CS907 Interim Report

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

Abstract

This project investigates the combinatorial structure of the PURECIRCUIT problem. We conjecture that its circuit-based formulation can provide insights and bounds on the counting complexity of PPAD-1. In this report, we present the progress made so far, including preliminary results and ongoing developments. In addition, we report on the construction of a visualisation tool designed to support the exploration and formulation of these combinatorial theorems. Lastly we outline the current state of the project and explore the subsequent next steps.

I. INTRODUCTION

In recent years, there has been a revolutionary initiative in the field of combinatronics. Combinatronics has been a field of study in mathematics that primarily focused on the notion of counting objects with certain properties. Over time, this notion has shifted, especially in the subfield of algebraic combinatorics, where there is no clear notion of the object that we are counting, and the numbers express something more abstract [1]. This created the need to assign combinatorial interpretations to such numbers. Being able to find such definitions or interpretations is be very important as it allows us to utilise tools from combinatorics to understand and reveal hidden structures and properties [1]. Moreover, there are several problems or numbers such as *Kronecker coefficients* [2], whose combinatorial interpretation, would lead to groundbreaking breakthroughs such as a step closer to the resolution of the $P \neq NP$ conjecture [3], [4].

In our current work, we focus on extending the work done by Ikenmeyer et al. [3], where they focused on the creation of frameworks that determines whether $f \in^{?}$ **#P**, by looking at the subclasses of #TFNP - 1 problems. In our work we focus specifically on the complexity class of PPAD, under the lens of PureCircuit, due to circuit-based nature. We hope to uncover many insights of the #PURECIRCUIT $- 1$ problem as well as explore the boundaries of #PPAD $- 1$.

A. Project objectives

Our project requires the extending the theory of #PURECIRCUIT as well as developing a software to visualise its instances.

R.1) Find a parsimonious reduction from the *EndOfLine* to *PureCircuit*.

R.2) Demonstrate that #PURECIRCUIT $- 1 \not\subseteq$ **#P** either by:

  a) Showing that #SOURCEOREXCESS(K,1) $\subseteq$ #PURECIRCUIT for some $k \in \mathbb{N}_{\geq 2}$
  b) Showing that #SOURCEOREXCESS(K,1) $\subseteq$ #ND-STRONGSPERNER for some $k, n \in \mathbb{N}_{\geq 2}$

R.3) Prove or disprove the following:

$$\forall n \in \mathbb{N}_{\geq 2}, \exists c \in \mathbb{N} : \#\text{SOURCEOREXCESS}(n, 1) \subseteq^{c} \#\text{PURECIRCUIT}$$

Below we will be representing the development milestones of the project.

S.1) Visualise and verify a PURECIRCUIT instance.

S.2) Generate a solution given a PURECIRCUIT instance for a small number of nodes.

S.3) Count the number of solutions of a PURECIRCUIT instance for a small number of nodes.

We will compile the rest of the report, based on our current findings, how we altered our previous objectives and general reflections.

B. Research Question

Our objectives revolve around the conjecture in I.1, where we investigate the boundaries of #PPAD $- 1$ with the help of the PURECIRCUIT problem.

Conjecture I.1: #PPAD $- 1$ **hardness**

Every language in PPAD can be parsimoniously reduced up to some polynomial factor, to the PURECIRCUIT problem, or more formally:

$$\forall L \in \text{PPAD}, \exists f \in n^{O(1)} : \#\text{L} \subseteq^{f} \#\text{PURECIRCUIT}$$

II. Preliminaries and Background review

A. Important Notation

To avoid ambiguity, we introduce the following notation conventions used throughout the paper. For any $n \in \mathbb{N}$, we write $[n] \triangleq \{1, \ldots, n\}$ and $[n]_0 \triangleq [n] \cup \{0\}$. We define the Boolean domain as $\mathbb{B} \triangleq \{0, 1\}$ and three-value domain $\mathbb{T} \triangleq \{0, 1, \perp\}$.

B. Background overview

1) Counting Complexity and Combinatorial Interpretations: Counting complexity looks into complexity of the number counting solutions of a given problem. In the current project we will focus specifically on the #P class. #P was created by Valiant [5], to demonstrate the difficulty of counting the number of solutions to a problem, where each solution can be polynomially verified. More formally:

Definition II.1: #P Complexity Clas [5]

A function $f : \mathbb{B}^* \to \mathbb{N} \in$ #P, if there exists a poly-function $p : \mathbb{N} \to \mathbb{N}$ and a poly-time TM M such that:

$$f(w) = \left|\left\{v \in \mathbb{B}^{p(|w|)} \mid M(w, v) = 1\right\}\right|$$

Given the above definition, Pak et al. [1], [6] gave the following reasons for choosing #P:

1) By polynomially bounding words, we avoid cases such as: $f(w) = |\{1, \ldots, f(w)\}|$.
2) Work with $f(\cdot)$, even if its direct computation is hard.
3) Formal system that can be highly flexible.

The current framework was used in several papers such as [3] and [6] to demonstrate the existance of combinatorial interpretations. Lastly we introduce the idea of correlating two counting problems with the help of *parsimonious reductions* II.2.

Definition II.2: Parsimonious reductions

Let R, R' be search problems and let f be a reduction of $S_R = \{x \mid R(x) \neq \emptyset\}$ to $S_{R'} = \{x \mid R'(x) \neq \emptyset\}$. We say f is **parsimonious** if:

$$\forall x \in S_R : |R(x)| = |R'(f(x))|$$

Below we are introducing a variant of parimonious reductions that allows for one-to-many reductions.

Definition II.3: Poly-Function Bounded Parsimonious Reductions

Given two counting problems $A, B : \mathbb{B}^* \to \mathbb{N}$ and a poly-function $f : \mathbb{N} \to \mathbb{N}$, we say that:

$$A \subseteq^f B \iff \forall x \in \mathbb{B}^n : A(x) \leq B(x) \leq f(|x|) \cdot A(x)$$

If $\forall x : f(x) = c$ for $c \in \mathbb{N}$, we use the abbreviation:

$$A \subseteq^c B$$

2) Total Search Problems and PPAD: We give a brief overview of total search problems and their relevancy with the project.

Definition II.4: Search Problems and Total Search Problems

Search problems can be defined as relations $R \subseteq \mathbb{B}^* \times \mathbb{B}^*$, where given $x \in \mathbb{B}^*$, we want to find $y \in \mathbb{B}^*$ such that xRy. **Total Search problems** are search problems such that for each input, there must exist at least one solution.

Definition II.5: FNP and TFNP

FNP are *search problems* such that there exists poly-time TM $M : \mathbb{B}^* \to \mathbb{B}$ and a poly function $p : \mathbb{N} \to \mathbb{N}$ such that:

$$\forall x \in \mathbb{B}^*, y \in \mathbb{B}^{p(|x|)} : xRy \iff M(x, y) = 1$$

Lastly **TFNP** $= \{L \in$ **FNP** $\mid L$ is total$\}$

Definition II.6: *EndOfLine* problem [7]

Given poly-sized circuits $S, P \in \mathbb{B}^n \to \mathbb{B}^n$, we define a digraph $G = (V, E)$, such that $V = \mathbb{B}^n$ and E defined as:

$$E = \{(x, y) \in V^2 : S(x) = y \land P(y) = x\}$$

We define source or sinks $\forall v \in V : deg(v) = (0, 1)$ or $deg(v) = (1, 0)$, respectively. We also syntactially ensure that the 0^n node is always a source, meaning $S(P(0^n)) \neq 0 \land P(S(0^n)) = 0^n$. A node $v \in V$ is a solution if and only if $deg(v)$ is either $(0, 1)$ or $(1, 0)$.

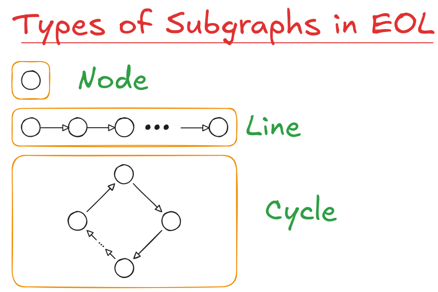


Fig. 1: Types of subgraphs in ENDOFLINE

An illustrative example of an ENDOFLINE instance can be seen in the figure 1. Using that, we define the PPAD complexity class II.7.

Definition II.7: PPAD complexity class [7]

PPAD is defined as the set of search problems that are reducible to the ENDOFLINE problem II.6.

PPAD has been created by Papadimitriou [7] to demonstrate a subset of problems in NP that guarantee a solution but believed to be very inefficient to find. In the next section, we introduce relevant PPAD problems.

a) The PureCircuit problem: The definition of PURECIRCUIT is based Kleene's three-valued strong logic of indeterminancy, which extends the traditional $\mathbb{B}$ logic [8]. This problem was created by Deligkas et al. [9] to demonstrate the hardness of approximating PPAD problems and was shown to be PPAD-COMPLETE.

Definition II.8: PURECIRUIT Problem Definition [9]

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 $\mathbf{x} : V \to \{0, \perp, 1\}$ such that for all gates $g = (T, u, v, w)$ we have:

1) *NOR*:

$$\mathbf{x}[u] = \mathbf{x}[v] = 0 \implies \mathbf{x}[w] = 1$$
$$(\mathbf{x}[u] = 1 \lor \mathbf{x}[v] = 1) \implies \mathbf{x}[w] = 0$$
$$\text{otherwise} \implies \perp$$

2) *Purify*:

$$\forall b \in \mathbb{B} : \mathbf{x}[u] = b \implies \mathbf{x}[v] = b \land \mathbf{x}[w] = b$$
$$\mathbf{x}[u] = \perp \implies \{\mathbf{x}[v] \cup \mathbf{x}[w]\} \cap \mathbb{B} \neq \emptyset$$

In addition to the gates above, we will make use of the standard set of gates $\{\lor, \land, \neg\}$, whose behaviour is depicted in the tables I. Moreover, we will make use of the *Copy* gate which we can define as *Copy*$(x) = \neg(\neg x)$. These gates are well defined based on the *NOR* gate as showed by Deligkas et al. [9] and do not directly affect the complexity of the problem. Lastly with respect to the *Purify* gate, Deligkas et al. [9] showed that the only *Purify* solutions essential for PPAD-COMPLETENESS are $\{(0, \perp), (\perp, 1), (0, 1)\}$, and the addition of more solutions does not affect its complexity. We will base our constructions around that limited set of solutions. We acknowledge that this simplified variant of the problem captures subset of the solutions but, one can observe that #PURECIRCUIT-SIMPLIFIED $\subseteq$ #PURECIRCUIT, and therefore any proposition or argument of the sort: #A $\subseteq$ #PURECIRCUIT-SIMPLIFIED $\implies$ #A $\subseteq$ #PURECIRCUIT. All solution sets will be made explicit before analysing the counting complexity of the problem, and we will refer to all such simplified variants as #PURECIRCUIT to avoid confusion.

`not`	
0	1
$\perp$	$\perp$
1	0

(a) `not` gate

`and`	0	$\perp$	1
0	0	0	0
$\perp$	0	$\perp$	$\perp$
1	0	$\perp$	1

(b) `and` gate

`or`	0	$\perp$	1
0	0	$\perp$	1
$\perp$	$\perp$	$\perp$	1
1	1	1	1

(c) `or` gate

TABLE I: Three-valued logic [8]

b) Sperner problems: Below we will refer to the notion of Sperner problems which involve the idea of using the topology of a problem and a colouring scheme to ensure that a substructure is panchromatic. There two variants of colouring scheme that are used: one of them will be referred to as the **linear** colouring where for dimension d, assign $d+1$ distinct colours to each point [10], [11]. Below we will refer to **bipolar** colouring, which has been used grid-like topologies of the Sperner property [9], [12], [13]. We note that these terms are not standard in literature; we adopt this naming convention for the sake of clarity.

Definition II.9: Bipolar colouring

Given a set S^d, where S is some arbitrary set, we refer to bipolar colouring C as:

$$\forall v \in S, j \in [d] : [C(v)]_j \in \{-1, 1\}$$

We say that a set of points $A \subseteq S^d$ **cover all the labels** if:

$$\forall i \in [d], \ell \in \{-1, +1\}, \exists x \in A : [\lambda(x)]_i = \ell$$

Deligkas et al. [9] showed that it suffices for $|A|$ to be at most d for covering condition to hold.

Definition II.10: StrongSperner problem

Input: A boolean circuit that computes a bipolar labelling $\lambda : [M]^N \to \{-1, 1\}^N$ II.9 satisfying the following boundary conditions for every $i \in [N]$:

- if $x_i = 1 \implies [\lambda(x)]_i = +1$
- if $x_i = M \implies [\lambda(x)]_i = -1$

Output: A set of points $\{x^{(i)}\}_{i \in [N]} \subseteq [M]^N$, such that:

- *Closeness condition*: $\forall i, j \in [N] : \|x^{(i)} - x^{(j)}\|_\infty \leq 1$
- *Covers all labels* as defined in II.9

The above is a generalised variant of the traditional Sperner problem to a grid of dimensions N with width of M. Throughout literature the same variants of the problem or specifications have been defined using the names Sperner or discrete Brouwer [9]–[12]. For clarity, the subsequent analysis adopts StrongSperner.

Definition II.11: nD-StrongSperner problem

Input: A tuple $(\lambda, 0^k)$ of a StrongSperner instance but for only n dimensions, such that $\lambda : (\mathbb{B}^k)^n \to \{-1, +1\}^n$.
Output: A point $\alpha = (a_1, \ldots, a_n) \in (\mathbb{B}^k \setminus \{1^k\})^n$ such that

$$\{\alpha + x \mid x \in \mathbb{B}^n\} \text{ cover all the labels II.9}$$

We assume dimensionality $n \geq 2$.

The authors of these papers refer to both colouring schemes interchangeably when discussing their reductions, so we can assume that all reductions (not necessarily counting reductions) work the similarly for either colouring [9]–[12].

c) Other PPAD problems: We will define several problems in PPAD that are related with our project and demonstrate the counting complexity of PureCircuit.

Definition II.12: SourceOrExcess problem [3]

We define as *SourceOrExcess(k,1)* for $k \in \mathbb{N}_{\geq 2}$ the search problem as such: Given a poly-sized successor circuit $S : \mathbb{B}^n \to \mathbb{B}^n$ and a set of predecessor poly-sized circuits $\{P_i\}_{i \in [k]}$, where $P_i : \mathbb{B}^n \to \mathbb{B}^n$, we define the graph $G = (V, E)$ such that, $V = \mathbb{B}^n$ and E as:

$$\forall x, y \in V : (x, y) \in E \iff (S(x) = y) \wedge \bigvee_{i \in [k]} P_i(y) = x$$

We ensure that 0^n is as sink, meaning $\deg(0^n) = (0, 1)$. A valid solution is a vertex v such that *in-deg*$(v) \neq$ *out-deg*(v)

The SourceOrExcess(k,1) problem can be thought of as the generalisation of the EndOfLine problem for graphs with *in-deg*$(\cdot) \leq k$ and *out-deg*$(\cdot) \leq 1$. Lastly we will introduce Tarski II.14 which uses the Knaster-Tarski fixed point theorem II.1 and belongs in PLS ∩ PPAD, where PLS is a class of problems based on the idea of local search [14].

Definition II.13: Monotone functions

Given two partially ordered sets $(L_1, \preceq_{L_1})$ and $(L_2, \preceq_{L_2})$, a function $f : L_1 \to L_2$ is **monotone** if and only if:

$$\forall x, y \in L_1 : x \preceq_{L_1} y \implies f(x) \preceq_{L_2} f(y)$$

Theorem II.1: Knaster Tarski Fixed point theorem [15], [16]

Given a *monotone* function $f : L \to L$ II.13 of a lattice $(L, \wedge, \vee)$

$$\exists c \in L : f(c) = c$$

Definition II.14: TARSKI problem definition [16]

Given a *monotone* function $f : L \to L$ II.13 of a lattice $(L, \wedge, \vee)$ we define solutions to the problem as:

1) Find $x \in L : f(x) = x$
2) Find $x, y \in L$ such that $x \preceq y$ and $f(x) \npreceq f(y)$

We assume that f is described by circuit and which describes the size of the input.

d) Counting complexity of PPAD: Ikenmeyer et al. [3] demonstrated that several PPAD-COMPLETE problems behave differently under #PPAD $- 1$. More specifically #ENDOFLINE $- 1 \subseteq$ #P but $\exists A$: #SOURCEOREXCESS$^A - 1 \not\subseteq$ #P^A. We base the seperation between easy and hard problems using the two aforementioned problems as seen in the figure 2.

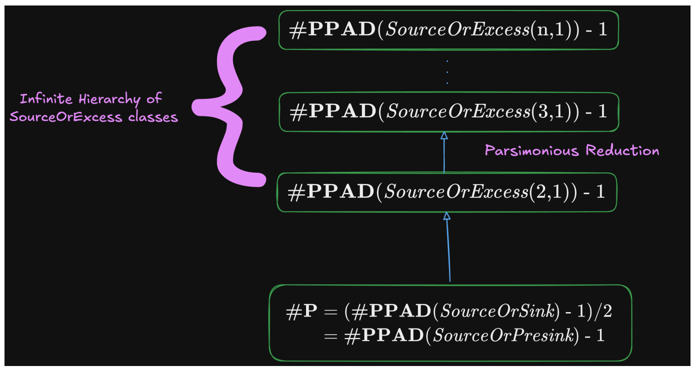


Fig. 2: Hierarchy graph between #PPAD problems. Figure from [3]

3) Kleene Logic and Hazard-Free Circuits: Kleene logic uses the boolean system with an additional *unstable* value [8]. Study of Kleene logic aided in the development of robust physical systems [17], as well as defining frameworks like *Alternative Turing machines* [18], and aiding in the development of lower bounds for monotone circuits [19]–[22]. The current section offers a brief summary to relevant notions and concepts in Kleene logic.

Definition II.15: Kleene value ordering [23]

The instability ordering for $\leq^u$ is defined as: $\perp \leq^u 0, 1$. The n-dimensional extension of the ordering is:

$$\forall x, y \in \mathbb{T}^n : x \leq^u y \implies \forall j \in [n] : (x_j \in \mathbb{B} \implies x_i = y_i)$$

Definition II.16: Kleene Resolutions [20], [23]

Given $x \in \mathbb{T}^n$, we define the *resolutions* of x, using the following notation:

$$\textsc{Res}(x) \triangleq \{y \in \mathbb{B}^n \mid x \leq^u y\}$$

Unless otherwise specified, we assume that all functions are *natural* II.17, since they can be represented by circuits **??**. Detecting hazards is a concept that is analysed heavily when talking about Kleene logic. The idea is ensuring robustness in our circuits, meaning if all resolutions of $x \in \mathbb{T}^n$ give the same value, then we should expect circuit to behave the same way II.18. An example of hazard values can be seen in the figure 3.

Definition II.17: Natural functions [20]

A function $F : \mathbb{T}^n \to \mathbb{T}^m$ for some $n, m \in \mathbb{N}$ is *natural* if and only if it satisfies the following properties:

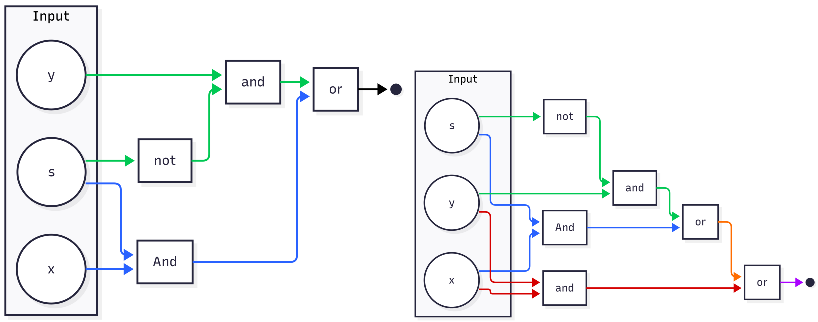
1) *Preserves stable values*: $\forall x \in \mathbb{B}^n : F(x) \in \mathbb{B}^m$
2) *Preserves monotonicity* II.13 II.16

Proposition II.1: Natural functions and Circuits [20], [23]

A function $F : \mathbb{T}^n \to \mathbb{T}^m$ can be computed by a circuit iff F is *natural* II.17

Definition II.18: Hazard [19], [20]

A *circuit* C, on n inputs has **hazard** at $x \in \mathbb{T}^n \iff C(x) = \perp$ and $\exists b \in \mathbb{B}, \forall r \in \text{RES}(x) : C(r) = b$. If such value does not exists then we say that C is hazard-free.



(a) Multiplexer with hazard at $(1, 1, \perp)$ (b) Hazard free multiplexer

Fig. 3: Hazard circuit and hazard-free circuit. Figure by [20]

Definition II.19: K-bit Hazard

For $k \in \mathbb{N}$ at $x \in \mathbb{T}^n \iff C$ has a *hazard* at x and $\perp$ appears at most k times in x

III. Current Progress

A. Theory Progress

This section focuses on the theory progress we made. We have experimented with various approaches such as investigating easier variants of PURECIRCUIT to find reductions to the ENDOFLINE. Moreover, we looked into modifying the PURIFY gate or finding a variant where we have more control over the $\perp$ value. We soon realised that under Marino et al. [24] continuity arguments, such variants are not guaranteed to exist. Our core breakthrough was realised when experimenting with Hazard-Free circuits and *UniqueEndOfPotentialLine*) problems. Specifically, we looked at a problem by Fearnley et al. [25] where they introduced a variant Brouwer problems known as OPDC, which is a local descent problem with fixed point topologies. Combining ideas of the two lead to the main find of our project in III-A1.

1) ND-STRONGSPERNER Constant Parsimonious Reduction: The reduction from STRONGSPERNER to PURECIRCUIT was first done by Deligkas et al. [9], under the condition of unbounded dimensionality and polynomial width and without any parsimonious bounds. We offer a polynomial parimonious reduction II.3 for every ND-STRONGSPERNER problem to PURECIRCUIT. We will subsequently demonstrate our reduction for 3 dimensions and then we will generalise.

Theorem III.1: 3D-StrongSperner to PureCircuit

$$\#\text{3D-STRONGSPERNER} \subseteq^7 \#\text{PURECIRCUIT}$$

When referring to the fact that x is a solution, we imply the definition described in II.11. Additionally we will restrict the solution set of the *Purify* gate of PURECIRCUIT to $(0, \perp), (\perp, 1), (0, 1), (1, \perp)$. II-B2a explains that this does not affect the PPAD-HARDNESS of the problem nor any counting arguments. Lastly $\forall b \in \mathbb{B}^n, \bar{b}$ will denote the decimal representation of b.

Proof. Given input $(\lambda, 0^m)$, we first construct a colouring function $\Lambda : (\mathbb{B}^{(m+1)})^3$ such that:

$$\forall (i,j,k) \in (\mathbb{B}^{(m+1)})^3 : \Lambda(i,j,k) \triangleq \lambda\left(\left\lfloor \frac{\bar{i}+1}{2} \right\rfloor, \left\lfloor \frac{\bar{j}+1}{2} \right\rfloor, \left\lfloor \frac{\bar{k}+1}{2} \right\rfloor\right)$$

Conversely we can say that we map from our original domain to our new domain as such

$$\forall x \in (\mathbb{B}^m)^3 : \lambda(x) \to \{\Lambda[(\max\{2\bar{x}_i - b_i, 0\})_{i \in [3]}] \mid b \in \mathbb{B}^3\} \tag{1}$$

The transformation can be visualised in the figure 4. The reasoning for doubling the dimensions will be explained in the later sections.

Given that S is the original set of solutions as described in II.11, we define the claim III.1.

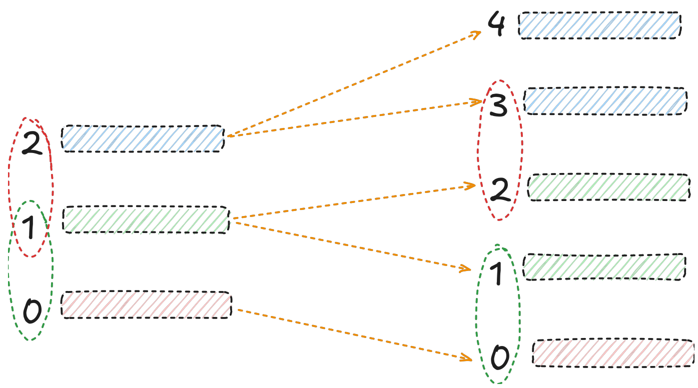


Fig. 4: Transformation of solutions. In the original cube assume we have solutions $(0, i, j)$ and $(1, i, j)$ for some i, j. In our new mapping, these solutions correspond to $(0, i, j), (2, i, j)$.

Claim III.1: Transformation claim

Given S_{even} the set of even solutions or more formally:

$$S_{\text{even}} \triangleq \left\{(i0, j0, k0) \in (\mathbb{B}^{(m+1)})^3 \mid (i0, j0, k0) \text{ covers all labels by II.9}\right\}$$

We claim $|S| = |S_{\text{even}}|$

Proof. We argue that we can bijectively map between $S \leftrightarrow S_{\text{even}}$. To show the left to right direction, assume $(i, j, k) \in S$. Using equation 1, only the point $(2\bar{i}, 2\bar{j}, 2\bar{k}) = (i0, j0, k0)$ will be mapped to all even coordinates. Additonally, we can observe its neighbourhood set:

$$N = \left\{\Lambda(2\bar{i} + x_1, 2\bar{j} + x_2, 2\bar{k} + x_3) \mid x \in \mathbb{B}^3\right\}$$

We can observe that WLOG $\Lambda(2i + 1, \cdot, \cdot) = \lambda(i + 1, \cdot, \cdot)$, which implies:

$$N = \left\{\lambda(\bar{i} + x_i, \bar{j} + x_j, \bar{k} + x_k) \mid x \in \mathbb{B}^3\right\}$$

And therefore the point $(i0, j0, k0)$ is also a solution. We can use the previous neighbourhood argument to prove the converse direction. □

We will redefine $m+1$ as n to keep the notation consistent. Given the above we provide a main sketch of our tranformation as such:

1) **Input bits**: We initialize input nodes s_1, s_2, s_3
2) **Bit generator**: We apply the *Bit Generator gadget* $\hat{P}$ for each s_i, to create numbers $u^1, u^2, u^3 \in \mathbb{B}^n$. We will use notation $\mathbf{x}[u^i]$ to describe the assingment of all nodes in u^i.
3) **Circuit application**: We apply circuit $\bar{\Lambda}(u^1, u^2, u^3)$ to create output values o_1, o_2, o_3, where $\bar{\Lambda}$ is a 3-bit hazard-free variant of Λ.
4) **Validation**: Copy the results to s_1, s_2, s_3

The transformation can be summarised in the figure 5. The goal of the above transformation is to show that if $o_1 = o_2 = o_3 = \perp$, then we have a solution to the original instance.

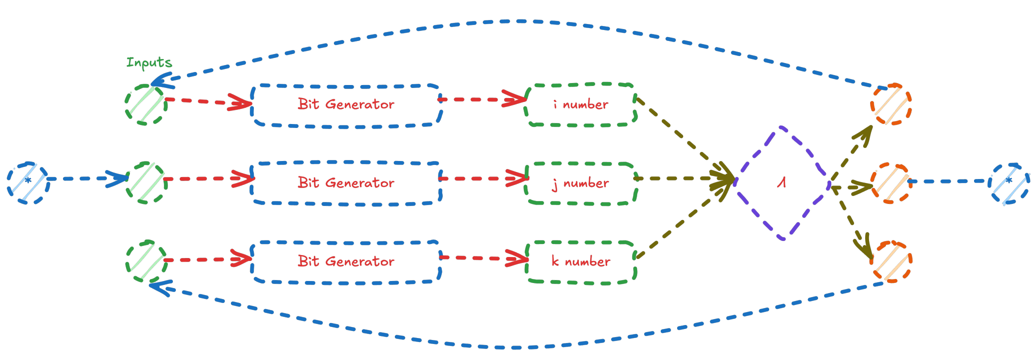


Fig. 5: Construction map. The green nodes indicate the start nodes. The green squares indicate u^1, u^2, u^3. The orange circle denote o_1, o_2, o_3 and the blue lines denote the *COPY* operation to the starting nodes.

a) Bit generator: We create the gadget described in the figure 6.

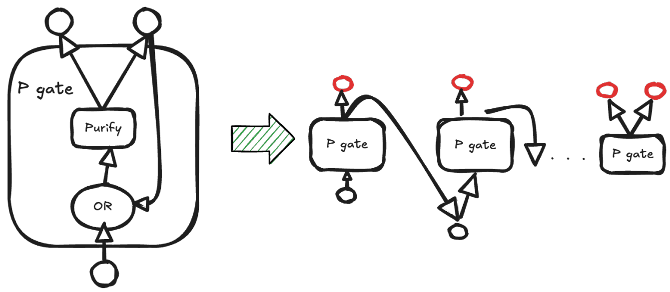


Fig. 6: Bit generator gadget which is defined as a linear stack of P gates. The red nodes denote the output of the construction.

Given that construction we make the following lemma

Lemma III.1 (Bit Generator Lemma)**.** *The following hold true in our construction:*

1) if $\mathbf{x}[s_i] = b \implies \mathbf{x}[u^i] = b^n$*, given* $b \in \mathbb{B}$
2) if $\mathbf{x}[s_i] = \perp \implies \forall j \in [n-1] : \mathbf{x}[u^i_j] \in \mathbb{B}$ *and* $\mathbf{x}[u^i_n] \in \{1, \perp\}$

The above translates to: if we have $\perp$ *in our number, it can only be found in the LSB.*

Proof. The first part follows trivially from the defintion of the *Purify* gate. To prove the second part of lemma, WLOG we choose u^i out of the three outputs and assume $\exists j \in [n-1] : \mathbf{x}[u^i_j] = \perp$. It implies that one of the purify gates had an output of $(\perp, 1)$. But due to our OR gate, the input would be 1 which implies the output is $1, 1$, which leads to a contradiction. □

b) Circuit Application: In the current phase, we apply a polynomial transformation to Λ to be 3-bit hazard-free using the construction by *Corollary 1.10* [20] or by *Corollary 1.9* [22]. By lemma III.1, we can have at most 3 $\perp$ values in our input. Using that property we create the lemma below:

Lemma III.1: Circuit Application Lemma

Given assignment $\mathbf{x}$ where $\mathbf{x}[o_1] = \mathbf{x}[o_2] = \mathbf{x}[o_3] = \perp$, it implies assingments of $\mathbf{x}[u^1], \mathbf{x}[u^2], \mathbf{x}[u^3]$ correspond to a solution $(i0, j0, k0) \in S_{\text{even}}$

Proof. First we argue that if $i\perp, j\perp, k\perp = \mathbf{x}[u^1], \mathbf{x}[u^2], \mathbf{x}[u^3]$, then its resolutions can represent hypercubes as shown in figure 7.

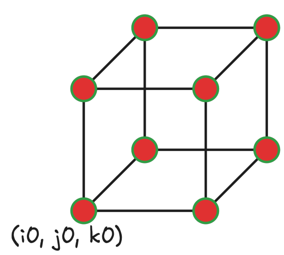


Fig. 7: Resolutions of $i\perp, j\perp, k\perp$. We say the coordinate with the smallest $\|\cdot\|$ anchors the cube

If $(i0, j0, k0)$ is a solution that implies:

$$\forall i \in \{1, 2, 3\}, \exists c, c' \in \textsc{Res}(i\perp, j\perp, k\perp) : [\bar{\Lambda}(c)]_i = +1 \land [\bar{\Lambda}(c')]_i = -1$$

Since $\bar{\Lambda}$ is 3-bit hazard free, if all outputs are $\perp$, we satisfy the colouring criteria II.9. □

This also highlights the necessity of the initial domain transformation: it guarantees that all cube-aligned solutions are anchored at the $(i0, j0, k0)$ coordinates. We can observe by fixing the LSB of the input to some value, we extract a specific edge or face of the cube.

Lemma III.2: Correctness lemma

A valid assignment $\mathbf{x}$ of the current instance, corresponds to a point in S_{even}

Proof. It suffices to show that a valid assignment only occurs when $\mathbf{x}[o_1] = \mathbf{x}[o_2] = \mathbf{x}[o_3] = \perp$. Assume $\exists i \in \{1, 2, 3\}$ such that label i is not covered, meaning $\mathbf{x}[o_i] \neq \perp$. WLOG assume that $\mathbf{x}[o_i] = 0$. Our verification stage, will copy 0 onto s_i. By lemma III.1, $\mathbf{x}[u^{(i)}] = 0^n$. From the boundary conditions of our circuits and the k-bit hazard-freeness construction, we know that $\Lambda(*, 0^n, *) = \{*, 1, *\}$. But that implies $\mathbf{x}[o_i] = 1$ which leads to a contradiction. We can make a similar argument to when $\mathbf{x}[o_i] = 1$. Therefore, by III.1, if $\mathbf{x}[o_1] = \mathbf{x}[o_2] = \mathbf{x}[o_3]\bot$, we have a found a panchromatic solution of S_{even}. ☐

☐

Counting Argument: We observe that the LSB can only be $1, \bot$. Moreover, we double count any edges or faces of the cube that cover all labels, which gives the equation:

$$\underbrace{\binom{3}{1}}_{\substack{\text{One of the sides is odd}\\ \text{and covers all labels}}} + \underbrace{\binom{3}{2}}_{\substack{\text{One of the edges is odd}\\ \text{and covers all labels}}} + \underbrace{1}_{\text{All LSBs are } \bot} = 2^3 - 1 = 7$$

Therefore, we can bound the number of solutions of 3D-StrongSperner by a factor of 7.

EndOfLine problem parsimoniously reduces to 2D-StrongSperner and 3D-StrongSperner under **linear** colouring [10], [11], but more work has to been done in order to determine whether this still holds for bipolar colouring.

The reduction described above can work with any dimensionality and $\forall n \in \mathbb{N}_{\geq 2}$, Chen et al. [12] showed that nD-StrongSperner is still *PPAD-Hard*. Therefore the following corollary holds III.1

Corollary III.1: nD-StrongSperner parsimonious reduction bounds

$$f(\cdot) = \sum_{i=1}^{n} \binom{n}{i} = 2^n - 1 = a_n$$

$$\#\text{nD-StrongSperner} \subseteq^{a_n} \#\text{PureCircuit}$$

2) Hazard Circuits and Tarski: We define the following subclass of Tarski II.14:

Definition III.1: KleeneTarski problem definition

Given $F : \mathbb{T}^n \to \mathbb{T}^n$, where F is a *natural* function II.17, find $x \in \mathbb{T}^n$ such that $F(x) = x$ We assume F will be represented as a circuit that uses set of gates $\{\wedge, \vee, \neg, \mathbf{0}, \mathbf{1}\}$.

Proposition III.2 (Parsimonious Reduction between KleeneTarski and PureCircuit)**.**

$$\#\text{KleeneTarski} \subseteq \#\text{PureCircuit}$$

Proof. Given an input circuit $C : \mathbb{T}^n \to \mathbb{T}^n$, construct PureCircuit instance:

1) Initiate a vector of n nodes, which we denote as s.
2) Construct the C using the standard set of gates and apply $C(x)$ to create output vector o.
3) Copy the results back into s.

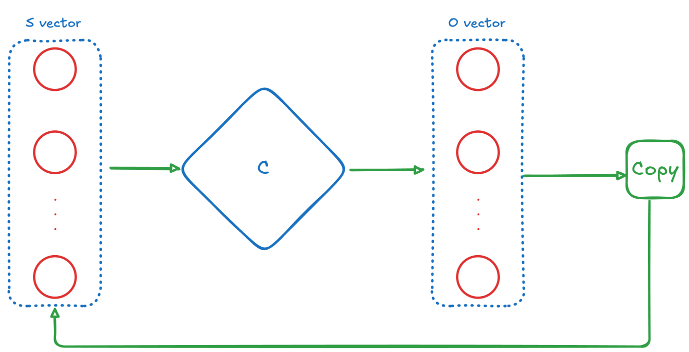


Fig. 8: Tarski construction

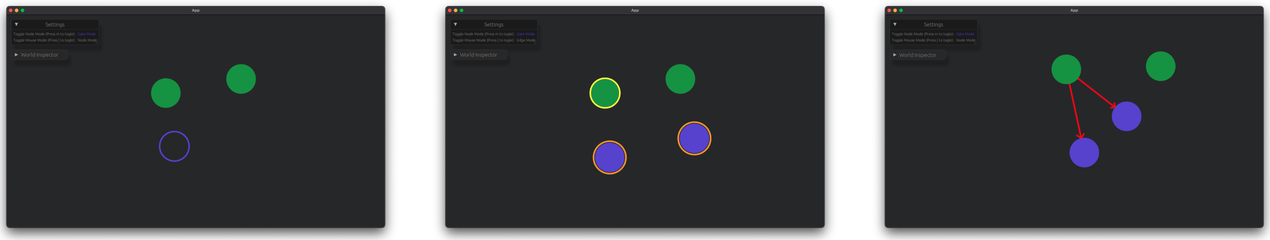
We can visualise the above construction in figure 8. We can observe that, for any valid assignment $\mathbf{x} \implies \mathbf{x}[s] = \mathbf{x}[o]$, which implies $\mathbf{x}[o] = F(x) = \mathbf{x}[s]$. ☐

B. Software Progress

In our project we are focus on developing a visualisation tool for PureCircuit instances. In our introduction we referred to three main milestones. As of time of writing we are close to the completion of the first one. We will provide a table of objectives that have been achieved, as well as the remaining requirements needed to complete the first milestone in II. Lastly, we provide a figure of our current visualisation in 9

Accomplished	Remaining
Ability to add and move nodes	Add values to value nodes
Toggle between value nodes and gate nodes	Specify the type of gate to add
Edge creation functionality. Ensure that edges can only be added between heterogeneous nodes	Add indicators as to how many edges are excess/remaining for the gate to be valid
Move whole graph	Inclusion of a status bar as to whether the current assignment is correct, wrong or syntactically incorrect
Create panel to show current state as well as some indicator and guides	

TABLE II: Finished and remaining issues



(a) Add nodes (b) Add edges (c) Final state

Fig. 9: Screenshots of different states of the visualisation tool

IV. Subsequent Steps

A. *Theory Crafting Next steps*

For the remaining duration, we will focus on three main objectives: One will be to demonstrate that, the *bipolar* colouring of nD-StrongSperner will keep the reduction parsimonious. Subsequently, we will to demonstrate hardness over #P class either by finding reductions from the SourceOrExcess problem. And lastly, we will be to generalise our reduction to all #PPAD problems.

B. *Software Next Steps*

In the next steps we aim to complete the rest of the milestones. We believe that the usage of methods by Eichelberger et al. [19] and or polynomial algorithm for specific cases by [9] will allow us to identify solutions. With regards to counting, we anticipate to implement this functionality for small instances.

C. *Timetable*

The figure 10 depicts how the remaining project will progress with regards to the milestones we aim to achieve. Overall we modified our timeline to prioritise theory crafting and therefore we hope this adjusted timeline will capture accurately the remaining duration of the project.

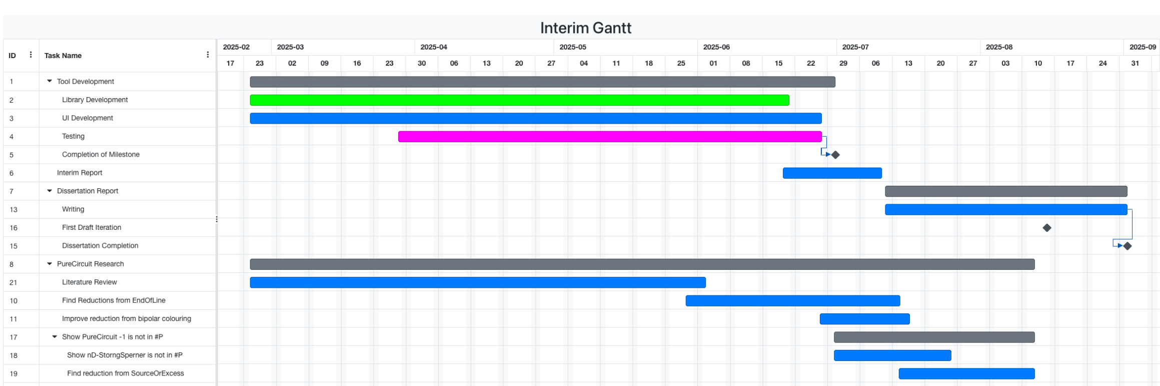


Fig. 10: Updated Gantt chart

V. Project Management

To manage our software we are using *Rust*, as the main language of development, due to its high expressibility and low level management. We utilise methods such as Test-Driven-Development with the help of `Proptest` for property testing and kanban boards to accurately track our progress as seen in figure 11. Lastly we are making use of *Bevy* which is a game engine that use the Entity-Component-System for render interactions between entities as seen in 11.

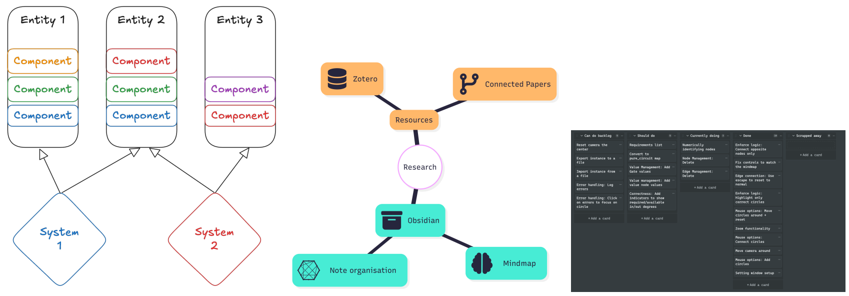
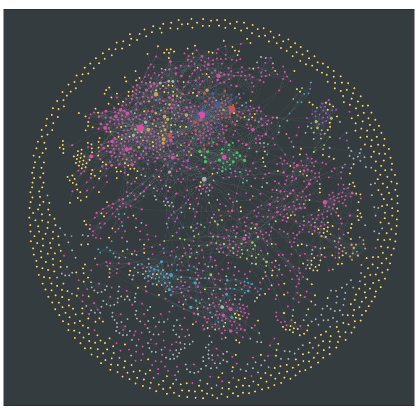
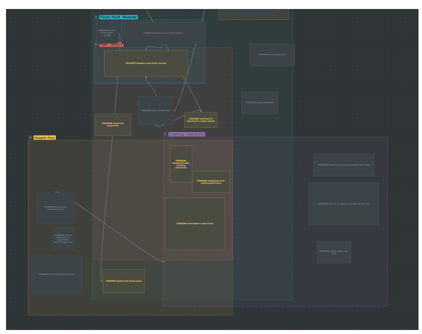


Fig. 11: ECS workflow visualisation. Entities which exist in a world state are equipped with components. Systems query entities with specific components and modify the world state.

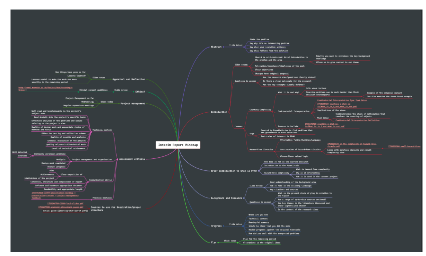
With regards to the our research management, we revolved our work around obsidian. As we can observe in the figure 12, `Obsidian` beyond its traditional usage of note taking, entails several handy tools such as note organisation and mind-mapping.



(a) Obsidian Graph



(b) Obsidian Canvas



(c) Obsidian mindmap

Fig. 12: Usages of Obsidian

A. Risk Management

With regards to risks and mitigations, the biggest risk we face is the inability to resolve our main question. The discovery of our reduction allowed us to step closer to our question. Our table III expand on the risks of the project as well as their mitigation tasks.

Severity	**Probability**	**Description**	**Mitigation**	**Address**
High	Medium	Software may not be feasible within the remaining time frame.	Focus on the the first objective where the project. Restrict to solution finding or counting when the number of nodes is small.	Not yet
Low	Medium	Software not identifying a correct solution	Usage of TDD techniques and property testing to ensure correctness. Comparison with hand-made instances	Not yet

High	High	Inability to extend the reductions of the PureCircuit problem to SourceOrExcess	Find a path of parsimonious reductions between the two problems. If we cannot ensure a one-to-one relation between the solutions, we will focus on $1-c$ for some $c \in \mathbb{N}$	✓
Medium	High	Inability to prove the main conjecture	We can make some heuristical arguments or find reductions between other problems. We argue that if we are able to unravel enough correlations, we will hope to at least bridge the problems. Develop useful gadgets one can use with PureCircuit	Not yet
High	High	Incorrect proofs or reductions.	Analyse the problem under different constraints. Apply the duck method, where attempt to explain the solution to a person which not necessarily an expert. Validate proof with supervisor	Not yet
High	Medium	Develop combinatorial friendly variants of PURECIRCUIT	Apply robustness on the gate set of PURECIRCUIT. Develop new gates or variants are easier to work with.	✓

TABLE III: Risk management table.

VI. Appraisals

A. Reflections

Overall, the current project progressed as anticipated. We managed to keep consistent progress in both theory and development aspect. In addition, due to the theoretical overlap the project has with other fields, we were able to experiment with great variety of problems as well as develop research methodologies for tackling these problems.

Of course understanding how to conduct such research came with a lot of trials and error. We quickly realised that such projects require a much more extensive literature review than anticipated. Moreover, when it comes to theorising and proof development, we faced many failures. Often, we spent an excessive amount of time working on a problem without being methodical. From that, we understood that when trying to prove statements, it is often more beneficial exploring the field as a whole, tackling various problems and gathering ideas from each. Moreover, each failed attempt allows us to understand a problem in greater depth, how it behaves and how to formulate our attempt to increase our chances. This can be observed in, we will analyze this is greater depth.ur canvas board 12b, where we used "whiteboard" approach to keep track of the project from a bird's-eye view. We hope to use this exploratory methodology to uncover even more mysteries that surround the complexity of counting problems.

References

[1] I. Pak, "What is a combinatorial interpretation?" Sep. 2022.

[2] R. H. Makar, "On the Analysis of the Kronecker Product of Irreducible Representations of the Symmetric Group," *Proceedings of the Edinburgh Mathematical Society*, vol. 8, no. 3, pp. 133–137, Dec. 1949.

[3] C. Ikenmeyer and I. Pak, "What is in #P and what is not?" in *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, Oct. 2022, pp. 860–871.

[4] C. Ikenmeyer, K. D. Mulmuley, and M. Walter, "On vanishing of Kronecker coefficients," *computational complexity*, vol. 26, no. 4, pp. 949–992, Dec. 2017.

[5] L. G. Valiant, "The complexity of computing the permanent," *Theoretical Computer Science*, vol. 8, no. 2, pp. 189–201, Jan. 1979.

[6] C. Ikenmeyer, I. Pak, and G. Panova, "Positivity of the Symmetric Group Characters Is as Hard as the Polynomial Time Hierarchy," *International Mathematics Research Notices*, vol. 2024, no. 10, pp. 8442–8458, May 2024.

[7] 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.

[8] S. Kleene and M. Beeson, *Introduction to Metamathematics*. Ishi Press International, 2009.

[9] A. Deligkas, J. Fearnley, A. Hollender, and T. Melissourgos, "Pure-Circuit: Tight Inapproximability for PPAD," *J. ACM*, vol. 71, no. 5, pp. 31:1–31:48, Oct. 2024.

[10] C. Daskalakis, P. Goldberg, and C. Papadimitriou, *The Complexity of Computing a Nash Equilibrium*. Association for Computing Machinery, Jan. 2006, vol. 39.

[11] X. Chen and X. Deng, "On the complexity of 2D discrete fixed point problem," *Theoretical Computer Science*, vol. 410, no. 44, pp. 4448–4456, Oct. 2009.

[12] 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.

[13] C. Daskalakis, S. Skoulakis, and M. Zampetakis, "The complexity of constrained min-max optimization," 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. 1466–1478.

[14] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis, "How easy is local search?" *Journal of Computer and System Sciences*, vol. 37, no. 1, pp. 79–100, Aug. 1988.

[15] K. Bronisław, "Un theoreme sur les functions d'ensembles," vol. 6, pp. 133–134, 1928.

[16] J. Fearnley, D. Pálvölgyi, and R. Savani, "A Faster Algorithm for Finding Tarski Fixed Points," *ACM Trans. Algorithms*, vol. 18, no. 3, pp. 23:1–23:23, Oct. 2022.

[17] S. Friedrichs, M. Függer, and C. Lenzen, "Metastability-Containing Circuits," *IEEE Transactions on Computers*, vol. 67, no. 8, pp. 1167–1183, Aug. 2018.

[18] D. Kozen, *Theory of Computation*, ser. Texts in Computer Science. Springer London, 2006.

[19] E. B. Eichelberger, "Hazard Detection in Combinational and Sequential Switching Circuits," *IBM Journal of Research and Development*, vol. 9, no. 2, pp. 90–99, Mar. 1965.

[20] 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.

[21] C. Ikenmeyer, B. Komarath, and N. Saurabh, "Karchmer-Wigderson Games for Hazard-free Computation," Nov. 2022.

[22] J. Bund, C. Lenzen, and M. Medina, "Small Hazard-Free Transducers," *IEEE Transactions on Computers*, vol. 74, no. 5, pp. 1549–1564, May 2025.

[23] M. Mukaidono, "On the B-ternary logic function," *Trans. IECE*, vol. 55, pp. 355–362, Jan. 1972.

[24] L. R. Marino, "General theory of metastable operation," *IEEE Transactions on Computers*, vol. C-30, no. 2, pp. 107–115, Feb. 1981.

[25] J. Fearnley, S. Gordon, R. Mehta, and R. Savani, "Unique end of potential line," *Journal of Computer and System Sciences*, vol. 114, pp. 1–35, Dec. 2020.