Counting Problem of *PureCircuit*

Christos Demetriou *Department of Computer Science*
University of Warwick
Coventry, UK
u2018918@live.warwick.ac.uk, u2018918

Abstract

The current paper is an investigation of the *#PPAD(PureCircuit)* problem under counting separations with respect to other counting problems. We build upon the paper [1], which derived tools that separate counting problems from one another and showed that several equivalent problems can lead to a counting hierarchy under relativising reductions and oracle separations, when augmented with counting separations. More precisely, the goal is to extend that hierarchy, by introducing counting separations to the counting version of the *PureCircuit* problem. Moreover, we aim to build a tool to visualise the problem and aid the development of the theorems. Lastly, we conjecture the possibility of Cook-Levin-type of theorem to characterize the upper limit of **#PPAD**-hardness.

Index Terms

Complexity Theory, Counting Complexity, Relativization, PPAD, Search Problems

I. Introduction

A. Background

In the world of computational complexity, there are a variety of problems that have an impact on our daily lives. The most notorious of such problems is known as **P** vs **NP**, which is part of the millennium problems [2], [3]. It asks the question of: can we solve problems in a polynomial amount of time if their solutions can be verified in a polynomial amount of time? This problem is notorious due to its difficulty, as it is one of the easiest problems to define but one of the hardest to solve, as well as having an impact on many fields such as protein folding [4], cryptography [5], game theory [6] and many others. There are various approaches that are encompassed under the umbrella of meta-complexity where researchers are trying to find better lower bounds for **NP** problems through the lens of circuit complexity [7]–[9], proof complexity [10] and many other types in order to reveal a separation between the two complexity classes. But interestingly, making progress towards this problem is a difficult task, as there are several barriers, as shown by Razborov [11] (Natural Proofs Barrier), Baker–Gill–Solovay [12] (Algebrizing proofs) and Baker [13] (Relativising proofs) that can slow down the progress off **P** vs **NP**.

On the other side, people have added extensions to **P** and **NP** classes, from decision problems into search problems where instead of checking the existence of a solution we return the solution itself, also known as **FP** and **FNP** classes. An interesting subset of these problems, developed by Papadimitriou, is known as **PPAD**, which is a class of problems that are guaranteed to have a solution due to some combinatorial property of the problem [14]–[16]. Motivation for such problems is due to their inherent difficulty to be computed optimally but guaranteed to have a solution as well as several connections with other fields such as economics [6], [14], [15], [17].

Interestingly, a class of languages was built on top of the **NP** class, which is known as **#P**. This is the class of problems that counts the number of solutions to polynomially verifiable problems. Valiant introduced this extension to demonstrate that while finding a solution can be computationally efficient, counting the total number of solutions can be significantly harder [18]. This theory opened the floodgates for many other discoveries, such as Toda's theorem [19], holographic reductions [20], statistical physics [21] and connections with geometric complexity theory [1], [22].

B. Problem statement

Curiously enough, a lot of counting problems that seem to be in **#P** seem to behave differently than their decision counterparts. Specifically, as demonstrated in the paper [1], some counting problems are much harder than others, as observed in the **#PPAD** hierarchy, where several problems despite being **PPAD**-complete, their counting counterparts with counting separations can form a hierarchy. Additionally, they were able to create a mathematical framework that takes advantage of representation theory and algebraic topology to deduce whether a language is contained in **#P** class. In the current paper, we use the above observation to investigate a different **PPAD**-complete problem, known as *PureCircuit*. This was a problem that was created to argue the inapproximability bounds of the **PPAD** class [23], [24]. Due to its circuit structure, it seems to offer much more powerful representations than the other **PPAD**-complete problems. We aim to find a robust definition of **#PPAD**(PureCircuit) with its counting separations, as well as find parsimonious reductions to the other counting problems or any possible oracle separations. We believe that the flexible definition of the class will have more degrees of freedom, and conjecture the possibility of a Cook-Levin-type theorem [2] to demonstrate **#PPAD**-hardness.

Lastly, due to the definition of *PureCircuit*, we can observe that it resembles very closely another structure in the world of meta complexity, known as hazard-free circuits [24]. A possible research direction would be to analyse the correlation between the two circuits and potentially uncover any hidden statements for either of the two, as already pointed out [24].

II. Aims and Objectives

The core concept of the project is characterized under the following research question:

*Is there a counting separation of the **#PPAD**(PureCircuit) where we can develop a Cook-Levin-type theorem to describe the upper limit of **#PPAD**(PureCircuit)-hardness?*

Based on that we enumerate the following objectives:

1) Create a robust definition for **#PPAD**(*PureCircuit*) and several of its variants like **#PPAD**(*PureCircuit*) $-$ 1 **#PPAD**(*PureCircuit*)$/2$ etc.
2) Find relativising parsimonious reductions or oracle separations to other counting problems, such as the ones shown in [1].
3) Attempt or make arguments towards the development of a Cook-Levin-type theorem for **#PPAD**(*PureCircuit*) with other counting problems.
4) If the previous objectives prove to be too hard for the scope and the timeframe of the current project, look deeper into the connections of hazard-free circuits and *PureCircuit* circuits.

In addition to the above we aim to develop a visualisation software to aid the development of theorems by calculating solutions for small sample sizes. Specifically we expect the following requirements:

1) Must be capable of representing a *PureCircuit* instance.
2) Could be capable of visualizing the instance.
3) Should identify whether instance is polynomially solvable, as the notion of weaker notions of the problem might be necessary for theory proving.
4) Must be able to count the number of solutions for small number of nodes. If possible, could experiment with different sizes, depending on the complexity of the instance.

III. Preliminaries and Background review

We assume the reader has an understanding basic complexity theory and of the following definitions **P**, **NP**, **FP**, **FNP** Turing Machines, Non-Deterministic Turing Machines. We refer to [25] for definitions and standard notations. We will first introduce the necessary preliminaries for the project. After that, we will then refer as to how the current literature analysed counting relations across various domains.

A. Preliminaries

To understand the current literature, we will break the current section into three parts: a brief overview of the **TFNP** complexity class, what do we imply by counting complexity and what is the *PureCircuit* problem.

*1) **TFNP** class and important subclasses:* **TFNP** is a subclass of search problems that are guaranteed to have a solution due to some property of the problem [16]. Based on that principle, several classes were created such as **PPAD** (Polynomial Parity Of Augmented Topologies) which takes advantage of the fact that if a digraph contains an unbalanced node, it should contain at least another one. Additional total search classes are: **PPP**(Polynomial Pigeonhole Principle) which is based on the Pigeonhole Principle, or even **CLS** (Continuous Local Search) which uses gradient descent to find solutions in convex domains [16], [26]. These problems are guaranteed to have a solution, but have no known polynomial time algorithm for finding them. Crucial to note the usage of Levin reductions, which reduce one search problem to another with the usage of two functions f, g such that f is used to map the input instance of one problem to another, and g takes the current instance and the output of the reduced problem, to construct a solution for the original problem. Moreover, many of these classes can form a hierarchy between themselves as observed in the figure 1 [1], [16].

2) Counting Complexity: In order to understand the ideas behind counting and the objectives of the current project we will offer a formal definition of the **#P** class and parsimonious reductions. As stated previously, **#P** is defined as the class of search problems in which we count the number of solutions for problems that are polynomially verifiable. Due to their nature, it is common to use Levin reductions to correlate counting problems. A particular subset of these problems of high interest, are known as *Parsimonious problems* which preserve the number of solutions across the two problem domains.

3) Relativisation: Relativisation is a concept that is built on the idea of oracle turing machines, where we augmented the set of operations of the machine with an arbitrary language $A \subseteq \{0,1\}^*$. This concept was created by [13] to introduce the *Relativisation Barrier*. Counting complexity theorists, took interest in this concept as it can be used to define **#P**-hardness but also relativising reductions or oracle separations across languages [1], [25], [27].

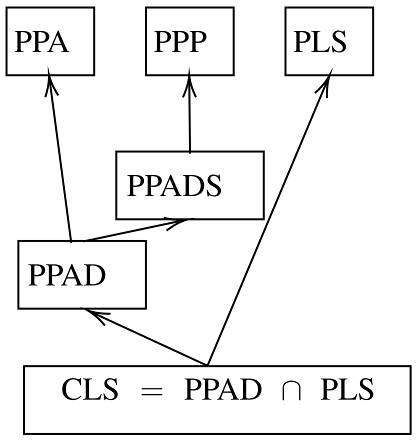


Fig. 1. Complexity hierarchy of some **TFNP** problems. Figure found from Ikenmeyer [1]

4) PureCircuit Problem:

Definition 3.1 (PureCiruit Problem Definition [23]): An instance of *PureCircuit* is given by vertex set $V = [n]$ and gate set G such that $\forall g \in G : g = (T, u, v, w)$ where $u, v, w \in V$ and $T \in \{\text{NOR}, \text{Purify}\}$. Each gate is interpreted as:

1) *NOR*: Takes as input u, v and outputs w
2) *Purify*: Takes as input u and outputs v, w

And each vertex is ensured to have in-deg$(v) \leq 1$. A solution to input instance (V, G) is denoted as an assignment $\mathbf{x} : V \to [0, 1]$, where we denote $0 \to 0, 1 \to 1, (0, 1) \to \bot$ such that for all gates $g = (T, u, v, w)$ we have

1) *NOR*:

$$\mathbf{x}[u] = \mathbf{x}[v] = 0 \implies \mathbf{x}[w] = 1$$
$$(\mathbf{x}[u] = 1 \vee \mathbf{x}[v] = 1) \implies \mathbf{x}[w] = 0$$
$$\text{otherwise} \implies \{0, 1, \bot\}$$

2) *Purify*:

$$\forall b \in \{0, 1\} : \mathbf{x}[u] = b \implies \mathbf{x}[v] = b \wedge \mathbf{x}[w] = b$$
$$\mathbf{x}[u] = \bot \implies \{\mathbf{x}[v] \cup \mathbf{x}[w]\} \cap \{0, 1\} \neq \emptyset$$

This problem was created by Deligkas, as a basis of another problem known as ϵ-*GCircuit* to demonstrate tight inapproximability bounds for **PPAD** problems and was proven to be **PPAD**-complete [23], [24]. In addition to the definition above they were able to extend the set of possibles gates to $\{$*NOT*, *COPY*, *AND*, *OR*, *NAND*$\}$ to ensure the robustness that was defined from the NOR gate, and showed that for set of gates types $\{$*Purify*$, X, Y\}$ where $(X, Y) \in \{$*NOT*$\} \times \{$*OR*, *AND*, *NOR*, *NAND*$\}$ or, $(X, Y) \in \{$*COPY*$\} \times \{$*NOR*, *NAND*$\}$, the problem remains **PPAD**-complete. Moreover, they showed that for other relaxations of the problem that do not make use of negations, the *Purify* gate, or drop the robustness requirement, will make the problem polynomially solvable [23], [24].

B. Literature Review

Over the years, people have attempted to find various of closure properties for the **#P** class, such as given $f_1, f_2 \in$ **#P**, is $f_1 - f_2 \in^?$ **#P**, which turns out to be false [28]. A more precise example, can be observed in the [1] paper, where they showed **PPAD**-complete problem *SourceOrExcess(2,1)*, under the decrementation separation -1 has an oracle separation with the **#P**, whereas *SourceOrSink* under halving separation $1/2$ and decrementation separation is **#P**-complete.

Expanding upon this, various classes that are equipped with various closure properties have been created such as **GapP** which measures the difference between the number of accepting and rejecting solutions, and it includes the subtraction closure as shown by [27], [28]. To demonstrate an example of such closures, we can use the function $\phi(f) = f - C_2^f + C_3^f$ where $\phi(0) = 0, \phi(1) = 1$ and $\phi(2) = 1$ from the paper [1]. We will use the *CircuitSAT*, which represents the class of boolean circuits that are satisfiable. We can show that $\phi($*#CircuitSAT*$) \notin$ **#P** as follows: Suppose there exists non-deterministic polynomial time Turing Machine N such that $\forall x \in \{0, 1\}^* : \phi($*CircuitSAT(x)*$) = \#acc_N(x)$. If the circuit is not satisfiable, then there should be no accepting paths, otherwise that would lead to a contradiction. Assume scenario where there is an exactly only one accepting path τ. There must exist at least one call to the oracle with input a such that $C(a) = 1$. We can create circuit C' such that $C'(a) = 0$ and $C'(b) = 1$ where b is a call on path τ' which does not contain a. If we create a circuit $\bar{C}$ where it

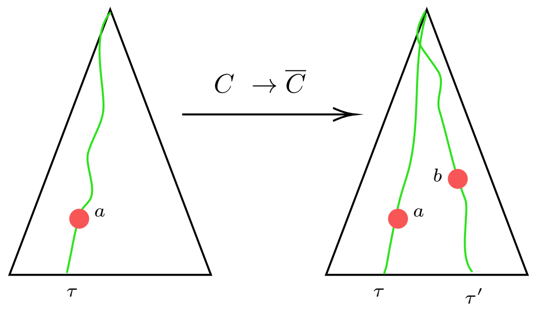


Fig. 2. Transformation of accepting paths between the original and augmented circuit. The triangles represent the executions paths of the machines. The highlighted lines show the accepting path with the red circles showing the accepting calls.

accepts both a and b, that would imply that the function has 2 accepting paths as observed in the figure 2. But this results $\phi(2) = 1 \neq \#acc_N(x) = 2$, which leads to a contradiction.

Based on the previous ideas, authors from the paper [1] have taken several problems from the **TFNP** hierarchy under counting separations and found several relativising parsimonious reductions and oracle separations as shown in 3. They developed a generalized diagonalization argument for closure properties and problems to find separations between **#P** class [1].

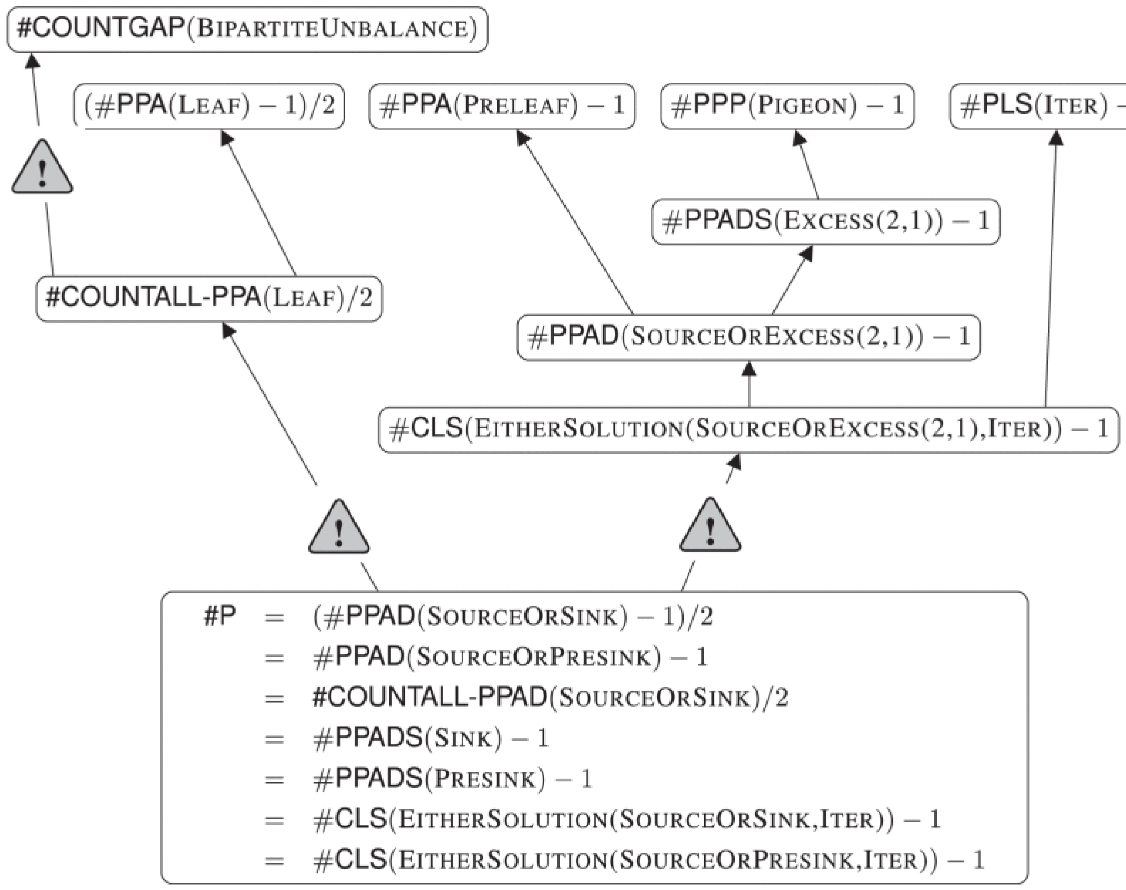

#COUNTGAP(BipartiteUnbalance)
(#PPA(Leaf) − 1)/2
#PPA(PreLeaf) − 1
#PPP(Pigeon) − 1
#PLS(Iter) − 1
#PPADS(Excess(2,1)) − 1
#COUNTALL-PPA(Leaf)/2
#PPAD(SourceOrExcess(2,1)) − 1
#CLS(EitherSolution(SourceOrExcess(2,1),Iter)) − 1
#P = (#PPAD(SourceOrSink) − 1)/2
= #PPAD(SourceOrPresink) − 1
= #COUNTALL-PPAD(SourceOrSink)/2
= #PPADS(Sink) − 1
= #PPADS(Presink) − 1
= #CLS(EitherSolution(SourceOrSink,Iter)) − 1
= #CLS(EitherSolution(SourceOrPresink,Iter)) − 1

Fig. 3. The relativizing reductions and oracle separations across several classes. From Ikenmeyer [1].

It is crucial to point out, that a variety of the problems that have been used, have very simplistic but restrictive representations. Examples of these include the *SourceOrExcess(2,1)* where *SourceOrSink*, which have different expressibility in the counting hierarchy but no obvious ways to reduce them between other problems in **#PPAD**. In the current paper, we conjecture that the circuit properties and representations of the *PureCircuit* under halving or decrementating separations will offer a new **#PPAD**-hardness perspective as it has not been studied under the counting version lens.

Lastly we are also presenting the motivation for incorporating hazard free circuits. These are special type of circuits to incorporate an undefined value in the algebra of the circuit and had an impact on circuit complexity to show bounds for monotone circuits [8] as well as communication complexity [29]. Those circuits could be of particular interest as, pure circuits seem to be generalizations of hazard free circuits, by treating $\perp$ values similarly as hazard free circuits treat the undefined values [24]. This is a potential avenue to pursue given the other efforts are not feasible within the time constraints of the dissertation.

IV. Research Methodology

In the current section we will outline the methodology we will employ to tackle the objectives we previously outlined. We will divide the section into three parts: problem approach, software development and proof verification.

A. Problem Approach

In order to, find any useful results, we essentially enquire a two-step approach: First find define counting variants of **#PPAD**(*PureCircuit*) and second find reductions between the current problem and the other problems in the hierachy.

For the first part of the plan, it might be necessary to create several problems to find non-parsimonious reductions that offer a havling or decrementing separation to the counting *PureCircuit* problem. It would be helpful to additionally investigate how permutations of the possible gates affect the number of solutions or help in the reduction of the problem to other classes. We could experiment with weaker versions of the problem as [24] showed that certain enumeration of gates, make the problem polynomially solvable. As for the second part of the plan, it would be helpful to gather **PPAD**-complete problems throughout literature. A useful starting point would be the reductions that were defined in the [24] paper, such as the *StrongSpenser*, ϵ-*GCircuit* and various types of games. Moreover, reference from other total search classes such as **PPP** or **PLS** would also be useful as that would allow us to apply extensions from problems that can be observed in figure 3.

It is also essential to find relevant proof techniques. Most important ones will be relativising parsimonious reductions through the usage of theoretical gadgets as applied in several counting reductions proofs [1], [18]. For oracle separations, we can observe how the usage of the binomial representations of the problems can be used under the diagonalization theorem to argue about separations.

B. Software Development

In the current section, we will refer to the software methodology strategy we will employ to make the visualisation tool. Since the tool will be mainly used to aid the development of theorems, we expect a limited scope. First and foremost, we expect to use the waterfall methodology with the possibility of small iterations due to the limited scope and number of requirements. Adding on, to ensure correctness, Test Driven Development will be used where we iteratively develop the tool through test cases. Due to the lack of any fast algorithms for the full enumeration of the problem, we will use system software languages like *Rust* to make the most out of the system resources. For the visualisation aspect, we can experiment with either Rust-only or web solutions due to their flexibility and ease of use.

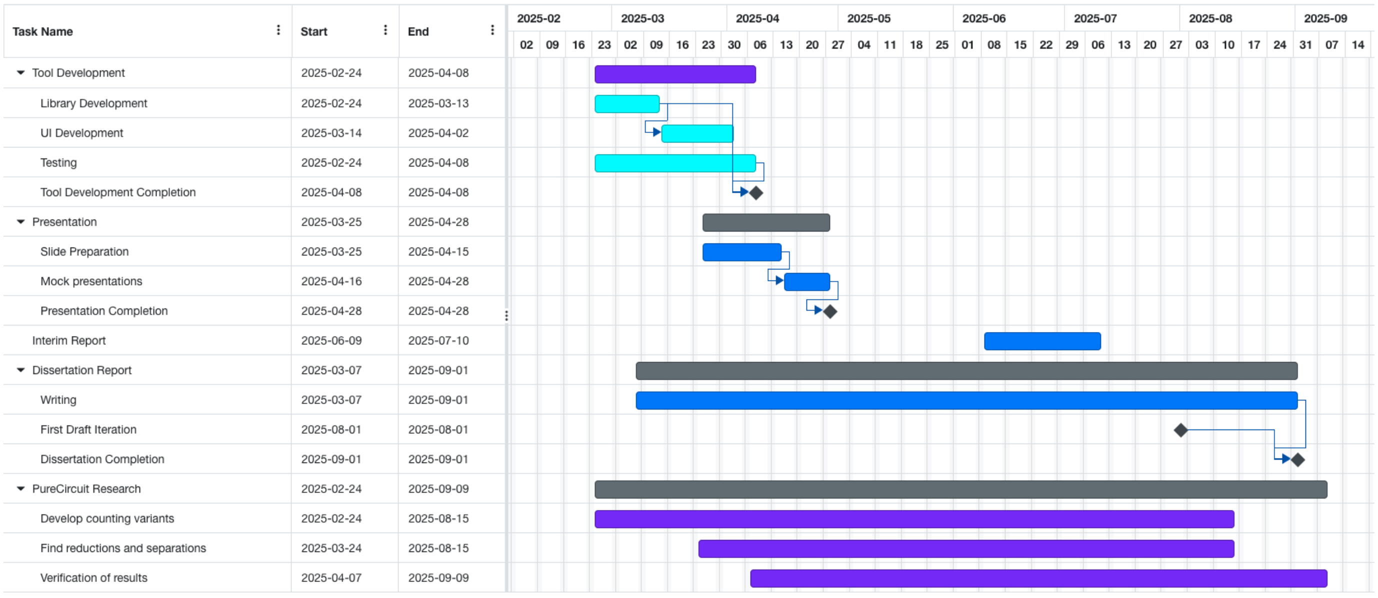
C. Proof Verification

Due to the theory aspect of the project, it is fundamental to ensure the correctness of the proofs. To do that we aim to use various techniques such as: formal verifiers, visualisation tools, peer reviewing and heuristical arguments. First, formal verification is a common technique to ensure correctness of a proof with the aid of proof assistants as shown here [30]. Second, our visualisation tool can be used to find the number of solutions given the problem size is small enough as well as the usage of TDD techniques to ensure correctness. For that we will need to refer to peer reviewing of our proofs. Communication with other individuals in the field that are outside of the project, will offer on objective opinion with respect to the correctness of our arguments. Lastly, the *PureCircuit* problem can be solved in polynomial time for a specific set of circuits. Developing algorithms for the weaker variants problems would be an additional option to show correctness.

V. Project Plan

A. Timeline and Milestones

In the figure 4, we will present an initial timeline of the project. It contains important deadlines such as the presentation and the final submission. The aims and objectives that we declared in the previous sections are highlighted by purple lines. It should be noted that, for the research part of the project, since the main goal is to uncover as many reductions and relations as possible, this will be a continuous process throughout the duration of the project. Additionally, for the tool development, there is a possibility that more requirements might appear, but for now we aim to have a simplified version of the tool available for the presentation deadline. Lastly, there is a possibility the current direction will not reveal anything of significance. If that is the case, then additional milestones will be added to analyse the secondary objective of hazard-free circuits that were previously mentioned.



Task Name	Start	End
Tool Development	2025-02-24	2025-04-08
Library Development	2025-02-24	2025-03-13
UI Development	2025-03-14	2025-04-02
Testing	2025-02-24	2025-04-08
Tool Development Completion	2025-04-08	2025-04-08
Presentation	2025-03-25	2025-04-28
Slide Preparation	2025-03-25	2025-04-15
Mock presentations	2025-04-16	2025-04-28
Presentation Completion	2025-04-28	2025-04-28
Interim Report	2025-06-09	2025-07-10
Dissertation Report	2025-03-07	2025-09-01
Writing	2025-03-07	2025-09-01
First Draft Iteration	2025-08-01	2025-08-01
Dissertation Completion	2025-09-01	2025-09-01
PureCircuit Research	2025-02-24	2025-09-09
Develop counting variants	2025-02-24	2025-08-15
Find reductions and separations	2025-03-24	2025-08-15
Verification of results	2025-04-07	2025-09-09

Fig. 4. Gantt Chart

B. *Constraints and Risk management*

We will refer to the following table I to outline a list of possible risks, their severity, and how to possibly mitigate them. These mainly tackle research related risks. There are always external factors that can halt the progress of the project, but careful time management and resource allocation should suffice to mitigate such circumstances.

TABLE I
SEVERITY AND RISK TABLE

Severity	Probability	Description	Mitigation
High	High	No significant reductions or separations found or Cook-Levin formalisations.	We can switch to hazard-free circuits where it is guaranteed to find results as previously mentioned. Additionally, we can use weaker versions of the original problem where there are more constraints.
High	Medium	Project may not be infeasible within the time constraints	Weekly meetings with supervisors to assess the progress as well as usage of such as timetables and Gantt charts to keep track of the process.
High	Medium	Incorrect or not well-defined proofs.	Due to the nature of the project, mathematical rigour in key. Peer reviewing is integral to verify correctness of the project as well as proof assistants.
Low	Low	Tool produces incorrect results or does not work for sufficient cases.	We aim to make use of TDD methodologies to ensure correctness. Additional usage of *Rust* should provide the necessary speed and safety needed to cover as many cases as possible.

VI. Conclusion

In conclusion, we have outlined an investigation into the counting complexity of **#PPAD**(*PureCircuit*) under counting separations, building upon existing work to extend the hierarchy of total search problems. By introducing counting variants of *PureCircuit*, we aim to deepen the theoretical understanding of counting complexity of the **PPAD** class by conjecturing the existence of a Cook Levin type theorem that characterizes the upper bounds of **#PPAD**-hardness. The feasibility of this work is supported by existing theoretical foundations, and our next steps involve formalizing our conjectures and developing proof techniques to establish key results with the usage of a visualisation software and the pre-existing proof techniques.

References

[1] C. Ikenmeyer and I. Pak, “What is in #P and what is not?” Apr. 2022, arXiv:2204.13149 [cs]. [Online]. Available: http://arxiv.org/abs/2204.13149

[2] S. A. Cook, “The complexity of theorem-proving procedures,” in *Proceedings of the third annual ACM symposium on Theory of computing*, ser. STOC ’71. New York, NY, USA: Association for Computing Machinery, May 1971, pp. 151–158. [Online]. Available: https://dl.acm.org/doi/10.1145/800157.805047

[3] L. Fortnow, “The status of the P versus NP problem,” *Commun. ACM*, vol. 52, no. 9, pp. 78–86, Sep. 2009. [Online]. Available: https://dl.acm.org/doi/10.1145/1562164.1562186

[4] B. Berger and T. Leighton, “Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete,” *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology*, vol. 5, no. 1, pp. 27–40, 1998.

[5] R. Merkle and M. Hellman, “Hiding information and signatures in trapdoor knapsacks,” *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 525–530, Sep. 1978, conference Name: IEEE Transactions on Information Theory. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1055927

[6] C. Daskalakis, P. Goldberg, and C. Papadimitriou, *The complexity of computing a Nash equilibrium*. Association for Computing Machinery, Jan. 2006, vol. 39, journal Abbreviation: SIAM Journal on Computing Pages: 78 Publication Title: SIAM Journal on Computing.

[7] I. Carboni Oliveira and R. Santhanam, “Hardness Magnification for Natural Problems,” in *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, Oct. 2018, pp. 65–76, iSSN: 2575-8454. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8555094

[8] C. Ikenmeyer, B. Komarath, C. Lenzen, V. Lysikov, A. Mokhov, and K. Sreenivasaiah, “On the complexity of hazard-free circuits,” *Journal of the ACM*, vol. 66, no. 4, pp. 1–20, Aug. 2019, arXiv:1711.01904 [cs]. [Online]. Available: http://arxiv.org/abs/1711.01904

[9] J. Pich, “Localizability of the approximation method,” Dec. 2022, arXiv:2212.09285 [cs]. [Online]. Available: http://arxiv.org/abs/2212.09285

[10] P. Pudlak, “The canonical pairs of bounded depth Frege systems,” Dec. 2019, arXiv:1912.03013 [math]. [Online]. Available: http://arxiv.org/abs/1912.03013

[11] A. A. Razborov and S. Rudich, “Natural Proofs,” *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 24–35, Aug. 1997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002200009791494X

[12] S. Aaronson and A. Wigderson, “Algebrization: a new barrier in complexity theory,” in *Proceedings of the fortieth annual ACM symposium on Theory of computing*, ser. STOC ’08. New York, NY, USA: Association for Computing Machinery, May 2008, pp. 731–740. [Online]. Available: https://doi.org/10.1145/1374376.1374481

[13] T. Baker, J. Gill, and R. Solovay, “Relativizations of the $\mathcal{P} = ?\mathcal{NP}$ Question,” *SIAM Journal on Computing*, vol. 4, no. 4, pp. 431–442, Dec. 1975, publisher: Society for Industrial and Applied Mathematics. [Online]. Available: https://epubs.siam.org/doi/10.1137/0204037

[14] S. Aaronson, “Why Philosophers Should Care About Computational Complexity,” Aug. 2011, arXiv:1108.1791 [cs]. [Online]. Available: http://arxiv.org/abs/1108.1791

[15] P. W. Goldberg, “A Survey of PPAD-Completeness for Computing Nash Equilibria,” Mar. 2011, arXiv:1103.2709 [cs]. [Online]. Available: http://arxiv.org/abs/1103.2709

[16] C. H. Papadimitriou, “On the complexity of the parity argument and other inefficient proofs of existence,” *Journal of Computer and System Sciences*, vol. 48, no. 3, pp. 498–532, Jun. 1994. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022000005800637

[17] X. Chen, X. Deng, and S.-H. Teng, “Settling the complexity of computing two-player Nash equilibria,” *J. ACM*, vol. 56, no. 3, pp. 14:1–14:57, May 2009. [Online]. Available: https://doi.org/10.1145/1516512.1516516

[18] L. G. Valiant, “The complexity of computing the permanent,” *Theoretical Computer Science*, vol. 8, no. 2, pp. 189–201, Jan. 1979. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0304397579900446

[19] S. Toda, “PP is as Hard as the Polynomial-Time Hierarchy,” *SIAM Journal on Computing*, vol. 20, no. 5, pp. 865–877, Oct. 1991, publisher: Society for Industrial and Applied Mathematics. [Online]. Available: https://epubs.siam.org/doi/10.1137/0220053

[20] L. Valiant, “Holographic algorithms,” in *45th Annual IEEE Symposium on Foundations of Computer Science*, Oct. 2004, pp. 306–315, iSSN: 0272-5428. [Online]. Available: https://ieeexplore.ieee.org/document/1366250

[21] A. Bandyopadhyay and D. Gamarnik, “Counting without sampling. New algorithms for enumeration problems using statistical physics,” Oct. 2005, arXiv:math/0510471. [Online]. Available: http://arxiv.org/abs/math/0510471

[22] K. Mulmuley and M. Sohoni, “Geometric Complexity Theory, P vs. NP and Explicit Obstructions,” in *Advances in Algebra and Geometry: University of Hyderabad Conference 2001*, Jan. 2003, pp. 239–261.

[23] A. Deligkas, J. Fearnley, A. Hollender, and T. Melissourgos, “Pure-Circuit: Strong Inapproximability for PPAD,” in *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, Oct. 2022, pp. 159–170, iSSN: 2575-8454. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9996749

[24] ——, “Pure-Circuit: Tight Inapproximability for PPAD,” *J. ACM*, vol. 71, no. 5, pp. 31:1–31:48, Oct. 2024. [Online]. Available: https://dl.acm.org/doi/10.1145/3678166

[25] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*. Cambridge: Cambridge University Press, 2009. [Online]. Available: https://www.cambridge.org/core/books/computational-complexity/3453CAFDEB0B4820B186FE69A64E1086

[26] J. Fearnley, P. W. Goldberg, A. Hollender, and R. Savani, “The complexity of gradient descent: CLS = PPAD and PLS,” in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC 2021. New York, NY, USA: Association for Computing Machinery, Jun. 2021, pp. 46–59. [Online]. Available: https://doi.org/10.1145/3406325.3451052

[27] J. Torán, “A combinatorial technique for separating counting complexity classes,” in *Automata, Languages and Programming*, G. Ausiello, M. Dezani-Ciancaglini, and S. R. Della Rocca, Eds. Berlin, Heidelberg: Springer, 1989, pp. 733–744.

[28] S. A. Fenner, L. J. Fortnow, and S. A. Kurtz, “Gap-definable counting classes,” *Journal of Computer and System Sciences*, vol. 48, no. 1, pp. 116–148, Feb. 1994. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022000005800248

[29] C. Ikenmeyer, B. Komarath, and N. Saurabh, “Karchmer-Wigderson Games for Hazard-free Computation,” Nov. 2022, arXiv:2107.05128 [cs]. [Online]. Available: http://arxiv.org/abs/2107.05128

[30] G. Gonthier, “The Four Colour Theorem: Engineering of a Formal Proof,” in *Computer Mathematics*, D. Kapur, Ed. Berlin, Heidelberg: Springer, 2008, pp. 333–333.