $i(S) \leqslant min(2^{|S|}, 2^{2^{|\overline{S}|}})$, so we have to show $2^{\lfloor \frac{n}{2} \rfloor} \leqslant 2^{2^{\lceil \frac{n}{2} \rceil}}$ for any $n \geqslant 1$. This is trivially true since $\forall n \geqslant 1$, $\lfloor \frac{n}{2} \rfloor \leqslant \lceil \frac{n}{2} \rceil < 2^{\lceil \frac{n}{2} \rceil}$. Equivalently, for $\overline{S}$, it is sufficient to prove $2^{\lceil \frac{n}{2} \rceil} \leqslant 2^{2^{\lfloor \frac{n}{2} \rfloor}}$ for any $n \geqslant 1$, which is true since $\lceil \frac{n}{2} \rceil \leqslant \lfloor \frac{n}{2} \rfloor + 1 \leqslant 2^{\lfloor \frac{n}{2} \rfloor}$.

□

This brings us to the following property of entanglement.

Property 5.3.1. *Let $f : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$ be a Boolean function, then $\mathcal{T}(f) \leqslant 2^{\lfloor \frac{n}{2} \rfloor} + 2^{\lceil \frac{n}{2} \rceil}$.*

Observation 5.3.1. *With this range of values, it can be observed that the entanglement values in PARITY and MAJORITY are quite far from the upper limit, suggesting that they have relatively small values.*

Another interesting property is that the entanglement of a given function will not increase if we add variables that do not provide information. We use the definition of important variable that we established in Definition 4.2.1 to refer what we mean by variable that provides information. Since we are going to use the set of important variables several times, we fix the following notation.

Notation 5.3.1. *The subset of variables that are important is denoted by $I \subset \{1, ..., n\}$.*

In this way, using the restriction notation that we had established in the previous section, we can define the restriction of a Boolean function, f, to I as the important function of f. Formally,

Definition 5.3.1 (Important function)**.** *Let $f : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$ be a Boolean function. The function*

$$f_I : \mathbb{Z}_2^m \longrightarrow \mathbb{Z}_2, \text{ with } m = |I|,$$

is the important function of f if $\forall \vec{x} \in \mathbb{Z}_2^n$, $f(\vec{x}) = f_I(\vec{x}_I)$.

We need to see if this definition is consistent, which is shown in the following result.

Proposition 5.3.1. *Let $X := \{f : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2\}$ and $Y := \{f : \mathbb{Z}_2^m \longrightarrow \mathbb{Z}_2\}$. The application*

$$\begin{gathered} \gamma : X \longrightarrow Y \\ f \mapsto f_I \end{gathered}$$

is bijective. Therefore, every restriction of γ is also bijective to its image, i.e., if $X' \subset X$ then,

$$\gamma|_{X'} : X' \longrightarrow \gamma(X') \text{ where } \gamma(X') = \{\gamma(f) : f \in X'\}$$

is also bijective.

Proof:

- Well-defined. If $\gamma(f)$ for a given $f \in X$ has two different images, h^1, h^2, then there is a $z_0 \in \mathbb{Z}_2^m$ with $h^1(z_0) \neq h^2(z_0)$. Since z_0 contains the important part, it can be extended to an arbitrary vector $\vec{x} \in \mathbb{Z}_2^n$ whose restriction to I is z_0. Note that for every extension, we obtain the same value for f by definition of I. Therefore,

$$\begin{gathered} f(\vec{x}) = f_I(\vec{x}_I) = f_I(z_0) = \gamma(f)(z_0) = h^1(z_0) \neq \\ \neq h^2(z_0) = \gamma(f)(z_0) = f_I(z_0) = f_I(\vec{x}_I) = f(\vec{x}) \end{gathered}$$

which is a contradiction.

- Injective. Let $f^1, f^2 \in X$ such that $f^1 \neq f^2$, i.e, $\exists \vec{x}_0 \in \mathbb{Z}_2^n : f^1(\vec{x}_0) \neq f^2(\vec{x}_0)$. Then, $\overline{\gamma(f^1)(\vec{x}_{0_I})} = f_I^1(\vec{x}_{0_I}) = f^1(\vec{x}_0) \neq f^2(\vec{x}_0) = f_I^2(\vec{x}_{0_I}) = \gamma(f^2)(\vec{x}_{0_I})$.
- Surjective. Let $h \in Y$. Let's take $f : \mathbb{Z}_2^n \longrightarrow Z_2$ such that $f(\vec{x}) = h(\vec{x}_I)\ \forall \vec{x} \in Z_2^n$. By definition, $f_I \equiv h$, so $\gamma(f) \equiv f_I \equiv h$.

The second statement is immediate. □

Thanks to the bijection of γ in the previous proposition we can now prove the following property of entanglement.

Property 5.3.2. *According to the definition of entanglement, let $\mathcal{V} := \{1, ..., n\}$ be the set of variables of a Boolean function $f : \mathbb{Z}_n^2 \longrightarrow \mathbb{Z}_2$, and I the set of important variables of $\mathcal{V}$. Then,*

- $\mathcal{T}(f) = \min\{i(S) + i(\mathcal{V} \setminus S) : S \subset \mathcal{V},\ |S| = \lfloor \frac{n}{2} \rfloor\}$.
- $\mathcal{T}(f_I) = \min\{i(S_I) + i(I \setminus S_I) : S_I \subset I,\ |S_I| = \lfloor \frac{m}{2} \rfloor\}$, *where* $m = |I|$.

It is true that $\mathcal{T}(f) \leqslant \mathcal{T}(f_I)$.

Proof. To prove this statement, it is enough to show that

$$\forall S_I,\ \exists S : i(S) = i(S_I) \wedge i(\mathcal{V} \setminus S) = i(I \setminus S_I),$$

since this would imply

$$\forall S_I,\ \exists S : i(S) + i(\mathcal{V} \setminus S) = i(S_I) + i(I \setminus S_I),$$

i.e.,

$$\{i(S_I) + i(I \setminus S_I) : S_I \subset I, |S_I| = \lfloor \tfrac{m}{2} \rfloor\} \subset \{i(S) + i(\mathcal{V} \setminus S) : S \subset \{1, ..., n\}, |S| = \lfloor \tfrac{n}{2} \rfloor\}$$

showing that

$$\mathcal{T}(f_I) = \min\{i(S_I) + i(I \setminus S_I) : S_I \subset I, |S_I| = \lfloor \frac{m}{2} \rfloor\} \geqslant$$
$$\min\{i(S) + i(\mathcal{V} \setminus S) : S \subset \{1, ..., n\}, |S| = \left\lfloor \frac{n}{2} \right\rfloor\} = \mathcal{T}(f)$$

Let $S_I \subset I$, where $|S_I| = \lfloor \frac{m}{2} \rfloor$. We can find a subset $S \subset \{1, ..., n\}$ with the same information by adding non-important variables to S_I until $|S| = \lfloor \frac{n}{2} \rfloor$. This is, $S = S_I \sqcup U$,[3] where $U \subset \overline{I}$. Note that $S_I \subset S \wedge I \setminus S_I \subset \mathcal{V} \setminus S$, thus showing $i(S) = i(S_I)$ is the same as proving $i(\mathcal{V} \setminus S) = i(I \setminus S_I)$. Let's show the first statement.

[3]The symbol $\sqcup$ is used to represent disjoint union between sets.

It is true that for arbitrary $\vec{x} \in \mathbb{Z}_2^n$,

$$\begin{aligned}
g^{\vec{x}_{S_I}}(\vec{x}_{I\setminus S_I}) &= f_I(\vec{x}_I) && \text{(By definition of } g^{\vec{x}_{S_I}}\text{)}\\
&= f(\vec{x}) && \text{(By definition of } f_I\text{)}\\
&= g^{\vec{x}_S}(\vec{x}_{\mathcal{V}\setminus S}) && \text{(By definition of } g^{\vec{x}_S}\text{)}\\
&= g_I^{\vec{x}_S}((\vec{x}_{\mathcal{V}\setminus S})_I) && \text{(By definition of } g_I^{\vec{x}_S}\text{)}\\
&= g_I^{\vec{x}_S}(\vec{x}_{I\setminus S_I}) && (I\setminus S_I \text{ contains all the important variables of } \mathcal{V}\setminus S)
\end{aligned}$$

Therefore, $g^{\vec{x}_{S_I}} \equiv g_I^{\vec{x}_S}$. We have proved that,

$$\gamma(\{g^{\vec{x}_S} : \vec{x}_S \in \mathbb{Z}_2^{\lfloor \frac{n}{2} \rfloor}\}) = \{g_I^{\vec{x}_S} : \vec{x}_{S_I} \in \mathbb{Z}_2^{\lfloor \frac{m}{2} \rfloor}\} = \{g^{\vec{x}_{S_I}} : \vec{x}_{S_I} \in \mathbb{Z}_2^{\lfloor \frac{m}{2} \rfloor}\}$$

By Proposition 5.3.1,

$$i(S) = |\{g^{\vec{x}_S} : \vec{x}_S \in \mathbb{Z}_2^{\lfloor \frac{n}{2} \rfloor}\}| = |\{g^{\vec{x}_{S_I}} : \vec{x}_{S_I} \in \mathbb{Z}_2^{\lfloor \frac{m}{2} \rfloor}\}| = i(S_I),$$

as we wanted to prove. □

Example 5.3.1. *Let's show a case where $\mathcal{T}(f) < \mathcal{T}(f_I)$ and a practical method to calculate the entanglement of a function using its truth table.*

Let $f_2(x) = x_1 \wedge x_2$. Its truth table is

x_1	*1*	*0*	*1*	*0*
x_2	*1*	*1*	*0*	*0*
$x_1 \wedge x_2$	*1*	*0*	*0*	*0*

In order to calculate the entanglement, we denote $S = \{1\}$, so $\overline{S} = \{2\}$. By fixing $x_1 = 1$, we obtain

x_1	*1*	*1*
x_2	*1*	*0*
$g^1(x_2)$	*1*	*0*

and by fixing $x_1 = 0$, we obtain

x_1	*0*	*0*
x_2	*1*	*0*
$g^0(x_2)$	*0*	*0*

Therefore, since $g^1(x_2) \neq g^0(x_2)$, we have $i(S) = 2$. By applying the same method, we obtain $i(\overline{S}) = 2$, hence $\mathcal{T}(f_2) = 4$ (note that switching the roles of S and $\overline{S}$ does not alter the addition).

Let's now define $f_4(x) = x_1 \wedge x_2$. Its truth table is

x_1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
x_2	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
x_3	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
x_4	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
$x_1 \wedge x_2$	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0

In this case, S could be $S_1 = \{1,2\}, S_2 = \{1,3\}$, or $S_3 = \{1,4\}$. Note that, since variables x_3 and x_4 don't provide information, if $S = S_2$ or $S = S_3$ then the number of different g functions obtained will be the same as with $f = f_2$ and $S = \{1\}$, so $i(S_2) + i(\overline{S}_2) = i(S_3) + i(\overline{S}_3) = 4$. Let's calculate it when $S = S_1$.

Since we are fixing x_1 and x_2, which are the only variables that matter, $i(S) =^4 |\{(x_1 \wedge x_2)^4 : x_1, x_2 \in \mathbb{Z}_2\}| = 2$. For $\overline{S}$ it happens exactly the opposite, so no matter how we fix x_3 and x_4, we always obtain $g^{(x_3,x_4)}(x_1,x_2) = f_2(x_1,x_2)$, so $i(\overline{S}) = 1$. In short, $i(S) + i(\overline{(S)}) = 3$, so $\mathcal{T}(f_4) = 3$.

5.4. Implementation for $\mathcal{T}$

In order to obtain the time complexity required to calculate the entanglement value of a function of n input bits, we will present a C++ code of the algorithm and analyze its cost.

The main component of calculating the entanglement is determining $i(S)$ for every S, which involves computing the g-functions. The following function, given a subset $S \subset \{1,...,n\}$ such that $|S| = \lfloor \frac{n}{2} \rfloor$ (or $|S| = \lceil \frac{n}{2} \rceil$) and the truth table of a Boolean function, returns $\{g^{\vec{x}_S} : \vec{x}_S \in \mathbb{Z}_2^k\}$.

```
void calculate_g_functions(vector<int>& x, const vector<int>& S,
        const int pos_s, const bool B, vector<int>& g,
        unordered_set<vector<int>, hash_fn>& g_set, const int N,
        const truth_table & TT) {
        // Base condition
        if (pos_s < 0) {
                // S
                if (B) {
                        vector<int> S_c;
                        complementary_subset(S, S_c, N);
                        vector<int> g;
                        calculate_g_functions(x, S_c, S_c.size() - 1, false, g,
                            g_set, N, TT);
                        g_set.insert(g);
                }
```

[4] For $n \in \mathbb{N}$ and $k \in \mathbb{Z}_2$, we denote by $(k)^n \in \mathbb{Z}_2^n$ the element k repeated n times

```
                // S_c
                else {
                        g.push_back(TT.at(x));
                }
        }
        // Recursive conditions
        else {
                x[S[pos_s]] = 1;
                calculate_g_functions(x, S, pos_s - 1, B, g, g_set, N, TT);
                x[S[pos_s]] = 0;
                calculate_g_functions(x, S, pos_s - 1, B, g, g_set, N, TT);
        }
}
```

Since $|\{\vec{x}_S\}| = |\mathbb{Z}_2^{\lfloor \frac{n}{2} \rfloor}| = 2^{\lfloor \frac{n}{2} \rfloor}$ and for each of them we have to obtain every $\vec{x}_{\bar{S}}$ (there are $2^{\lceil \frac{n}{2} \rceil}$ of these) the total cost of obtaining the g-functions belongs to $O(2^{\lfloor \frac{n}{2} \rfloor} \cdot 2^{\lceil \frac{n}{2} \rceil}) = O(2^n)$. Given that we have to calculate the g-functions for every S (in order to obtain $i(S)$), and there are $\binom{n}{\lfloor \frac{n}{2} \rfloor}$ different subsets of size $\lfloor \frac{n}{2} \rfloor$, the total cost to obtain the entanglement of a Boolean function of n input bits is in $O(\binom{n}{\lfloor \frac{n}{2} \rfloor} \cdot 2^n) \equiv O(2^{2n})$, i.e., quadratic on the size of the truth table.

5.5. Do we still need equanimity?

We could think that entanglement works well on its own and that equanimity would not be necessary to determine when a function needs to be computed by a superpolynomial circuit. In this section, we are not going to show a counter-example as we did in Section 4.4 for equanimity, but a problem which is in $P_{/poly}$ that apparently has high values of entanglement and low values of equanimity, namely the PRIMALITY decision problem.

The problem that PRIMALITY has is that the "what" matters, causing each subset of variables to contribute a great amount of information to its complementary. This would result in a high degree of entanglement.

However, if we take into account the prime number theorem (PNT) proved independently by Jacques Hadamard [Had96] and Charles Jean de la Vallée Poussin [VP96] regarding the density of prime numbers, equanimity will decrease as n rises.

The prime number theorem describes the asymptotic distribution of the prime numbers among the positive integers, formalizing the intuitive idea that primes become less common as they become larger by precisely quantifying the rate at which this occurs. Because of this behavior of prime numbers, intuitively, the variables with high indices will not be

considered equally important to variables with low indices. This is reflected empirically in Chapter 6.

Another important consideration to take into account is that entanglement, as we saw in Section ??, is a metric that can be computed in polynomial time with respect to the truth table, thus satisfying the first condition of the natural proof. Therefore, it will be necessary to combine this metric with equanimity to narrow down the set of functions and avoid the second condition of natural proof.

Chapter 6

Experimental results

6.1. Performance of metrics in the dataset

If we recall our standpoint, we aim to observe that a highly equanimous and entangled function must be computed only by super polynomial Boolean circuits. Therefore, experimentally, it should happen that for functions computed by small circuits, low values are obtained for at least one of the metrics.

José Ramón Cobián, in his Mathematics bachelor's thesis [CF23], has programmed a circuit generator that solely employs NAND gates, which we will employ as a reference for functions computed by small circuits. The dataset displays *all* Boolean functions of 5 bits in one whose minimal circuits of this type have up to 10 gates.[1] We will compare these functions with others from a set of equal size[2] that are selected in a random manner, making sure they do not belong to the dataset.

Firstly, we will present the results of the metrics separately, and later we will examine if the metrics are capable of distinguishing the randomly generated functions from those in the dataset.

6.1.1. Testing the metrics individually

The graphs that we will present all share the same structure:

- Two histograms, one for the functions within the dataset and another for those outside the dataset, are displayed on the same figure corresponding to the respective metric.

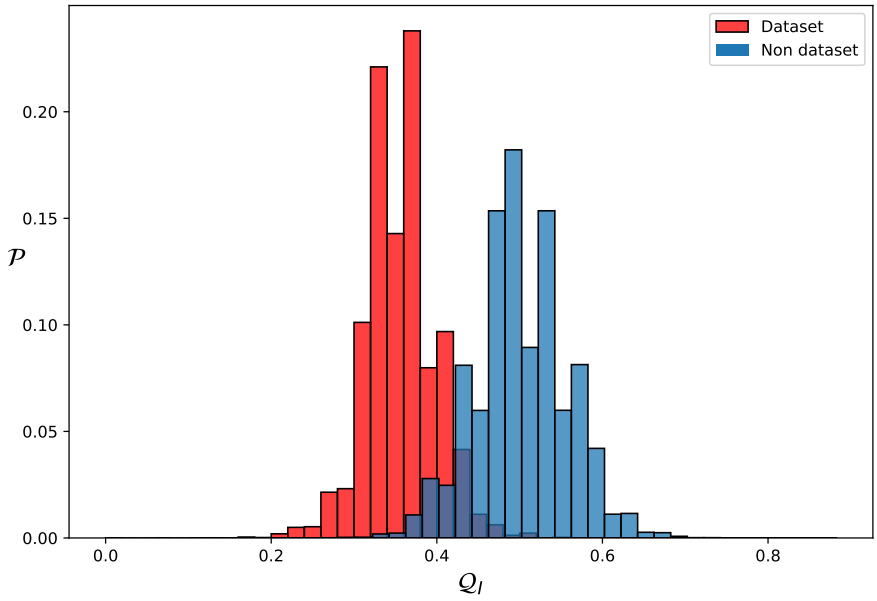
[1] In his unfinished bachelor's thesis, which he has yet to submit, he demonstrates that all circuits can be transformed into an equivalent one using only NAND gates with a polynomial difference in the number of gates. Therefore, studying this type of circuits does not entail a loss of generality.

[2] The dataset contains 9,268,995 functions, which is approximately 0.2158% of the total number of different Boolean functions with 5 input bits.

- The x-axis is discretized into intervals representing the metrics values.
- The y-axis represents the proportion of functions that fall within the corresponding interval, which we will represent with $\mathcal{P}$.

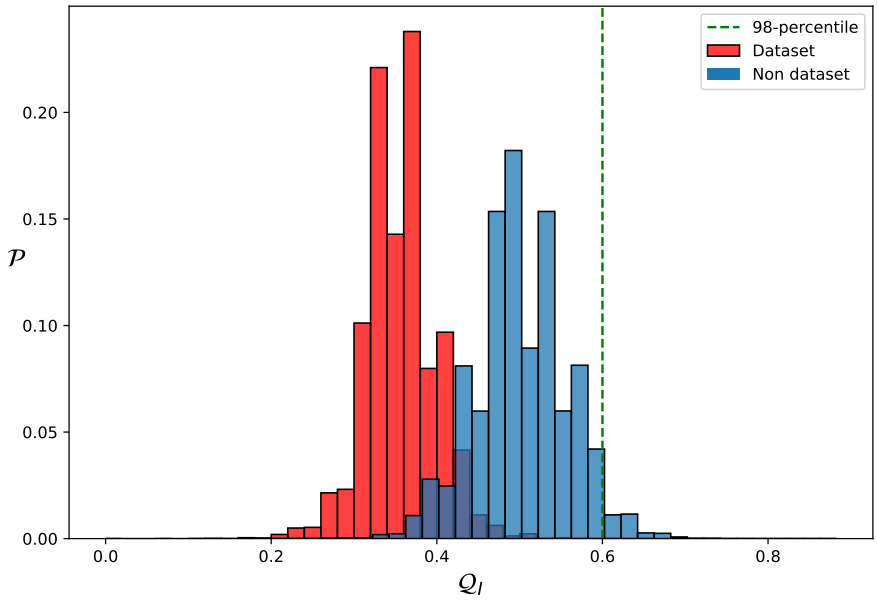
We initiate the analysis with equanimity. As discussed in Chapter 4, this metric can potentially indicate a larger circuit size. However, there are certain exceptional cases among smaller circuits that leverage unknown techniques to achieve a balanced distribution of their variables, as exemplified by the PARITY function.

In Chapter 4, we had defined several approaches to equanimity. Let us first examine the one **based on the importance of variables**.



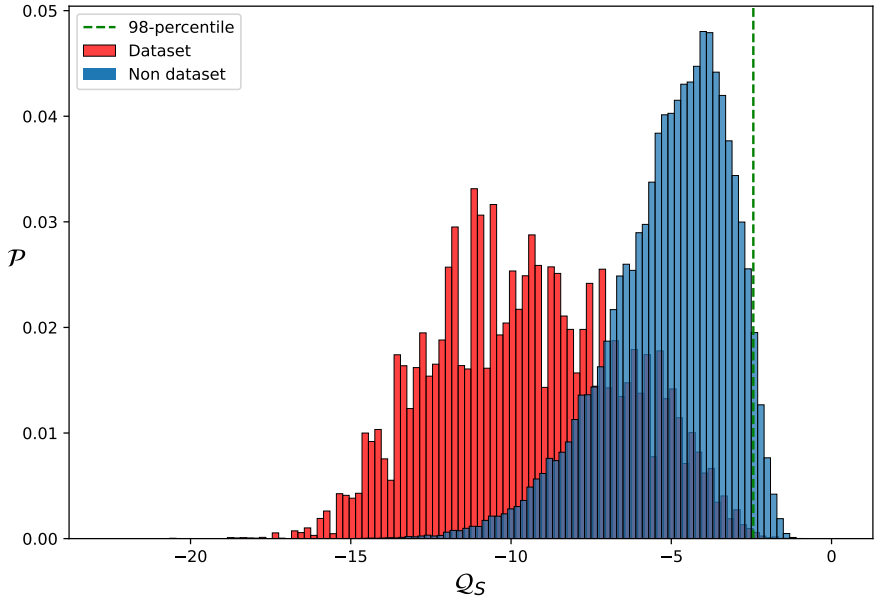
From this image, we can observe a quasi-separation between the set of functions within the dataset and those outside of it. Indeed, the intuitions we raised at the beginning of the study are confirmed with this result, as functions outside the dataset exhibit higher values of equanimity than those within.

The cases that truly interest us are the extremes, that is, those that are highly equanimous and highly entangled. For the equanimity case, we do not know the exact value that would determine whether a function is equanimous or not. To address this, we consider the 98th percentile, which corresponds to the equanimity value that only 2% of the analyzed functions surpass. We can see below in a green dashed line the value for this threshold.

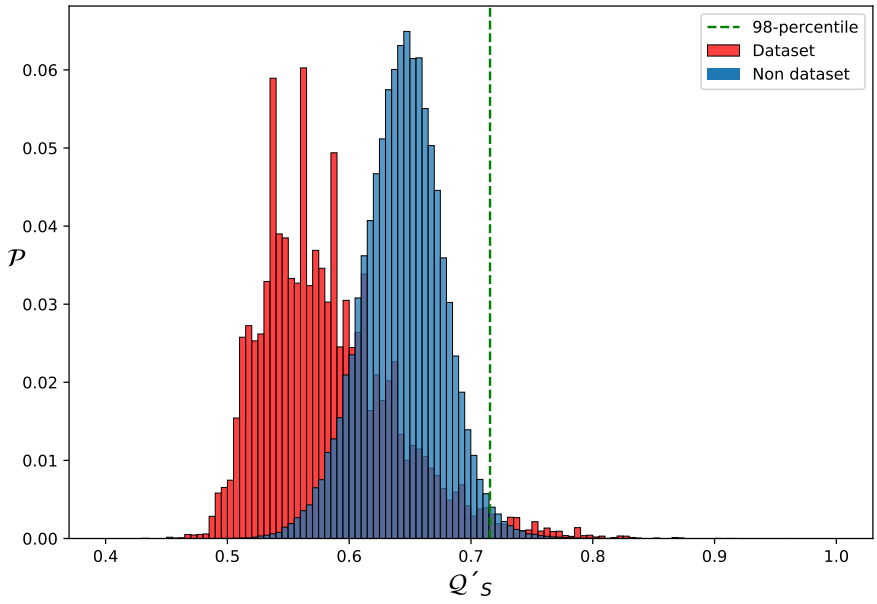


It can be observed that there is no function in the dataset that surpasses this threshold, which could serve as a bound to determine when a 5-bit function is equanimous. To refine the value of this bound further, one should experiment with n-bit functions and observe their behavior as n rises. That is precisely what we will do in the experiment of the next section.

Let's analyze the results we obtain when examining the other approach to equanimity, considering both its primary version and its normalized version. Our objective is to examine the behavior of **subset-based equanimity**, its ability to accurately separate datasets, and compare these conclusions with previously discussed approach to equanimity. First, let's examine the **non-normalized version**.



Despite the fact that the highest values of equanimity are mostly achieved by functions outside the dataset, there are indeed functions within the dataset that exhibit high levels of equanimity. This was already expected, as constant functions are highly equanimous according to this approach. Now let's analyze the **normalized one**.



Here, it is impossible to determine a clear separation between the dataset and outside the dataset. Perhaps it is more interesting to highlight that the least equanimous functions

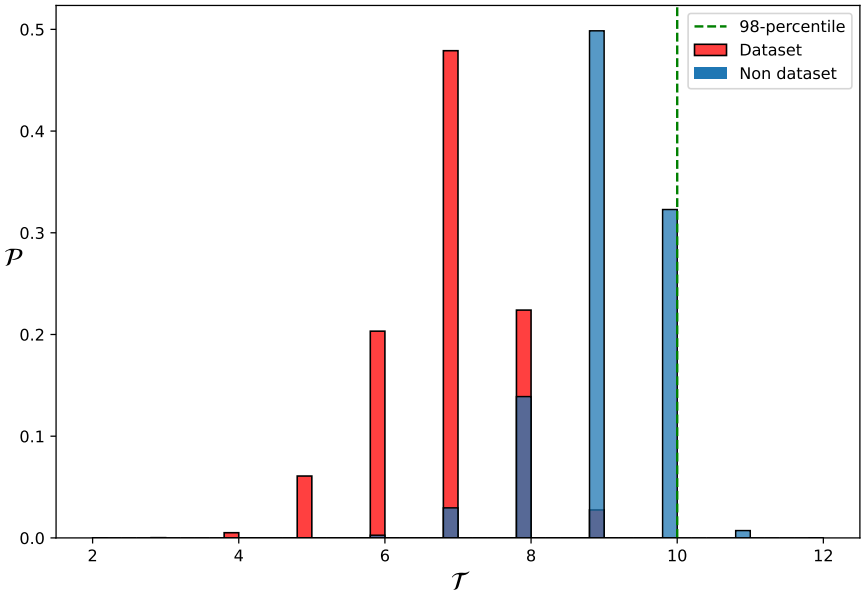
are the small functions, confirming the initial hypothesis that small functions introduce bias.

To provide a more numerical result regarding these three versions of equanimity, we calculate which percentage of the functions that exceed the threshold are outside the dataset:

- For the equanimity based on importance, **99.98** %.
- For the equanimity based on subsets, **97.07** %.
- For the normalized version of equanimity based on subsets, **37.13** %.

Among all the approaches, in the pursuit of finding a clear boundary between functions inside the dataset and the rest, the one that provides the best separation is the variable importance-based approach, followed by the non-normalized subset-based approach, although the separation is significantly reduced. It could be inferred that what determines a function's equanimity is not the subsets, but rather the variables that have the same effect on the function's output. It should be noted that this is an assumption, and another possibility could be that the subset-based approach requires further adjustment to accurately capture equanimity.

Finally, let's see how the corresponding graph would look like if we only consider the **entanglement** value.



Among all the metrics, this is the one that provides the best separation, as we can observe that the entanglement of all the functions in the dataset does no exceed the value of 10, which is the 98th percentile.

Note that even though 10 may seem like a large entanglement value, considering the maximum value is 12, the definition of entanglement selects the minimum value from a set of values. Therefore, increasing by one unit in larger values (e.g., from 10 to 11) is much challenging than increasing by one unit in smaller values (e.g., from 2 to 3).

From all the results obtained by studying the metrics separately, we can conclude that they all appear to be good indicators for determining when a function is computed by a large circuit, except perhaps for the normalized version of subset-based balance. It is crucial to recognize that the presence of an intersection does not invalidate the initial conjecture. The assertion that a function is computed by a super-polynomial circuit does not imply that it necessarily possesses a significant level of both metrics. What we can indeed deduce is that by combining both metrics, we will be able to observe a clearer separation between the two sets, thus empirically confirming the main conjecture we started with. We will study the combination of both metrics in the next section.

6.1.2. Testing metrics combined

A good way to assess how data is classified based on two parameters is by using **heat maps**: graphical representations of data where values are depicted using a color scale. The graphs we will display a significant amount of information that we need to clarify before presenting them.

- The x-axis represents the different values of equanimity. Since there are three versions, we will provide one heat map for each version of equanimity.

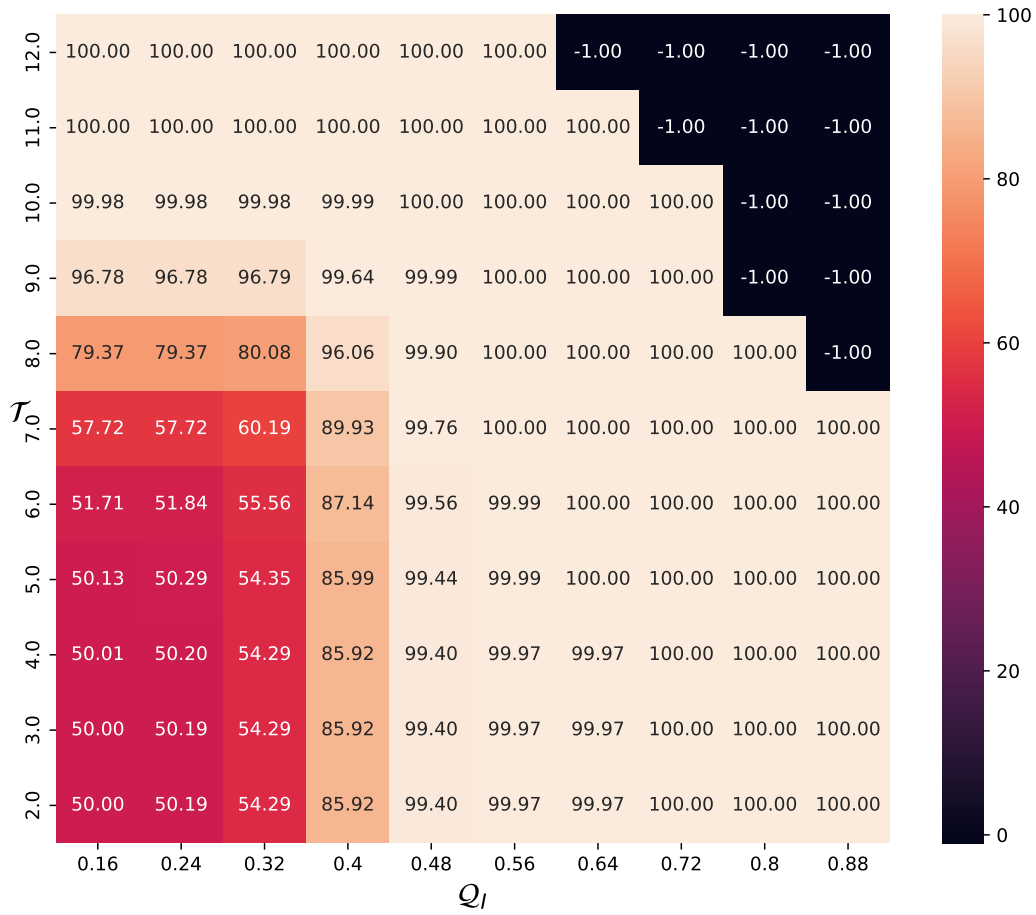
- The y-axis represents the different values of entanglement. These values range from 2, the minimum, to 12, the maximum.

- At each position in the matrix, determined by the equanimity, $\mathcal{Q}$, and entanglement, $\mathcal{T}$, values, we calculate the percentage of functions outside the dataset that have metric values $(\mathcal{Q}(f), \mathcal{T}(f))$ greater than or equal to $(\mathcal{Q}, \mathcal{T})$, from all the functions that have been analyzed and meet the same restriction. Formally,

$$M[\mathcal{T}, \mathcal{Q}] = \frac{|\{f \in \mathcal{F} \setminus \mathcal{D} : \ (\mathcal{Q}(f), \mathcal{T}(f)) \geqslant (\mathcal{Q}, \mathcal{T})\}|}{|\{f \in \mathcal{F} : \ (\mathcal{Q}(f), \mathcal{T}(f)) \geqslant (\mathcal{Q}, \mathcal{T})\}|} \cdot 100$$

  where $\mathcal{F}$ represents all the analyzed functions and $\mathcal{D}$ represents the functions in dataset. Note that the denominator can be 0, indicating there does not exist a function in $\mathcal{F}$ satisfying the condition. In such cases, we set it to -1.

Let's first examine the heat map when considering the equanimity metric based on variable importance.

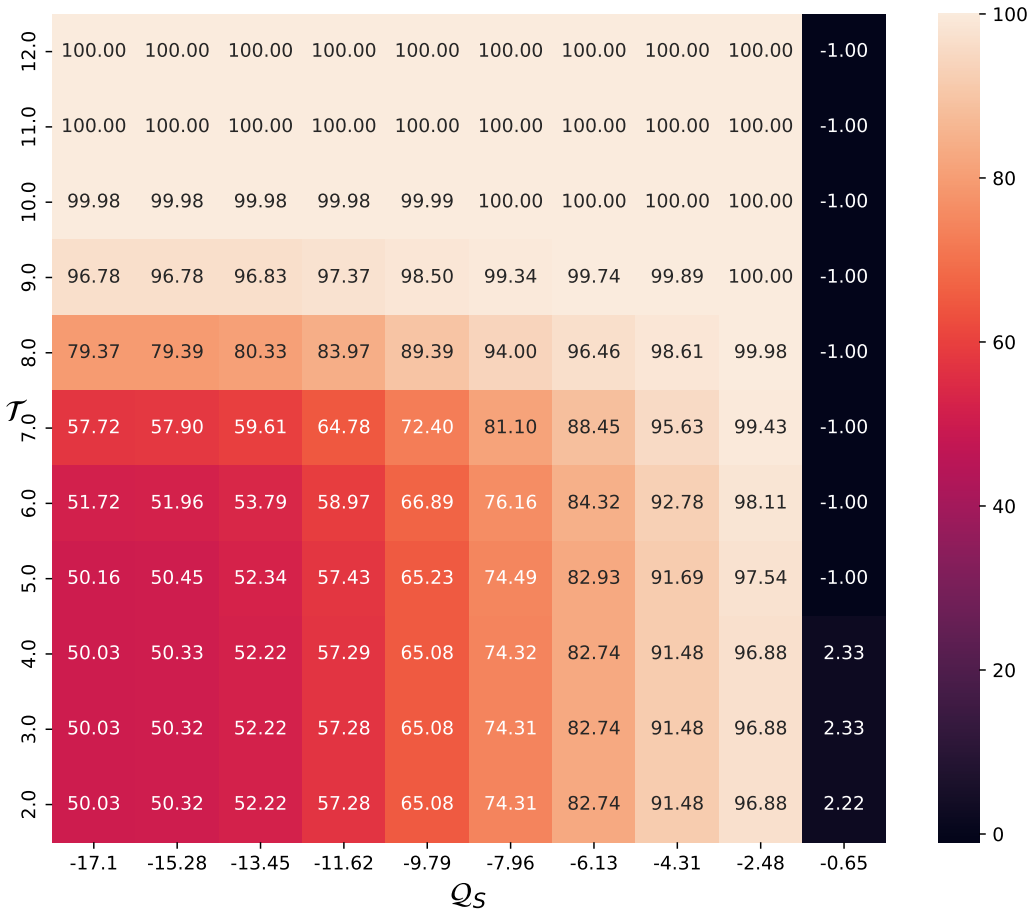


From here, we can deduce the following:

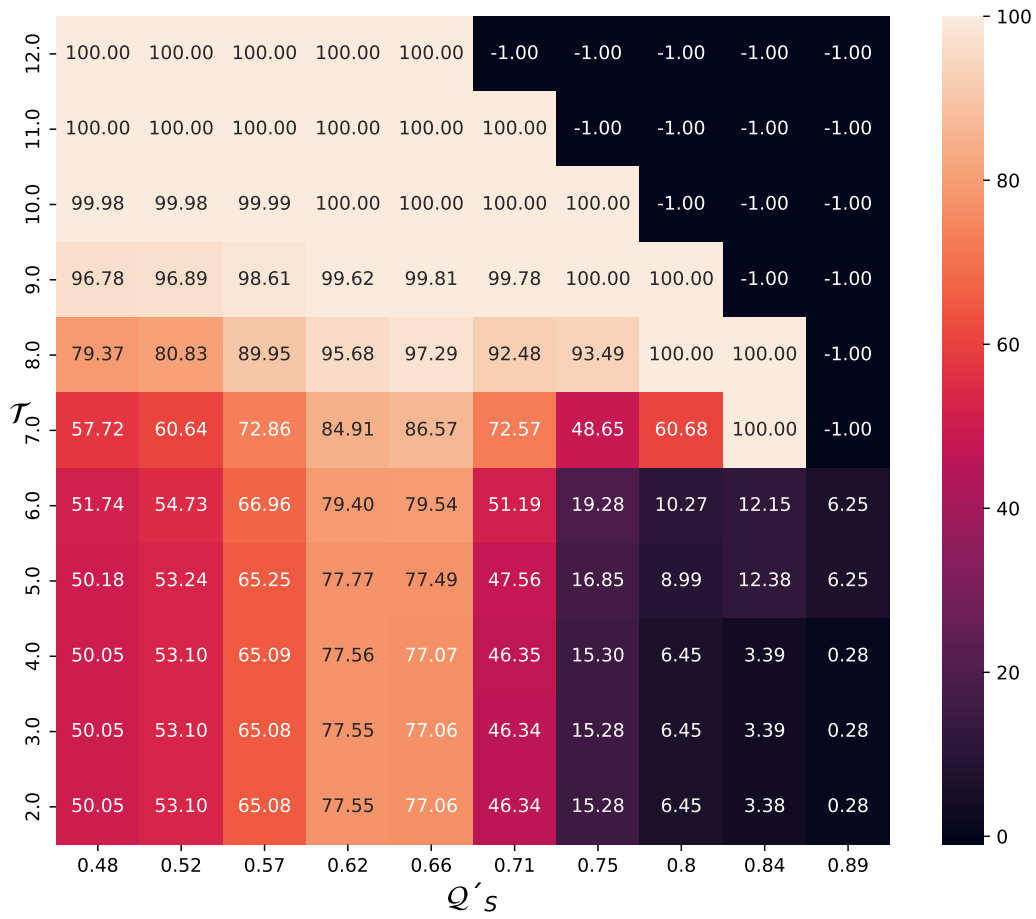
- When moving from left to right within each row, we can observe an increase in the percentage. This indicates that as equanimity increases, the percentage of functions outside the dataset also increases.
- The same can be said when moving from bottom to top with entanglement.
- The most interesting conclusion from this graph is that except for the lower-left square and the -1 values in the upper-right corner, only functions outside the dataset surpass the imposed constraints on equanimity and entanglement.
- Note that the -1 values appearing in the top right corner could indicate that finding highly equanimous and highly entangled functions is very challenging. In the search of identifying a mathematical property that can indicate when a function is not in $P_{/poly}$, as we previously mentioned, it cannot be a natural proof. As we have

seen, these metrics can be computed in polynomial time with respect to the input truth table, satisfying the first condition of a natural proof. The fact that finding a highly equanimous and highly entangled function is so difficult helps us bypass this definition of natural proof (in particular, its largeness condition), making it plausible to define that mathematical property based on these metrics.

Now let's examine the resulting graph when using the equanimity based on subsets.

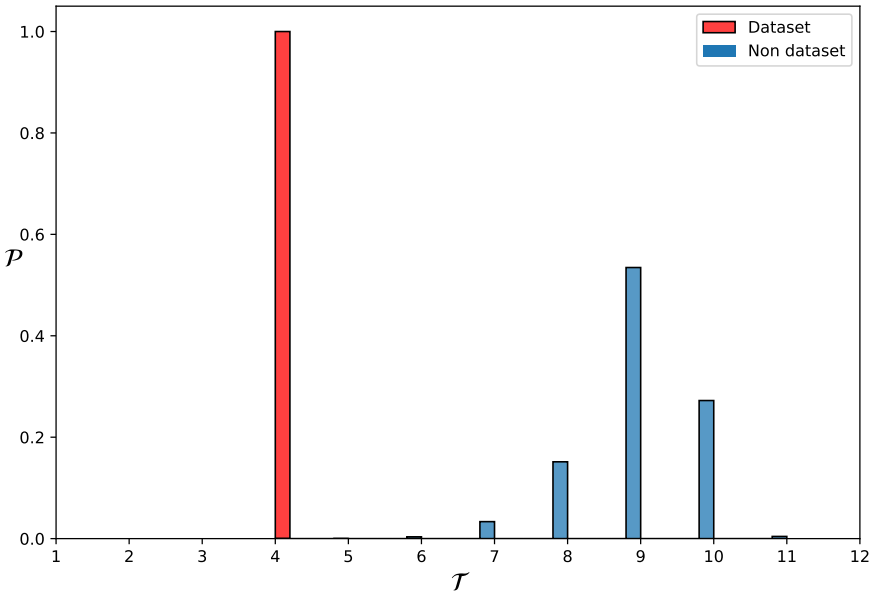


Here, an increase in the size of the square in the bottom-left can be observed. However, it still functions as a good indicator, leading to the same conclusions as in the previous case. A worst scenario is presented when we use the equanimity based on subsets normalized.

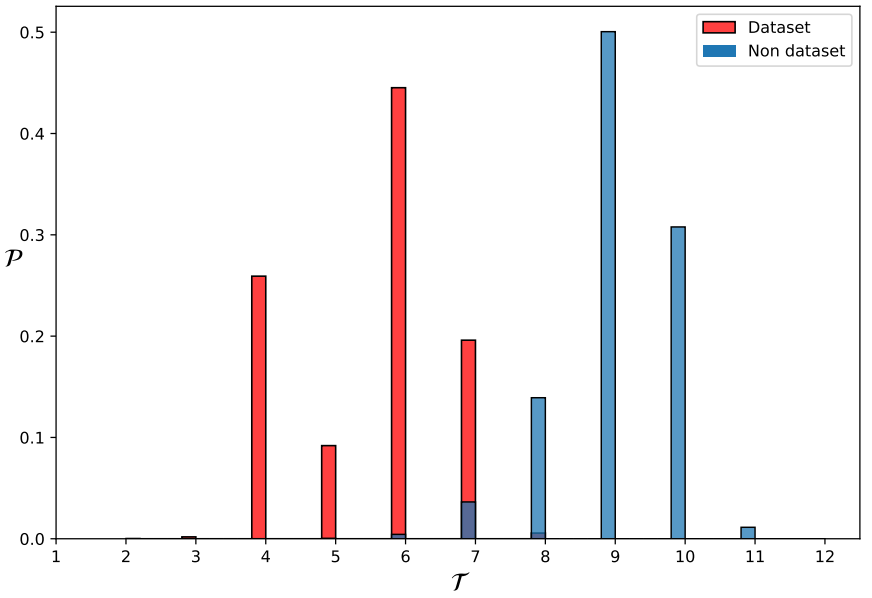


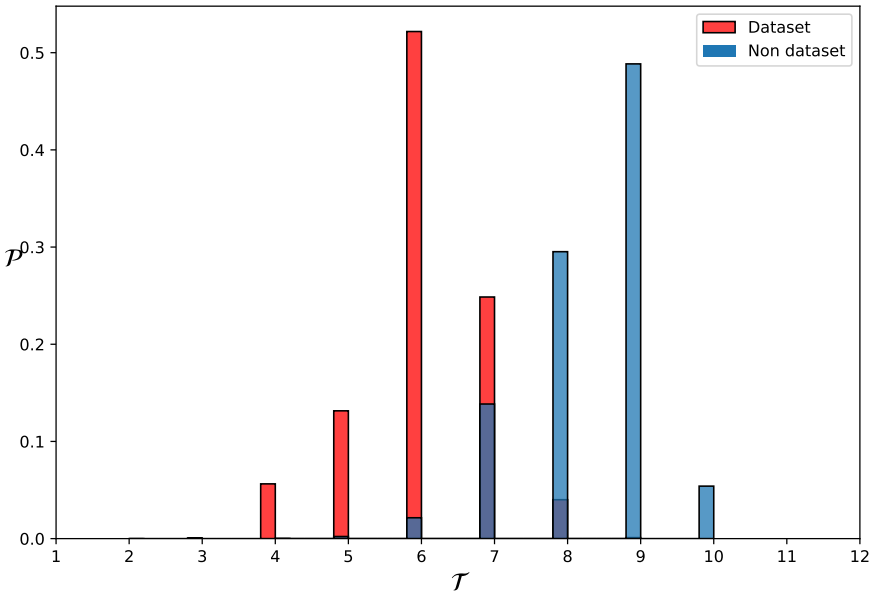
In this graph we see an unusual phenomenon that did not happen in the previous two cases. For entanglements less than 6, it no longer holds that as equanimity increases, the percentage also increases.

On the other hand, it is worth considering how the histograms for entanglement would look if we only take into account the functions that exceed the 98^{th} percentile of equanimity. For example, for equanimity based on variable importance, we would obtain the following graph.



Here, the separation is clear, and both sets are perfectly distinguishable. However, the graphs for equanimity based on subsets do not separate entirely both datasets and there is still some intersection, although is considerably smaller.





In conclusion, the results demonstrate that it would be possible to find a threshold using both metrics in order to classify whether a function is computed by a small or large circuit. It is worth noting that, based on the data obtained, the best definition of equanimity appears to be the one based on variable importance. In order to define a potential mathematical property of functions outside of $P_{/poly}$ based on these metrics, large bounds should be chosen for these values to avoid it being satisfied by a large proportion of functions. As observed, it is challenging to find a function that is highly equanimous and entangled.

6.2. Scalability of metrics

This experiment shows how our metrics perform as the input bits size increases. In order to so, we have calculated our metrics on 100,000 functions of n bits to 1, for $n \in \{4, ..., 10\}$. Besides, we obtained the value of our metrics on the decision problems introduced in Section 3.2[3].

Note that 100,000 represents a really small percentage of 2^n for $n \geqslant 5$, but calculating our metrics becomes increasingly challenging as n increases. This is due to the fact that many of our metrics have quadratic complexity in relation to the size of the truth table (which in turn is exponential with the number of input variables), as we have shown in previous chapters. Anyway, it will be a good indicator to understand the behavior of the decision problems.

[3]Let us recall the notation introduced in previous chapters: p_n for PARITY; m_n for MAJORITY; P_n for PRIMALITY; C_n for CLIQUE; C'_n for CLIQUE′

Before showing the results, we will define a new problem. As we explained in Section 5.5, the PRIMALITY problem (P_n) could be an example of a high-entangled, low-equanimous problem. With a similar reasoning we used to define CLIQUE′ (i.e., that combining a problem with PARITY makes it more equanimous) we can define the following problem:

Definition 6.2.1 (PRIMALITY′)**.** *Given a binary input, it returns 1 if:*

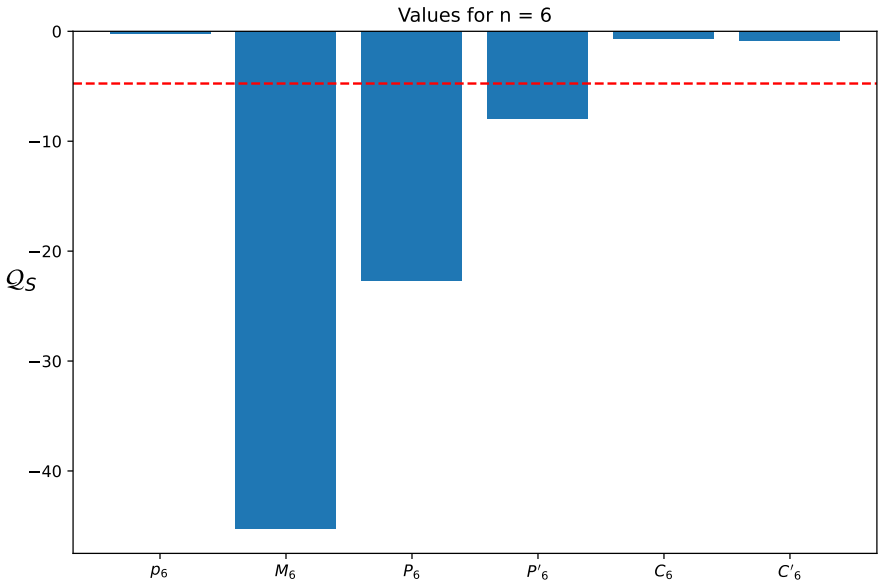
- *The number of ones is even and the corresponding number is prime*
- *The number of ones is odd and the corresponding number is not prime*

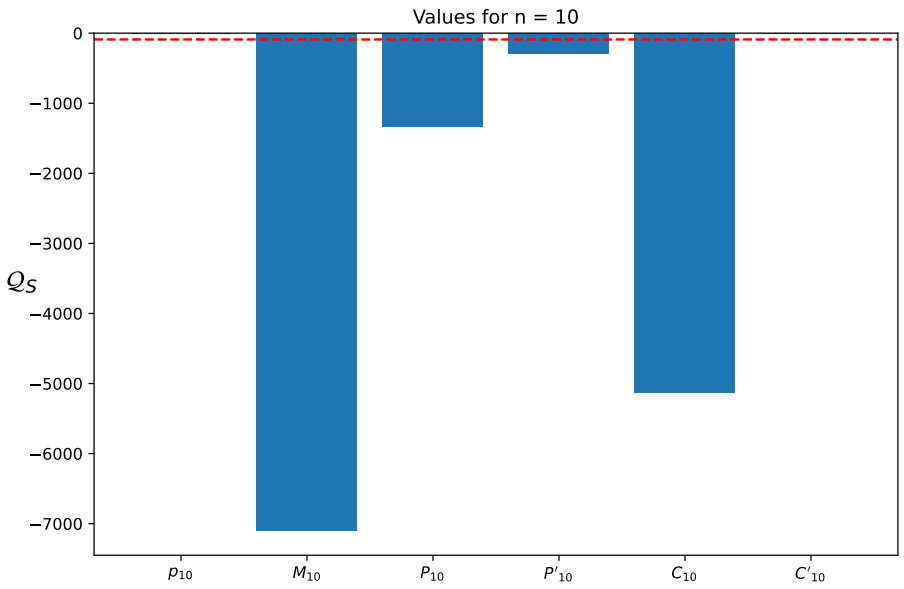
It can be represented as the family of functions $\{P'_n : n \in \mathbb{N}\}$

$$P'_n : \mathbb{Z}_2^n \longrightarrow \mathbb{Z}_2$$
$$x \mapsto P'_n(x) = p_n(x) \oplus P_n(x)$$

Note this problem belongs to $P_{/poly}$ (since it belongs to P), so if it was highly equanimous, since it seems to be entangled, it would go against our conjecture.

In the following figure, we will show how the chosen decision problems perform against the random sample of functions (with the equanimity based on the survival of subsets). To achieve this, we will compute the 98^{th} percentile for each metric within that sample of functions and examine whether the values of the selected decision problems fall below or above this threshold. In particular, we consider $n = 6$ and $n = 10$, since for other values of n a complete graph cannot be formed (for a graph of v vertices, a complete graph needs $\frac{v(v-1)}{2}$ edges).

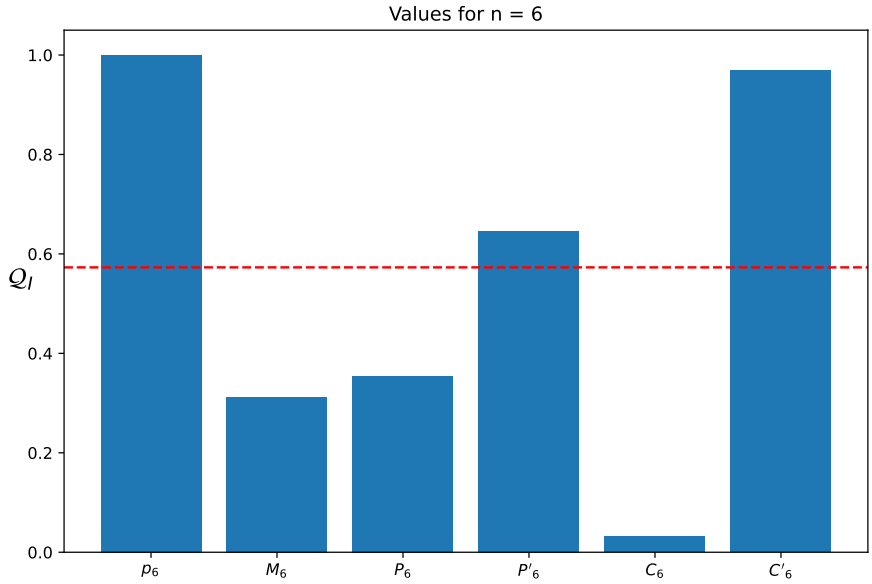


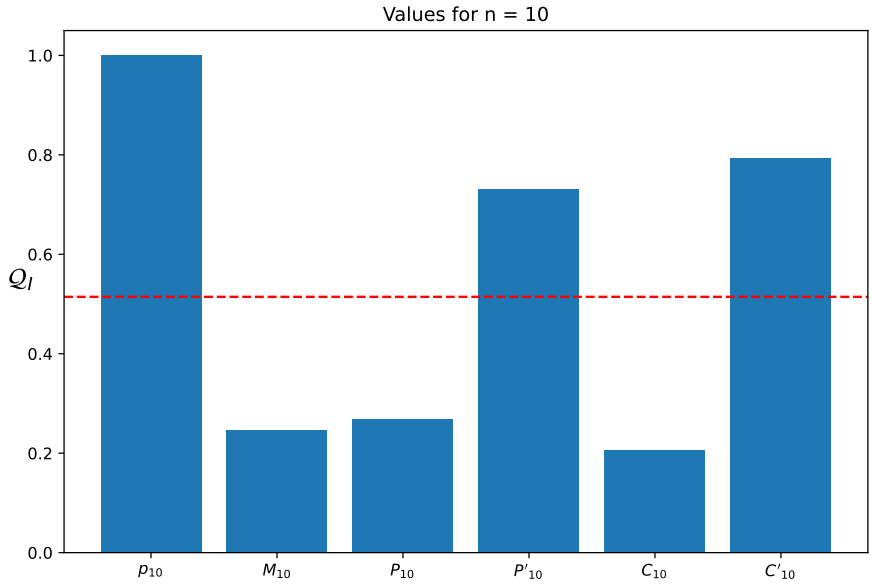


As we can observe, the equanimity value of CLIQUE′ is always above the threshold. In fact, in both cases its value is greater than the maximum value obtained from the random sample. This observation suggests CLIQUE′ is indeed a really equanimous problem.

It is evident that the unbiased nature of the PARITY problem has also influenced the PRIMALITY problem. However, although PRIMALITY′ gains equanimity, it is still not enough to surpass the threshold. Somehow, this *unbiasing* effect PARITY has is more effective on the CLIQUE problem, and this could be related to the inherent nature of the problem. Since CLIQUE is invariant under isomorphism, it becomes easier to enhance its equanimity since it already possesses the property of being unbiased in that regard.

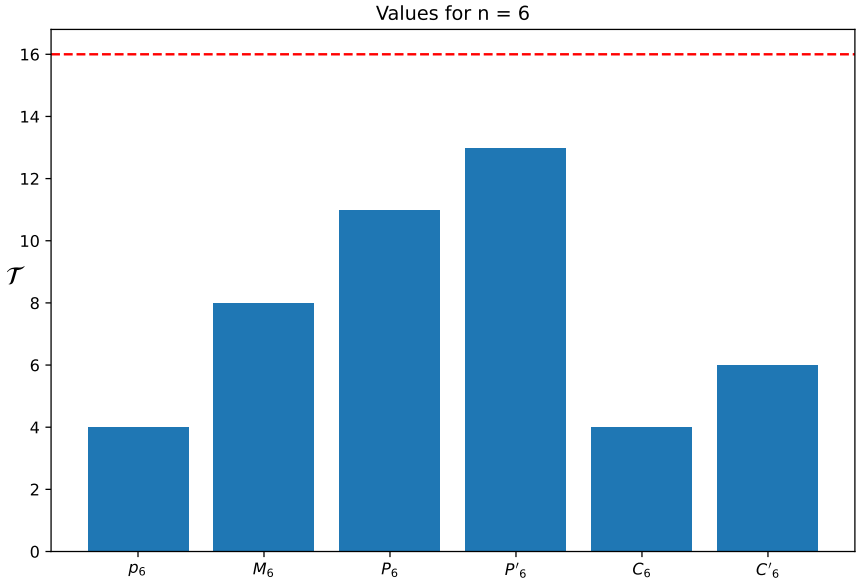
In the following graph, we display the results for the equanimity based on the importance.

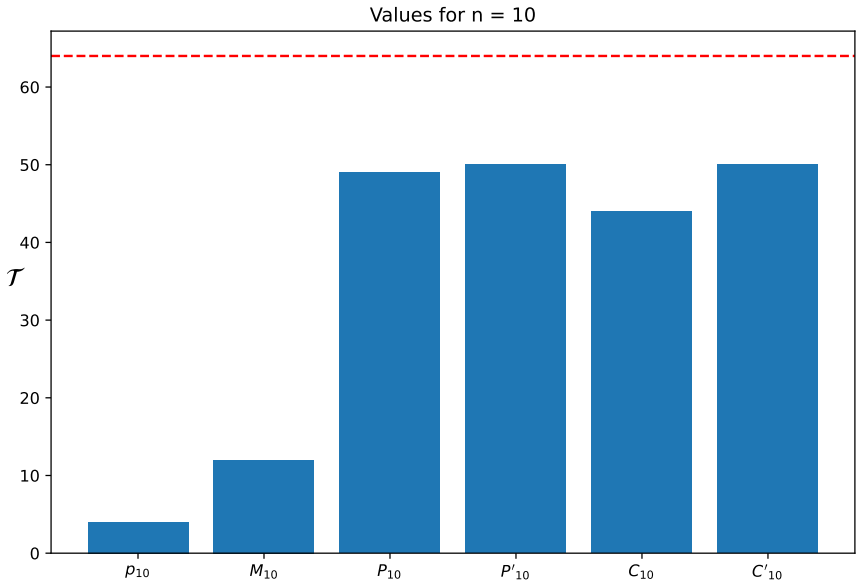




The conclusions that can be extracted are similar. However, although its value is inferior to the CLIQUE′, in this case PRIMALITY′ does surpass the threshold. As we can observe, the value at the 98^{th} percentile is not significantly high. This suggests that while this definition of equanimity may be useful for determining whether a function is equanimous or not in general terms, it might not be as effective in identifying extreme cases (which is our goal).

Let's see what happens with the entanglement metric:

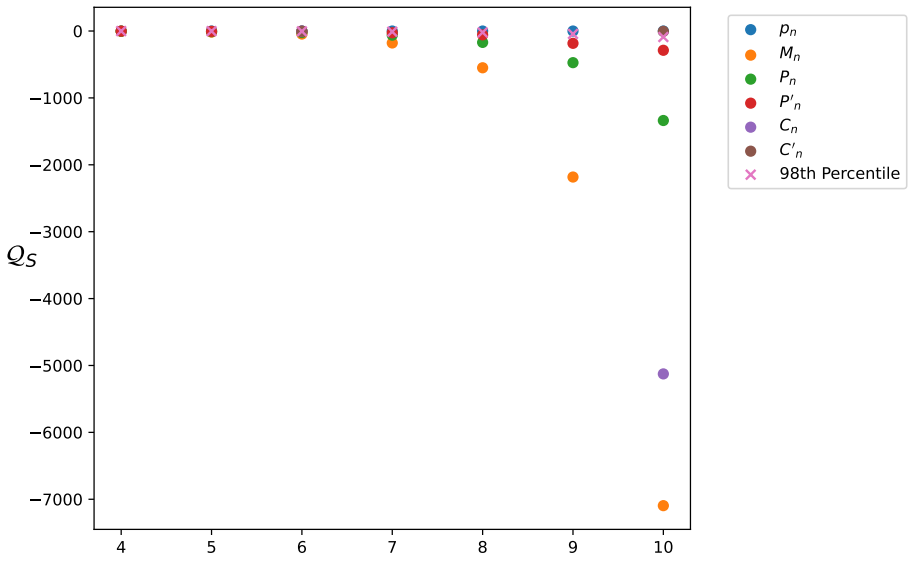




As it becomes evident, even though CLIQUE′ has a high level of entanglement, it does not surpass the threshold. Following our intuitions from previous chapters, keeping equanimity while becoming more entangled is gate-consuming. This is why it is not necessary that the problem we are looking for is totally equanimous and totally tangled, since we suspect that, for highly equanimous problems, there exists a certain value of entanglement

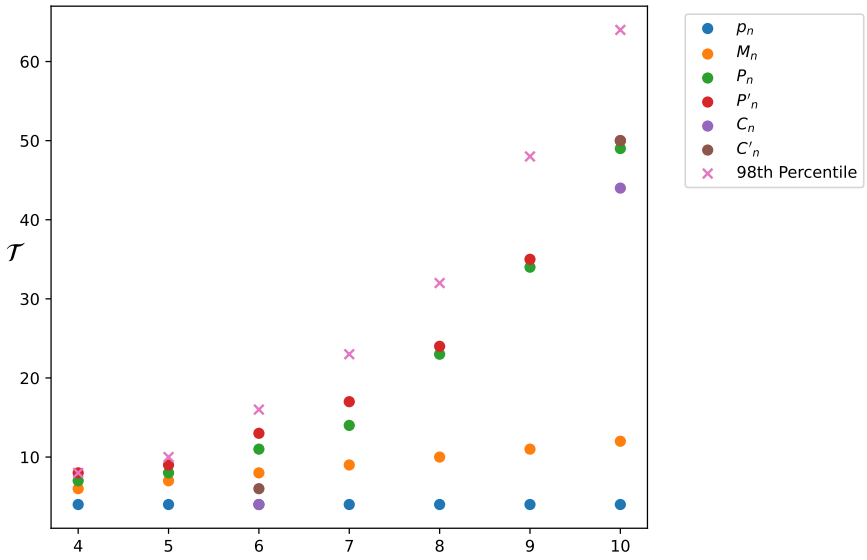
beyond which the circuit that computes the function cannot be polynomial in size (and this value does not have to be the maximum).

In the following graphics, we will observe the evolution of out metrics' values as n gets bigger starting with $\mathcal{Q}_S$.



Despite the fact that the range of possible values of $\mathcal{Q}_S$ notably expands within n, the value of the 98^{th} percentile remains relatively consistent. As we explained with the previous diagrams, CLIQUE′ always surpasses that threshold, while PRIMALITY′ seems to distance from it. The other decision problems also act as expected by distancing from the boundary. This indicates their lack of equanimity gets more noticeable as n increases.

Let's show now the graphic for the entanglement.



As we mentioned earlier, every decision problem falls below the value of the 98^{th} percentile. However, it is noticeable that the growth tendency of PARITY and MAJORITY is very different from the one CLIQUE and PRIMALITY have. We expect the growth of CLIQUE becomes more notorious as n increases than the growth of PRIMALITY. This is because the CLIQUE problem becomes more complex while the PRIMALITY problem becomes simpler as n increases (as a result, the information shared between the partitions will become more concise relative to n).

6.3. Mitigating bias in Boolean circuits

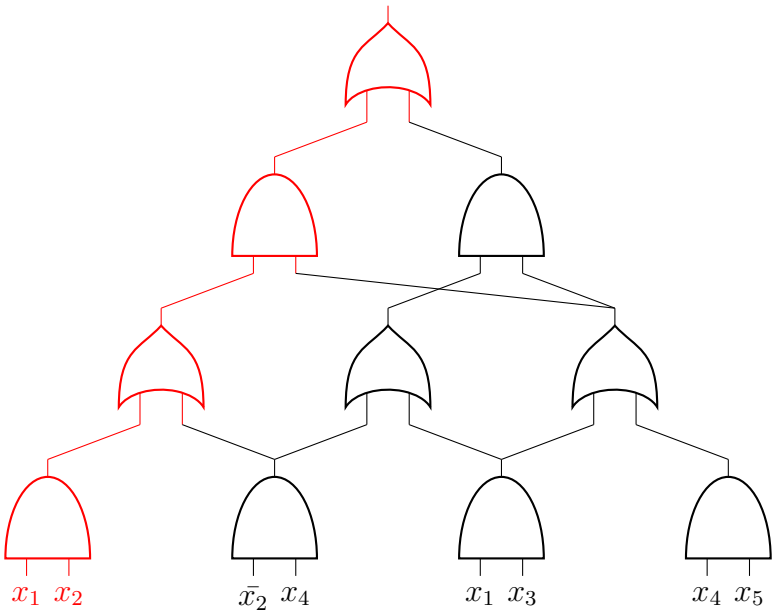
In this section we will study how Boolean circuits that are biased towards a certain subset of the input in the first level may (or may not) be capable of mitigating this bias and becoming more equanimous. We will test this behavior in Boolean circuits expressed in Alternating Circuit (AC) Form, defined in 3.2.2.

It is reasonable to think that a circuit in which a pair of inputs are connected with an AND gate in the first level is more likely to bias towards that pair in the function computed by that circuit rather than a circuit without that gate. Based on this, we will create random circuits that satisfy: i) x_1 AND x_2 is ALWAYS in level 1; ii) x_3 AND x_4 is NEVER in level 1. This should imply, based on our assumptions about equanimity, that functions computed by these circuits are almost always more biased towards $\{x_1, x_2\}$ than $\{x_3, x_4\}$ regarding the columns of its truth table where the output is 1.

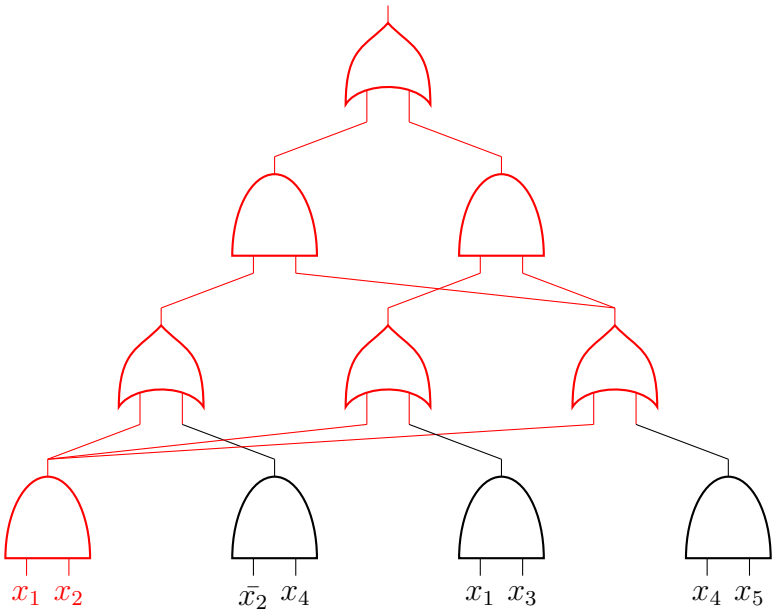
There are two extreme cases we take into consideration. In the first one, the output of x_1 AND x_2 is preserved in one branch until the last level, i.e., there exists only one gate

in level 2 whose input is the output of $x_1\ AND\ x_2$, and the output of this level-2-gate is the input of only one gate in level 3 (and so on). On the contrary, in the second one, we *poison* the second level with $x_1\ AND\ x_2$, i.e., every gate of level two has the output of $x_1\ AND\ x_2$ as one of its inputs.

Let's show an example of each case to visualize them, starting with the first case (the output is preserved in a branch).



As evident from the diagram, the gates influenced by the output of the gate $x_1\ AND\ x_2$, which are highlighted in red, are kept on a specific part of the circuit. Let's now see the other case, in which we *poison* every gate of the first level.



As we can observe, by injecting the output of $x_1\ AND\ x_2$ in every gate of the second level, it continues to propagate and influence every gate through the following levels.

With the aim of testing if our intuitions are accurate, we have generated two samples of functions computed by circuits that satisfy our restrictions (one sample for those in which we keep $x_1\ AND\ x_2$ in a branch and another for those in which we poison the first level) and consist of a maximum of 50 gates. The results of our analyses are outlined below.

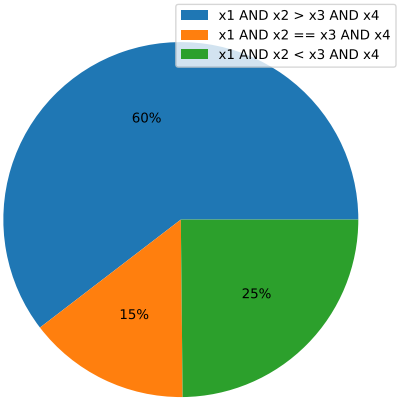
6.3.1. Keeping $x_1\ AND\ x_2$ in a branch

As we have introduced, these circuits keep $x_1\ AND\ x_2$ in a certain branch and the rest of the circuit is formed (almost) randomly — keep in mind x_3 and x_4 are never connected in the first level.— This was achieved by selecting a random number of gates for each level. One of the inputs of the first gate is connected to the output of the branch that computes $x_1\ AND\ x_2$, and the other input is selected randomly among the outputs of the inferior level. The inputs of the rest of the gates are also selected randomly, but in this cases the output of the branch that computes $x_1\ AND\ x_2$ is not considered.

Intuitively, the bias towards $\{x_1, x_2\}$ could be evened out in two ways:

- The part of the circuit that does not contain this branch should somehow connect x_3 and x_4 during the upper levels, thereby augmenting the influence of $\{x_3, x_4\}$.
- Introducing $\neg x_1$ or $\neg x_2$ into the branch, thereby diminishing the influence of $\{x_1, x_2\}$.

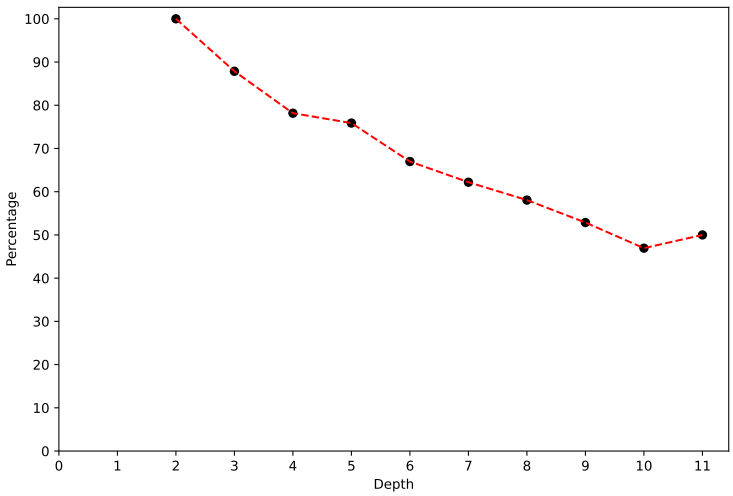
Let's show which percentage of the sample manages to mitigate the bias by comparing the proportion of circuits whose computed function returns 1 for input cases where each pair is respected more than the other pair.



As we expected, the bias towards $\{x_1, x_2\}$ is preserved in more than half of the cases. However, it is remarkable that 1 out of 4 circuits successfully shift the balance towards x_3, x_4, despite our efforts to prevent their connection at the first level. Therefore, there is an important factor to consider: depth.

Obviously, as a circuit grows deeper, it becomes more likely that it satisfies the requirements to even out the bias we pointed out before. If we were to solely consider the first level, the circuit would exhibit an inherent bias towards x_1, x_2 rather than x_3, x_4 (as one gate is consistently added while the other is not). Hence, it becomes imperative to incorporate additional levels to enhance the interconnections between outputs and achieve a more equanimous circuit.

In the graph below, we show how depth affects the percentage of times $\{x_1, x_2\}$ is more present in the output than $\{x_3, x_4\}$, i.e., the bias is not mitigated.



The bias decreases as the circuits grows deeper, as anticipated. In fact, there comes a point (around depth = 9) where x_1, x_2 ceases to be predominant in half of the instances.

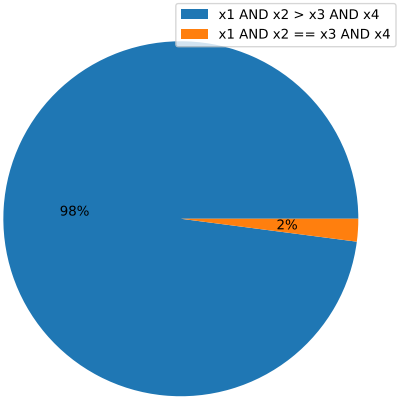
6.3.2. Poisoning the first level with $x_1\ AND\ x_2$

In this section, we will analyze the sample that contains the circuits in which every gate of the second level has $x_1\ AND\ x_2$ as one of its inputs and never includes $x_3\ AND\ x_4$.

Contrary to the previous case, reducing the influence of $\{x_1, x_2\}$ in this type of circuits is way more complicated, since it appears in almost every branch.

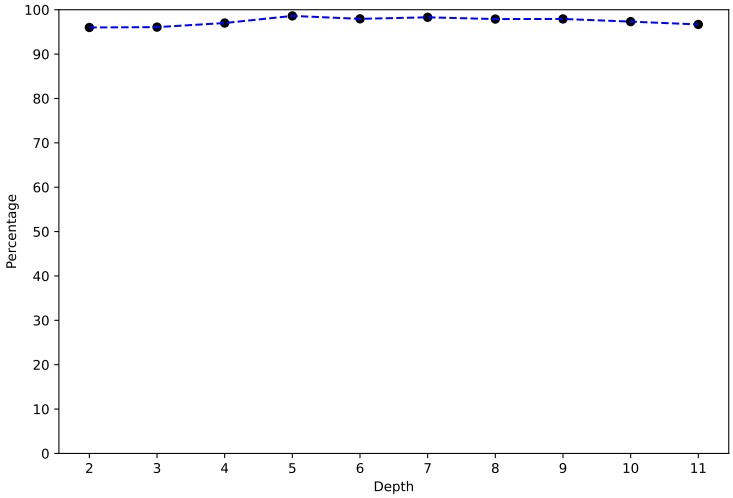
We can draw a parallel between this situation and an infectious disease. If we isolate some cases and treat them locally, it would be viable to stop it from spreading. However, once it has already spread extensively, reducing its impact becomes exceedingly challenging.

In the following graph we display the percentage of circuits of the sample that are able to mitigate the bias:



As we anticipated, it is almost impossible to mitigate the bias towards $\{x_1, x_2\}$ in this type of circuits. Furthermore, the balance is never shifted towards $\{x_3, x_4\}$.

In contrast to the previous scenario, not even growing deeper will solve the problem because more and more branches will be created and x_1 AND x_2 will have an influence in most of them. We appreciate this in the following chart:



6.3.3. Conclusions

In circuits in AC form, connecting two variables in the first level amplifies their impact on the output (while the opposite occurs if they are not connected). Based on the experiment conducted above, mitigating the bias towards those inputs is possible when keeping them in a branch since the bias diminishes along depth, and it is almost impossible if we do exactly the opposite.

The first approach is similar to the one utilized by the PARITY function. The standard XOR-based circuit that computes this function keeps each output of the first level on a single branch (and the equivalent circuit in AC form bounds the reach of each output of the first level to only two gates of each level of the circuit). This is how it manages to give each pair of inputs the same importance, being totally equanimous but very low entangled. As demonstrated, in order to achieve bias mitigation, circuits must possess a certain depth. This gives an intuition on why the PARITY function does not belong to the AC_0 class, which consists of polynomial-sized, constant-depth circuits.

The second approach shows that when a circuit is entangled by a specific pair, it is almost impossible to mitigate its bias. Intuitively, the only feasible method to counteract this effect would be to employ a similar strategy with the remaining possible pairs. This could be achieved by injecting every output of the first level into various branches, and since gates have fan-in 2, this would result in an increase on the number of gates. Something similar would happen in the levels above.

In conclusion, an entangled circuit needs to keep growing in order to assure equanimity. This is why our conjecture suggests that, for an entangled function to achieve equanimity, it would require a large (ideally superpolynomial) number of gates.

Chapter 7

Conclusions and future work

The study conducted on Boolean functions in this text aims to make a useful contribution to Computational Complexity. The results obtained so far show that, although ambitious, it seems plausible to provide a deeper analysis on what makes a function computable with either small or big circuits based on our metrics. Since the conception of circuit complexity, there have been scarce advancements in this field. Therefore, any progress is interesting by itself.

Throughout this text, we have explained and measured certain properties (equanimity and entanglement) that have the potential to make circuits superpolynomial in size. These properties are significant in the context of establishing lower bounds for circuits, as they provide insights into the inherent complexity of computational problems. As we have explained, establishing these lower bounds is a key aspect of tackling the P vs NP problem. The initial conjecture, which stated that a problem with high values of entanglement and equanimity must be computed by a circuit with high number of gates, was supported by the experimental results displayed on Chapter 6. As it seems to be promising, we propose several paths for further development of this approach.

Firstly, the metrics could be refined and perfected. One could think that, since there appears to be a correlation between entanglement and number of required gates, a promising way to find lower bounds would be to show inductively that circuits with a certain value of entanglement cannot be computed by polynomial-sized circuits, and then showing an NP-Complete problem (CLIQUE′ for example) which has at least that value of entanglement. However, as we anticipated in the first chapters, a strategy along these lines is not likely to succeed since it would be considered a natural proof. This is why we think that, even though our metrics seem to work fine for small values of n, they must be defined in a more complex way in order to capture more accurately the desired behaviors. For example, the entanglement could be calculated recursively, i.e., for each partition, obtain the local entanglement of each subset, and so on. This would highlight the fact that a very tangled function is tangled at every level.

Secondly, we could study the equanimity more in depth by analyzing potential biases that can arise at different gate counts in circuits. To achieve this, we can proceed as follows. For $k \in \mathbb{N}$, take every Boolean function that can be computed by a circuit of at most k gates. For each function, for every subset of input variables of a certain size obtain all the possible values c_{x_S} and add them into a list. Then sort each list in either ascending or descending order and add them into a set, denoted A_k. Note that, since functions computed by circuits of the same size are expected to exhibit similar bias structures, the number of elements of A_k should not be large. The ultimate goal would be to identify a certain invariant that evidences how the bias pattern becomes more complex as k grows.

Following the line of the first proposal and as a possible critique to the conjecture proposed, one might wonder if the equalizing effect that PARITY has on CLIQUE could be transferred to another simple and entangled problem; as we saw in Section 5.5, PRIMALITY fits this description. In fact, this is what happens, as we saw in Section 6.2. This potentially contradicts the conjecture we put forward. However, as we observed in the experiment, the equalizing effect provided by PARITY in PRIMALITY is not as effective as the one provided in CLIQUE. What we can conclude is that the trick would lie in determining sufficiently high bounds that avoid both the definition of natural proof and the identification of PRIMALITY as a super-polynomial function, while keeping CLIQUE' within those bounds. Furthermore, perhaps PRIMALITY is not as entangled as we assumed.

During the process of studying the defined metrics, we have realized that, in order to obtain a promising approach to tackle the P vs NP problem based on the proposed ideas, it would be necessary to utilize various branches of discrete mathematics. Perhaps the most prominent branch would be combinatorics, due to the nature of the definitions and the proofs presented in the text. Counting techniques are needed to estimate the growth of circuits that compute Boolean functions as the complexity increases. Besides, knowledge of set theory would also be necessary, as it relates to counting combinatorics. Determining applications that exhibit injectivity or even bijectivity would be crucial in establishing relationships between the cardinalities of both sets. Finally, with the last experiment proposed as future work — the one involving the A_k's — we believe it could be interesting to apply Number Theory to determine a possible pattern that these vectors satisfy in order to provide further insights into how these Boolean functions behave with respect to the metric of equanimity.

Personal contributions

Following this, each of the authors will outline their personal contributions to the project. However, we would like to emphasize that the vast majority of the work has been developed collaboratively, and there is no part of the project that has been exclusively developed by any one of us.

Enrique Carro Garrido

The project began with a research phase, aimed at familiarizing ourselves with the concepts of circuit complexity that have been frequently mentioned throughout the exposition of the work, as well as the new concepts explained to us by our tutors. These new concepts — equanimity and entanglement — were gradually modeled throughout the work to fit our intended capture. In particular, I realized that it was not feasible to consider all possible partitions for entanglement since it would significantly reduce the range of the metric, as mentioned in Section 5.2.

During this phase, our working philosophy consisted of distributing the readings among ourselves in order to achieve a broader spectrum of knowledge, and then gathering them to share the acquired insights. I specifically had to study Celia's work [RM22] in order to understand the demonstrative techniques she presented at the end of her projects as future work.

I was the creator of the Git repository [CC23]. In this repository we implemented the code for the metrics described in Sections 4.2.1 and 4.3.2. There can also be found the experiments described in Chapter 6. There has not been any part of the code that we did not work on together, but eventually each of us has contributed more significantly to certain parts of that repository. In particular, I took more responsibility in the implementation of the entanglement metric and the benchmarks.

Regarding the experiments developed in the project, it is impossible to indicate who contributed more significantly as we gathered in study rooms to conduct them. The work

in these study rooms consisted in sharing ideas using the room's whiteboard, considering them valid or providing critiques. We greatly appreciated the opportunity to work together as it allowed us to develop a greater number of ideas than would have been possible working individually.

There has been a greater division of tasks in the writing of the report, although everything that has been written by one of us was meticulously reviewed by the other to avoid typos and improve the writing as much as possible, many of them have been significantly modified by José. Therefore, it should be understood that I wrote the core content of the section and not that I completed the entire part. Similarly, I have made a smaller contribution to the sections he has written.

In Chapter 2, I was solely responsible for writing Section 2.2 in the different proof techniques that have been used in the past to attempt to solve the problem P vs NP. On the other hand, I did contribute more in Chapter 3, particularly in Section 3.1, where I give a brief prior explanation of both concepts, and Section 3.3 on the descriptions of the different experiments that have been carried out. However, I had less influence on Section 3.2 where I only provided Theorem 3.2.1, showing that CLIQUE' is NP-Complete.

Later on, in Chapters 4 and 5, the collaborative work between both of us is further intensified. The complexity of the definitions and the desire to fully capture both abstract concepts have perhaps been the slowest and most progressive process of all the project. The formalization of the concept came after trial and error with experiments until they aligned with our intended objectives. Subsequently, we were able to develop the theoretical results presented in the final sections, which validate our initial intuitions.

Therefore, what we did was formalize everything before writing it in the report and then distribute the results among ourselves to write them in the text. In Chapter 4, I took care of providing introduction to the concept, Section 4.1 and I also wrote the alternative to the subset-based definition of equanimity in Section 4.3.1 and Theorem 4.4.1 on the limits of equanimity. On the other hand, in Chapter 5, I took charge of writing Section 5.1, which introduces the concept of entanglement, and its formal definition in Section 5.2, except for Theorem 5.2.2 that José handled. Regarding Section 5.3 on properties of entanglement, I wrote Property 5.3.2. I also wrote Section 5.5 discussing the need to continue considering equanimity.

The distribution of work in Chapter 6 has been more separate. In particular, I took charge of Section 6.1, which describes and presents the results of the first experiment on testing the metrics using José's *dataset*. It is also worth mentioning that once we completed all the experiments, it occurred to me that we may not have considered the equanimizing effect that PARITY had with CLIQUE could also be applied to PRIMALITY. Therefore, we had to add this benchmark as well, realizing that it did indeed exhibit a significant level of equanimity, although not as much as CLIQUE' does.

Finally, the chapter on conclusions and future work was written collaboratively, although if we had to assign specific parts to each of us, I would take responsibility for the future work section, while my partner would be in charge of the conclusions part.

Setting aside the report, I have been the one with the most communication with the tutors through email. However, it is important to note that all messages were agreed upon beforehand between the two team members. I want to emphasize that the work has been done together, there has not been any person who has done more work that the other, and anything that I have stated as my own contribution has been highly influenced by my colleague José.

José Ramón Cobián Fernández

At the beginning of the project, I read in detail the sixth chapter of the book *Computational Complexity: A Modern Approach* [AB09] which explains how Boolean circuits are related to the $P \neq NP$ problem and why this relationship is important. At the same time, I was reviewing the bachelor's theses by Celia Rubio [RM22] and Jorge Villarrubia [VE21], since they were about the same topic and their conclusions could serve as a starting point for our work.

Then, after some meetings with our tutors, we started to formalize and experiment with the concepts of equanimity and entanglement. This process involved extensive brainstorming and in-depth discussions of ideas. This particular aspect of the project was highly collaborative, given that the concepts in question were abstract in nature and constituted a critical component of our work. Hence, it was imperative to ensure their definition was as accurate as possible.

In order to test them experimentally, we had to implement the code for calculating the metrics associated with the defined concepts. I programmed the files that solved the decision problems MAJORITY and PRIMALITY, introduced in Section 3.2. I also developed the code that implemented each definition of the equanimity metric. This required revisiting and adjusting the code, since each approach was perfected over time. Additionally, there were some approaches we created and then discarded as they did not reflect the expected behaviors.

Regarding the writing of the report, I took responsibility for Chapter 1, which serves as an introduction to the topic. In this chapter, we present our objectives, outline our work plan and provide an overview of how the project is structured, including the contents of each chapter. Furthermore, I wrote sections 2.1 and 2.3, in which we introduce essential definitions necessary to understand the P vs NP problem and circuit complexity, as well as explaining how they are connected. This required extensive research to find the most

relevant and up-to-date references and information related to the topic. Following that, I composed a significant portion of Section 3.2, where we introduced the chosen decision problems and their relevance. This included developing most part of the definitions and the proof of Lemma 3.2.1, which states the Half CLIQUE problem is NP-Complete.

In chapters 4 and 5, which are the theoretical core of the project, we collaborated in formulating the definitions and results. In particular, I conceived the idea that it could be interesting to consider the importance of variables in order to define equanimity. This was the starting point of the creation of the equanimity based on the importance of each variable introduced in Section 4.2. Furthermore, I observed that the entanglement values of PARITY and MAJORITY followed a certain pattern as n increased, which led to the inclusion of Theorems 5.2.1 and 5.2.2. Regarding the proofs, we came up together with the main ideas and then distributed the work to formalize them. I took charge of those concerning Proposition 4.3.1, Theorems 4.4.2 and 5.2.2, and Lemmas 5.3.1 and 5.2.1.

Moreover, I had a major influence on the developments of sections 4.2.1, 4.3.2 and 5.4 regarding the implementation of the metrics. We divided the inclusion of the definitions into the report, and I took responsibility for those related to equanimity, which are documented in Sections 4.2 and 4.3 (except for the alternative to the definition covered in Section 4.3.1, which was written by Enrique). Besides, I selected and included suitable examples to enhance the comprehension of these definitions.

With respect to the experiments, we selected which experiments to conduct and analyzed their results together, whereas we divided the implementation of the required code. I wrote the code used to obtain the samples used in the experiment *Scalability of metrics* explained in Section 6.2 and the one that generated the circuits used in Section 6.3.2. Then, I created the Python code needed to obtain the graphs displayed in Section 6.2 and in Section 6.3. Lastly, I wrote the concepts of those two sections, which reflect the results and conclusions derived from our collaborative analysis.

Finally, we agreed on which were the most significant conclusions we derived and determined the approach for future work. In the corresponding chapter, Chapter 7, I was responsible for the paragraphs that focus on the conclusions, while Enrique explained how the work could be further advanced. It is also worth mentioning that when our tutors provided feedback on specific sections of the project, I was in charge of revisiting those parts and refining them based on their comments and suggestions.

Putting the report aside, my responsibility involved thoroughly reviewing and providing comments on the code contained within the repository. This included testing every functionality to ensure every piece of code was performing as expected and the data that was being generated and used in our experiments was correct. Furthermore, I made adjustments to align the names of functions and variables with those utilized in the report. Additionally, I added comments preceding each function to explain their purpose and functionality.

In conclusion, I want to highlight the high level of collaboration that defined our work. Throughout the project, every aspect that required the creation of new definitions and results was a collaborative effort aimed at ensuring their quality and accuracy. This is why, although we divided the technical and writing aspects of the report between us, this work is the result of mutual effort.

Bibliography

[AB09] S. Arora, B. Barak: *Boolean circuits*. En *Computational Complexity: A Modern Approach*, pp. 106–122. Cambridge University Press, 2009. https://doi.org/10.1017/CBO9780511804090.009

[AS16] S. Aaronson: $P \stackrel{?}{=} NP$. *Open problems in mathematics*, Springer, 2016, pp. 1–122. https://link.springer.com/chapter/10.1007/978-3-319-32162-2_1

[BGS75] T. Baker, J. Gill, R. Solovay: *Relativizations of the P=?NP Question*, SIAM Journal on Computing, 4(4):431–442, 1975. https://doi.org/10.1137/0204037

[CF23] J. R. Cobián Fernández: *Generación de circuitos para experimentos de complejidad*. Trabajo de Fin de Grado en Matemáticas, Departamento de Sistemas Informáticos y Computación, Universidad Complutense de Madrid, 2023.

[CC23] E. Carro Garrido, J. R. Cobián Fernández: *Repositorio del presente trabajo*, 2023. https://github.com/encarro12/TFG-Circuit-Complexity.git

[Had96] J. Hadamard: *Sur la distribution des zéros de la fonction $\zeta(s)$ et ses conséquences arithmétiques*. *Bulletin de la Société mathématique de France*, vol. 24, pp. 199–220, 1896. http://www.numdam.org/item/10.24033/bsmf.545.pdf

[Hå14] J. Håstad: *On the correlation of parity and small-depth circuits*, SIAM Journal on Computing, vol. 43, no. 5, pp. 1699–1708, 2014. Publisher: SIAM. https://epubs.siam.org/doi/abs/10.1137/120897432

[Ho09] R. Horodecki, P. Horodecki, M. Horodecki, and K. Horodecki, *Quantum entanglement*, *Reviews of Modern Physics*, vol. 81, no. 2, pp. 865, 2009. https://journals.aps.org/rmp/abstract/10.1103/RevModPhys.81.865

[Po35] T. Popoviciu: *Sur les équations algébriques ayant toutes leurs racines réelles*, *Mathematica*, vol. 9, no. 129-145, p. 20, 1935.

[RM22] C. Rubio Madrigal: *Búsqueda de funciones booleanas complejas: construcciones mediante monotonía y relación entre repetitividad y endogamia*, 2022.

[RR94] A. A. Razborov, S. Rudich: *Natural proofs*, Proceedings of the twenty-sixth annual ACM symposium on Theory of computing, pp. 204–213, 1994. https://dl.acm.org/doi/pdf/10.1145/195058.195134

[RW93] A. Razborov, A. Wigderson: *nΩ (log n) lower bounds on the size of depth-3 threshold cicuits with AND gates at the bottom*, Information Processing Letters, vol. 45, no. 6, pp. 303–307, 1993. Publisher: Elsevier. https://www.sciencedirect.com/science/article/pii/0020019093900417

[Sha49] C. E. Shannon: *The synthesis of two-terminal switching circuits*, The Bell System Technical Journal, vol. 28, no. 1, pp. 59–98, 1949. Publisher: Nokia Bell Labs. https://ieeexplore.ieee.org/abstract/document/6771698/

[Tur36] A. M. Turing: *On Computable Numbers, with an Application to the Entscheidungsproblem*. Proceedings of the London Mathematical Society, s2-42(1):230–265, 1936. https://doi.org/10.1112/plms/s2-42.1.230

[VP96] C. de la Valle-Poussin, *Recherches analytiques sur la théorie des nombres premiers*, *Ann. Soc. Sc. Bruxelles*, 1896. https://cir.nii.ac.jp/crid/1573950400259760768

[VE21] J. Villarrubia Elvira: *Identificación experimental de las funciones booleanas que requieren circuitos extensos y aplicación al estudio de P vs NP*, 2021.

[Wig06] A. Wigderson: *P, NP and mathematics–a computational complexity perspective*. Proceedings of the ICM, vol. 6, pp. 665–712, https://www.math.ias.edu/~avi/PUBLICATIONS/MYPAPERS/W06/w06.pdf