PLS is contained in PLC

Takashi Ishizuka
Artificial Intelligence Laboratory, Fujitsu Limited, Japan
ishizuka-t@fujitsu.com

December 8, 2023

Abstract

Recently, Pasarkar, Papadimitriou, and Yannakakis [PPY23] have introduced the new TFNP subclass called PLC that contains the class PPP; they also have proven that several search problems related to extremal combinatorial principles (e.g., Ramsey's theorem and the Sunflower lemma) belong to PLC. This short paper shows that the class PLC also contains PLS, a complexity class for TFNP problems that can be solved by a local search method. However, it is still open whether PLC contains the class PPA.

1 Introduction

1.1 Notation

First of all, we present terminologies that we will use in this short paper.

We denote by $\mathbb{Z}$ the set of all integers. For an integer $a \in \mathbb{Z}$, we define $\mathbb{Z}_{\geq a} := \{x \in \mathbb{Z} \ : \ x \geq a\}$ and $\mathbb{Z}_{>a} := \{x \in \mathbb{Z} \ : \ x > a\}$. We use $[n] := \{1, 2, \dots, n\}$ for every positive integer n in $\mathbb{Z}_{>0}$. Let X be a finite set. We denote by $|X|$ the cardinality of the elements in X. For any function $f : X \to X$ and any sequence of elements $\xi_0, \dots, \xi_k$ in X, the *unfilled set* of X is defined as $X \setminus \{f(\xi_0), \dots, f(\xi_k)\}$; we write this for $\mathrm{unfill}_f(X \mid \xi_0, \dots, \xi_k)$. When X is a finite set of integers, for a positive integer κ, we denote $X[k]$ to be the set of κ smallest elements of X; that is, $|X[\kappa]| = \kappa$ and $\xi < \eta$ for each pair of two elements $\xi \in X[\kappa]$ and $\eta \in X \setminus X[\kappa]$.

Let $\{0,1\}^*$ denote the set of binary strings with a finite length. For every string $x \in \{0,1\}^*$, we denote by $|x|$ the length of x. For each positive integer n, we write $\{0,1\}^n$ for the set of binary strength with the length n. Throughout this short paper, we sometimes regard $\{0,1\}^n$ as the set of positve integers $[2^n]$.

Search Problems Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be a relation. We say that R is *polynomially balanced* if there is a polynomial $p : \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ such that for each $(x, y) \in R$, it holds that $|y| \leq p(|x|)$. We say that R is *polynomial-time decidable* if for each pair of strings $(x, y) \in \{0,1\}^* \times \{0,1\}^*$, we can decide whether (x, y) belongs to R in polynomial time. We say that R is *total* if for every string $x \in \{0,1\}^*$, there always exists at least one string y such that $(x, y) \in R$.

For a relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$, the search problem with respect to R is defined as follows[1]: Given a string $x \in \{0,1\}^*$, find a string $y \in \{0,1\}^*$ such that $(x, y) \in R$ if such a y exists, otherwise reports "*no*." When R is also total, we call such a search problem a total search problem. The complexity class FNP is the set of all search problems with respect to a polynomially balanced and polynomial-time decidable relation R. The complexity class TFNP is the set of all total search problems belonging to FNP. By definition, it holds that TFNP $\subseteq$ FNP.

[1]For simplicity, we call the search problem with respect to R the search problem R.

Reductions Let $R, S \subseteq \{0,1\}^* \times \{0,1\}^*$ be two search problems. A polynomial-time reduction from R to S is defined by two polynomial-time computable functions $f : \{0,1\}^* \to \{0,1\}^*$ and $g : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ satisfying that $(x, g(x,y)) \in R$ whenever $(f(x), y) \in S$. In other words, the function f maps an instance x of R to an instance $f(x)$ of S, and the other function g maps a solution y to the instance $f(x)$ to a solution $g(x,y)$ to the instance x.

For a complexity class $\mathcal{C}$, we say that a search problem R is $\mathcal{C}$-hard if all search problems in $\mathcal{C}$ are polynomial-time reducible to R. Furthermore, we say that a search problem R is $\mathcal{C}$-complete if R is $\mathcal{C}$-hard, and R belongs to $\mathcal{C}$.

1.2 Backgrounds

Consider the following two-player game: There are 2^n stones; we denote by U_0 the set of all stones. In the first round, Player 1 chooses one stone a_0 from U_0, then Player 2 partitions remaining stones $U_1 := U_0 \setminus \{a_0\}$ into two groups, denoted by U_1^0 and U_1^1. In the second round, Player 1 chooses one stone a_1 from $U_1^{b_1}$, then all stones in the opposite to a_1 (i.e., all stones in $U_1^{1-b_1}$) are removed from the game immediately. Player 2 partitions all stones in the group $U_2 := U_1^{b_1} \setminus \{a_1\}$ into two groups, denoted by U_2^0 and U_2^1. In the i-th round, Player 1 chooses one stone a_{i-1} from $U_{i-1}^{b_{i-1}}$, then all stones in $U_{i-1}^{1-b_{i-1}}$ are removed from the game immediately. Player 2 partitions all stones in the group $U_i := U_i^{b_{i-1}} \setminus \{a_{i-1}\}$ into two groups U_i^0 and U_i^1. They repeat such processes $n+1$ rounds. Player 1 wins if they can pick $n+1$ distinct stones $a_0, a_1, \ldots, a_n$ at the end of the game. If Player 1 cannot choose any stones during the above game, then Player 2 wins. In this paper, we call this game the Interactive Bipartition Stone-Picking Game[2].

It is straightforward to see that a winning strategy for Player 1 in Interactive Bipartition Stone-Picking Game always exists. Recently, Pasarkar, Papadimitriou, and Yannakakis [PPY23] have formulated the problem of finding a winning strategy for Player 1 in Interactive Bipartition Stone-Picking Game as a TFNP problem. Informally speaking, their problem, called Long Choice, is to find a sequence of distinct elements that satisfies suitable properties when we are given a description of Player 2's action at each round. The formal definition can be found in Definition 3.

TFNP problems [MP91; Pap94] — the existence of solutions guarantees, and the correctness of every solution is effortlessly checkable — comprise a fascinating field in computational complexity theory. It is known that many significantly important computational problems belong to the complexity class TFNP. For example, finding a Nash equilibrium [CDT09; DGP09], computing a fair division [FG18; DFM22; GHH23], integer factoring [Bur06; Jer16], and algebraic problems related to cryptographies [SZZ18; HV21]. A natural way to analyze the theoretical features of a complexity class is to characterize its class by complete problems. However, it is widely believed that TFNP has no complete problem [Pud15; Pap94]. Consequently, several TFNP subclasses with complete problems have been introduced over the past three decades. The best well-known such classes include PLS [JPY88], PPAD, PPA, PPP [Pap94], PWPP [Jer16], and EOPL [DP11; Fea+20; Göö+22].

We are interested in the boundary of total search problems. In particular, our central motive is to capture the most hard problems among *syntactic* TFNP problems. Previously, Goldberg and Papadimitriou [GP18] have introduced a TFNP problem that unifies the traditional TFNP subclasses. However, we are unaware of another TFNP problem that unifies PLS, PPP, and PPA. As a first step, this short paper sheds light on the relationship between the oldest and the newest TFNP subclasses.

1.3 Our Contributions

We make clear the relationship between two TFNP subclasses: PLS and PLC.

[2]Remark that the term "bipartition" implies that, at each turn, Player 2 partitions the current set into two groups. This game can be easily generalized to the multi-partition setting.

The complexity class PLS, introduced by Johnson, Papadimitriou, and Yannakakis [JPY88], is one of the most famous TFNP subclasses. The class PLS captures the complexity of the problems that can be solved by a local search method. Formally, this class is defined as the set of all search problems that are reducible to LocalOPT in polynomial time.

Definition 1. LocalOPT
Input: Two Boolean circuits $f : [2^n] \to [2^n]$ and $p : [2^n] \to [2^m]$.
Output: A point x in $[2^n]$ such that $p(x) \geq p(f(x))$

Definition 2. The complexity class PLS is the set of all search problems that are reducible to LocalOPT in polynomial time.

On the other hand, the complexity class PLC is the newest TFNP subclass, introduced by Pasarkar, Papadimitriou, and Yannakakis [PPY23]. This class is formulated as the set of all search problems that are reducible to the problem Long Choice in polynomial time.

Definition 3. Long Choice
Input: $n-1$ Boolean circuits $P_0, \dots, P_{n-2}$ such that $P_i : ([2^n])^{i+2} \to \{0,1\}$ for each $i \in \{0, \dots, n-2\}$
Output: A sequence of $n+1$ distinct elements $a_0, a_1, \dots, a_n$ in $[2^n]$ such that for each $i \in \{0, \dots, n-2\}$, $P_i(a_0 \dots, a_i, a_j)$ is the same for every $j > i$.

Definition 4. The complexity class PLC is the set of all search problems that are reducible to Long Choice in polynomial time.

As mentioned before, the class PLC captures the complexity of finding a winning strategy for Player 1 in Interactive Bipartition Stone-Picking Game. Let $P_0, \dots, P_{n-2}$ be a sequence of the predicates given by an instance of Long Choice. For each index $i \in \{0, \dots, n-2\}$, the predicate $P_i : ([2^n])^{i+2} \to \{0,1\}$ represents Player 2's behavior at the $(i+1)$th round. Then, we can easily regard a solution to Long Choice as a winning strategy for Player 1.

We show that the complexity class PLS is contained in PLC.

Theorem 5 (Main Contribution)**.** PLS *is contained in* PLC.

2 Technical Ingredients

To prove our main theorem, we introduce a search problem, an extension of Pigeon (see Definition 6). The complexity class PPP, introduced by Papadimitriou [Pap94], is the class for search problems related to the pigeonhole principle. A total search problem belonging to PPP is to find a collision under the self-mapping. For instance, consider the situation where we put 2^n pigeons in 2^n cages according to a function $C : [2^n] \to [2^n]$. Unfortunately, one of these cages is broken, denoted by v^*, and we cannot use it. The task of the problem is to find a collision or detect the broken cage being used.

The formal definition of the canonical PPP-complete problem is as follows.

Definition 6. PIGEON
Input: A Boolean circuit $C : [2^n] \to [2^n]$ and a special element $v^* \in [2^n]$
Output: One of the following

1. two distinct elements $x, y \in [2^n]$ such that $C(x) = C(y)$
2. an element $x \in [2^n]$ such that $C(x) = v^*$

Definition 7. The complexity class `PPP` is the set of all search problems that are reducible to PIGEON in polynomial time.

Theorem 8 (Pasarkar, Papadimitriou, and Yannakakis [PPY23])**.** `PPP` *is contained in* `PLC`*.*

From now on, we extend PIGEON to another `TFNP` problem called QUOTIENT PIGEON. Let U be a finite set, and let $\sim$ denote an equivalence relation over U. We now consider a PIGEON instance over the quotient set $U/\sim$. In other words, we focus on the following search problem: Given a function $C : U/\sim \to U/\sim$ and a special element v^* in U, find two distinct elements $x, y \in U/\sim$ such that $C(x) \sim C(y)$ or an element $x \in U/\sim$ such that $C(x) \sim v^*$. For example, consider the situation where we put N books away into M bookshelves, but one of these bookshelves is broken; namely, this one cannot be used. Our behavior can be represented by a function $C : [N] \to [M]$. We sort these books by genre, which can be classified into exactly M genres. Note that genre induces an equivalence relation E over these books. The task of the problem is to detect that two books of different genres are stored on the same bookshelf or that the broken bookshelf is being used.

To formulate the above variant of PIGEON, we allow to obtain another function $E : U \times U \to \{0, 1\}$ computing an equivalence relation over U. We denote by $\sim_E$ the binary relation defined by E; for each pair of elements x, y in U, $x \sim_E y$ if and only if $E(x, y) = 1$. Formally, the new search problem called QUOTIENT PIGEON is defined as follows.

Definition 9. QUOTIENT PIGEON
Input: Two Boolean circuits $C : [2^n] \to [2^n]$ and $E : [2^n] \times [2^n] \to \{0, 1\}$ and an element $v^* \in [2^n]$
Ouput: One of the following

1. two elements $x, y \in [2^n]$ such that $x \not\sim_E y$ and $C(x) \sim_E C(y)$
2. an element $x \in [2^n]$ such that $C(x) \sim_E v^*$
3. two elements $x, y \in [2^n]$ such that $x \sim_E y$ and $C(x) \not\sim_E C(y)$
4. an element $x \in [2^n]$ such that $E(x, x) = 0$
5. two elements $x, y \in [2^n]$ such that $E(x, y) \neq E(y, x)$.
6. three distinct elements $x, y, z \in [2^n]$ such that $x \sim_E y$, $y \sim_E z$, and $x \not\sim_E z$

Unfortunately, we are unaware of a way of syntactically enforcing the Boolean circuit E to compute an equivalence relation over the finite set $[2^n]$. Thus, we introduce violations as solutions to QUOTIENT PIGEON to ensure that this problem belongs to `TFNP`. More precisely, the fourth-type solution is a violation of the *reflexivity*. The fifth-type solution represents a violation of the *symmetry*. Finally, the sixth-type solution means a violation of the *transivity*.

Proposition 10. *QUOTIENT PIGEON is* `PPP`*-hard.*

Proof. It suffices to define the Boolean circuit $E : [2^n] \times [2^n] \to \{0,1\}$ as $E(x, y) = 1$ if and only if $x = y$ for all $x, y \in [2^n]$. $\square$

Before closing this section, we observe the useful properties of Quotient Pigeon. First, we show that we can assume that there is no fixed point for any Quotient Pigeon without loss of generality.

Proposition 11. *Let $\langle C : [2^n] \to [2^n], E : [2^n] \times [2^n] \to \{0,1\}, v^* \in [2^n]\rangle$ be an instance of Quotient Pigeon. We can assume that there is no element $x \in [2^n]$ such that $C(x) \sim_E x$ or $C(x) = v^*$ without loss of generality.*

Proof. We first redefine the Boolean circuit C as follows: For each element $x \in [2^n]$, $C(x) := u^*$ if $C(x) \sim_E v^*$, otherwise we do not modify the output of $C(x)$. Here, we pick arbitrary element u^* in $[2^n]$ with $u^* \neq v^*$. It is straightforward to see that we can recover a solution to the original instance from a solution to the modified instance.

We write U for the set $\{0,1\} \times [2^n]$. We construct new two Boolean circuits $C' : U \to U$ and $E' : U \times U \to \{0,1\}$ as follows. For each element $(b, x) \in U$, we define $C'(b, x) := (1 - b, C(x))$. For each pair of two elements (b, x) and (c, y) in U, define $E'((b, x), (c, y)) = 1$ if and only if $b = c$ and $E(x, y) = 1$. Informally speaking, we create a copy of equivalent classes induced by E. Finally, we define $(0, v^*)$ in U to be a special element for the reduced instance of Quotient Pigeon.

Since the first bit is always flipped by the Boolean function C', there is no fixed point of C'. Furthermore, it holds that $\xi \not\sim_{E'} \eta$ for all elements $\xi, \eta \in U$ whose first bits are different. This implies that there is no element $\xi \in U$ such that $C'(x) \sim_E$.

From our construction, it is easy to see that we can efficiently recover an original solution from a solution to the reduced instance. $\square$

Next, we show that we can suppose that there exist at least $2n$ equivalent classes without loss of generality. More precisely, we can assume that the elements $u_0 := v^*, u_1 := C(u_0), \ldots, u_i := C(u_{i-1}), \ldots, u_{2n}$ are distinct under $\sim_E$, i.e., $u_i \not\sim_E u_j$ for all $0 \leq i < j \leq 2n$, without loss of generality.

Proposition 12. *For every non-trivial[3] instance of Quotient Pigeon $\langle C : [2^n] \to [2^n], E : [2^n] \times [2^n] \to \{0,1\}, v^* \in [2^n]\rangle$, we can assume that the elements $u_0 := v^*, u_1 := C(u_0), \ldots, u_i := C(u_{i-1}), \ldots, u_{2n}$ are distinct under $\sim_E$, i.e., $u_i \not\sim_E u_j$ for all $0 \leq i < j \leq 2n$, without loss of generality.*

Proof. It is sufficient to prove that we can easily recover a solution to the original Quotient Pigeon instance when there are two elements u_i and u_j such that $u_i \sim_E u_j$. We prove this fact by induction.

In the base case, we can assume that $u_0 \not\sim_E u_1 = C(u_0)$ from Proposition 11.

In the inductive case, suppose that the elements $u_0 := v^*, u_1 := C(u_0), \ldots, u_i := C(u_{i-1})$ are distinct under $\sim_E$ for some positive integer $i < 2n$. If the element $u_{i+1} := C(u_i)$ collided with an element u_j under $\sim_E$ for some $0 \leq j \leq i$, we can effortlessly recover a solution to the original Quotient Pigeon instance as follows: The element u_i is the second-type solution when $j = 0$; and the elements u_i, u_{j-1} is the first-type solution since $u_i \not\sim_E u_{j-1}$ and $C(u_i) \sim_E u_{j-1}$ when $j > 0$. $\square$

3 Proof of Our Main Theorem

Theorem 5 immediately follows from the following two lemmata.

Lemma 13. *Quotient Pigeon is* `PLS`*-hard.*

Lemma 14. *Quotient Pigeon belongs to* `PLC`.

[3]We say that an instance is *trivial* if it can be solved by a nïve approach in polynomial time.

In other words, we first prove that there is a polynomial-time reduction from LocalOPT to Quotient Pigeon in Lemma 13. After that, we show a polynomial-time reduction Quotient Pigeon to Long Choice in Lemma 14. By the transitivity of the polynomial-time reduction, we have a polynomial-time reduction from LocalOPT to Long Choice. This implies that LocalOPT belongs to the complexity class PLC; therefore, we conclude our main theorem: PLS $\subseteq$ PLC.

The proofs of Lemma 13 and Lemma 14 can be found in Section 3.1 and Section 3.2, respectively.

3.1 Proof of Lemma 13

To prove this lemma, we show a polynomial-time reduction from LocalOPT to Quotient Pigeon. Our proof is inspired by the robustness proof of End of Potential Line by Ishizuka [Ish21].

Let two Boolean circuits $f : [2^n] \to [2^n]$ and $p : [2^n] \to [2^m]$ be an instance of LocalOPT. We first show, in Proposition 15, that we can assume that the point $1 \in [2^n]$ has the unit potential, and every point $x \in [2^n]$ with $f(x) \neq x$ satisfies that $p(f(x)) = p(x) + 1$, without loss of generality.

Proposition 15. *For every LocalOPT instance $\langle f : [2^n] \to [2^n], p : [2^n] \to [2^m]\rangle$, we have a polynomial-time reduction from $\langle f, p\rangle$ to a LocalOPT instance $\langle F : [2^m] \times [2^n] \to [2^m] \times [2^n], P : [2^m] \times [2^n] \to [2^m]\rangle$ that satisfies the following two properties: (1) For each point ξ in $[2^m] \times [2^n]$ with $F(\xi) \neq \xi$, it holds that $P(F(\xi)) = P(\xi) + 1$; and (2) we know a special point v^* in $[2^m] \times [2^n]$ with $P(v^*) = 1$.*

Proof. Let two Boolean circuits $f : [2^n] \to [2^n]$ and $p : [2^n] \to [2^m]$ be an instance of LocalOPT. We first reduce the above instance to another LocalOPT instance $\langle f' : [2^n] \to [2^n], p' : [2^n] \to [2^m]\rangle$ satisfying that $p'(1) = 1$. For each point $x \in [2^n]$, we define

$$f'(x) := \begin{cases} f(1) & \text{if } f(x) = 1 \\ f(x) & \text{otherwise,} \end{cases}$$

and $p'(x) = p(x)$ if $x \neq 1$, otherwise $p'(x) = 1$. It is easy to see that the instance $\langle f', p'\rangle$ holds the desired condition.

What remains is to prove that we can recover a solution to the original instance $\langle f, p\rangle$ from a solution to the new instance $\langle f', p'\rangle$ in polynomial time. Let $x \in [2^n]$ be a solution to $\langle f', p'\rangle$; that is, it holds that $p'(x) \geq p'(f'(x))$. First, we suppose that $x = 1$. This implies that $p(x) \geq p'(x) = 1 = p'(f'(x)) = p(f(x))$. Hence, the point $x = 1$ is a solution to the original instance. Next, we suppose that $x > 1$, $f(x) = 1$, and the special point 1 is not a solution to $\langle f', p'\rangle$. This implies that $p(x) = p'(x) \geq p'(f'(x)) = p(f(1))$. Therefore, we can see that at least one of x and $f(x)$ is a solution to the original instance $\langle f, p\rangle$. Finally, we suppose that $x > 1$ and $f(x) \neq 1$. This implies that $p(x) = p'(x) \geq p'(f'(x)) = p(f(x))$. Thus, the point x is a solution to the original instance.

We move on to proving another desired condition: For every point, its potential increases at most one. We will construct another LocalOPT instance $\langle F : [2^m] \times [2^n] \to [2^m] \times [2^n], P : [2^m] \times [2^n] \to [2^m]\rangle$ from the LocalOPT instance $\langle f' : [2^n] \to [2^n], p' : [2^n] \to [2^m]\rangle$ such that $P(F(i, x)) = P(i, x) + 1$ for each point $(i, x) \in [2^m] \times [2^n]$ with $F(i, x) \neq (i, x)$. Our idea is inspired by [Fea+20, Theorem 4].

For each vertex $(i, x) \in [2^m] \times [2^n]$, we say that (i, x) is *active* if it holds that $p'(x) \leq i < p'(f'(x))$; the vertex (i, x) is *inactive* if it is not active. For every inactive vertex $(i, x) \in [2^m] \times [2^n]$, we define the function $F(i, x) := (p(x), x)$. For each active vertex $(i, x) \in [2^m] \times [2^n]$, we define the function $F : [2^m] \times [2^n] \to [2^m] \times [2^n]$ as follows:

$$F(i, x) := \begin{cases} (i + 1, x) & \text{if } p'(x) \leq i < p'(f'(x)) - 1, \\ (p'(f'(x)), f'(x)) & \text{if } i = p'(f'(x)) - 1. \end{cases}$$

Furthermore, we define the potential function $P : [2^m] \times [2^n] \to [2^m]$ as follows: $P(i, x) = i$ if a vertex (i, x) is active, otherwise $P(i, x) = p'(x) - 1$. It is not hard to see that every vertex satisfies the desired condition. We can effortlessly obtain a solution to the original instance from every solution to the new instance. ☐

We move on to describe how to construct the reduced Quotient Pigeon instance $\langle C : [2^n] \to [2^n], E : [2^n] \times [2^n] \to \{0,1\} \rangle$. First, we define the equivalence relation $E : [2^n] \times [2^n] \to \{0,1\}$ with respect to the LocalOPT instance $\langle f, p \rangle$. For all $x, y \in [2^n]$, define $E(x,y) := 1$ if $p(x) = p(y)$; otherwise $E(x,y) = 0$. It is straightforward to see that the function E satisfies the requirements for the equivalence relation. There is no solution that holds the fourth, fifth, or sixth type of solution. We define the Boolean circuit $C : [2^n] \to [2^n]$ as follows: $C(x) = f(x)$ for every $x \in [2^n]$. Finally, we set the special element v^* to be 1. We complete constructing a Quotient Pigeon instance $\langle C, E, v^* \rangle$.

What remains is to prove that we can recover a solution to the original LocalOPT instance from each solution to the reduced Quotient Pigeon in polynomial time. Since the Boolean circuit E certainly computes the equivalence relation over $[2^n]$, every solution to the reduced instance $\langle C, E, v^* \rangle$ is one of the first-, second-, and third-type solutions.

First-type solution We first consider where we obtain two elements $x, y \in [2^n]$ such that $x \not\sim_E y$ and $C(x) \sim_E C(y)$. This implies that $p(x) \neq p(y)$ but $p(f(x)) = p(f(x))$. Suppose that the point x is not a solution to the LocalOPT instance (i.e., $f(x) \neq x$ and $p(f(x)) = p(x) + 1$), the other point y is a solution to the LocalOPT instance. Similarly, suppose that y is not a solution; the other point x is a solution. Hence, at least one of x and y is a solution to the LocalOPT instance $\langle f, g \rangle$.

Second-type solution Next, we consider the case where we obtain an element $x \in [2^n]$ such that $C(x) \sim_E v^*$. This implies that $1 = p(v^*) = p(f(x)) \leq p(x)$. Therefore, the point x is a solution to the LocalOPT instance $\langle f, g \rangle$.

Third-type solution Finally, we consider the case where we obtain two distinct elements $x, y \in [2^n]$ such that $x \sim_E y$ and $C(x) \not\sim_E C(y)$. This implies that $p(x) = p(y)$ but $p(f(x)) \neq p(f(y))$. From our assumption, exactly one of the points x and y is a fixed point of f. Therefore, we obtain a solution to the original LocalOPT instance $\langle f, g \rangle$.

3.2 Proof of Lemma 14

This section proves that the problem Quotient Pigeon belongs to the class PLC. To prove this, we will provide a polynomial-time reduction from Quotient Pigeon to Constrained Long Choice, a restricted variant of the problem Long Choice. Our reduction heavily relies on the PPP-hardness proof of Long Choice by Pasarkar, Papadimitriou, and Yannakakis [PPY23, Theorem 2].

Definition 16. Constrained Long Choice
Input: $n-1$ Boolean circuits $P_0, \dots, P_{n-2}$ such that $P_i : ([2^n])^{i+2} \to \{0,1\}$ for each $i \in \{0, \dots, n-2\}$ and an initial element a_0 in $[2^n]$
Output: a sequence of $n+1$ distinct elements $a_0, a_1, \dots, a_n$ in $[2^n]$ such that for each $i \in \{0, \dots, n-2\}$, $P_i(a_0 \dots, a_i, a_j)$ is the same for every $j > i$.

Proposition 17 (Pasarkar, Papadimitriou, and Yannakakis [PPY23])**.** *Long Choice and Constrained Long Choice are polynomial-time reducible to each other.*

Let two Boolean circuits $C : [2^n] \to [2^n]$ and $E : [2^n] \times [2^n] \to \{0,1\}$ and an element $v^* \in [2^n]$ be an instance of Quotient Pigeon. From Proposition 11, we assume that $C(x) \not\sim_E x$ and $C(x) \neq v^*$ for every $x \in [2^n]$, without loss of generality. Also, from Proposition 12, we assume that the $2n$ elements $u_0 := v^*, u_1 := C(u_0), \ldots, u_i := C(u_{i-1}), \ldots, u_{2n} := C(u_{2n-1})$ are distinct under $\sim_E$ each other (i.e., $u_i \not\sim_E u_j$ for all $0 \leq i < j \leq 2n$), without loss of generality. We will construct the instance of Constrained Long Choice $\langle P_0, P_1, \ldots, P_{n-2}, v^* \rangle$, where $P_i : ([2^n])^{i+2} \to \{0, 1\}$ for every $i \in \{0, 1, \ldots, n-2\}$.

Before constructing the predicates $P_0, P_1, \ldots, P_{n-2}$, we briefly sketch our reduction. Every solution to Constrained Long Choice is a sequence of distinct elements $a_0, a_1, \ldots, a_n$ in $[2^n]$. In such a sequence, each element a_i is chosen by depending only on the previously chosen elements $a_0, a_1, \ldots, a_{i-1}$. Here, we use the terminology *distinct* to mean that $x \neq y$. Recall that two distinct elements x and y such that $x \sim_E y$ and $C(x) = C(y)$ are not a solution to Quotient Pigeon. So, we need to avoid such a *bad* solution being constructed as a solution to Constrained Long Choice. In order to settle such an issue, we arrange the sequence before applying the predicates $P_0, P_1, \ldots, P_{n-2}$ that are defined in [PPY23]. We introduce the sub-procedures $\beta_0, \beta_1, \ldots, \beta_{n-1}$, where for each index $k \in \{0, 1 \ldots, n-1\}$, the sub-procedure β_k maps distinct $k+1$ elements $a_0, \ldots, a_k, a_{k+1}$ to $k+1$ elements $b_0, \ldots, b_k, b_{k+1}$ with a suitable property.

Roughly speaking, we require that the elements $b_0, \ldots, b_n$ are distinct under $\sim_E$ unless we can recover a solution to the original instance in polynomial time. Each element b_{k+1} depends only on $b_0, \ldots, b_k$, and a_{k+1}. Furthermore, if the sequence $(b_0, \ldots, b_n, b_{n+1})$ contains two distinct elements b_i, b_j such that $b_i \sim_E, b_j$, then we can recover a solution to the original Quotient Pigeon instance $\langle C, E, v^* \rangle$ from $(b_0, \ldots, b_n, b_{n+1})$ in polynomial time.

We first describe how to construct the sub-procedures $\beta_0, \ldots, \beta_{n-1}$. These sub-procedures are defined inductively. We can find the formal structure for each index $k \in \{0, \ldots, n-1\}$ in Algorithm 1.

Let $(a_0, \ldots, a_n)$ denote an input sequence of the sub-procedures, where it holds that $a_i \neq a_j$ for all $0 \leq i < j \leq n$. In the base case (i.e., $k = 0$), the sub-procedure β_0 directly outputs b_0 to be a_0. Suppose that for an index $k < n-1$, we have a sequence of elements $b_0, \ldots, b_k$ constructed by the sub-procedure β_{k-1}. We now define the element b_{k+1} from the elements $b_0, \ldots, b_k$, and a_{k+1}.

First, we check whether the elements $b_0, \ldots, b_k$ are distcinct under $\sim_E$. If not, we set it to be $b_{k+1} := a_{k+1}$. Otherwise, we also check whether $a_{k+1} \not\sim_E b_i$ for every $i \in \{0, 1, \ldots, k\}$. If yes, we define $b_{k+1} := a_{k+1}$. Also otherwise, we have the elements $b_0, \ldots, b_k$ and a_{k+1} such that (1) $b_i \not\sim_E b_j$ for all $0 \leq i < j \leq n$; and (2) there exists an elements b_i such that $a_{k+1} \sim_E b_i$. In this case, we verify whether we can recover a solution to the original Quotient Pigeon $\langle C, E, v^* \rangle$ from the elements $b_0, \ldots, b_k$ and a_{k+1} in polynomial time. Specifically, we check the following six properties:

1. There exist two elements b_i and b_j such that $b_i \not\sim_E b_J$ and $C(b_i) \sim_E C(b_j)$;
2. there is a elements x in $\{b_0, b_1, \ldots, b_k, a_{k+1}\}$ such that $C(x) \sim_E v^*$;
3. there is an element b_i such that $b_i \sim_E a_{k+1}$ and $C(b_i) \not\sim_E C(a_{k+1})$;
4. there exists an element x in $\{b_0, b_1, \ldots, b_k, a_{k+1}\}$ such that $E(x, x) = 0$;
5. there are two elements x and y in $\{b_0, b_1, \ldots, b_k, a_{k+1}\}$ such that $C(x, y) \neq E(y, x)$; and
6. there exist distinct three elements $x, y, z \in \{b_0, b_1, \ldots, b_k, a_{k+1}\}$ such that $x \sim_E y$, $y \sim_E z$, and $x \not\sim_E z$.

We call the algorithm that performs these above tests CheckSolutions (see also, Algorithm 2).

If it passes at least one of the above six tests, then we can efficiently recover a solution to the original Quotient Pigeon $\langle C, E, v^* \rangle$. Note that these tests can be computed in polynomial time. Thus, we define $b_{k+1} := a_{k+1}$. Finally, if the sequence of elements $(b_0, \ldots, b_k, a_{k+1})$ is rejected by all of the above six tests, then we find the smallest positive integer ℓ such that $C^\ell(v^*) \not\sim_E x$ for every element $x \in \{b_0, \ldots, b_k, a_{k+1}\}$, and we define $b_{k+1} := C^\ell(v^*)$. Note that such an integer ℓ always exists and is bounded by $2n$ from our assumption.

We complete constructing the sub-procedures $\beta_0, \dots, \beta_{n-1}$. It is not hard to see that every sub-procedure β_k is polynomial-time computable. Moreover, a sequence of elements $b_0, \dots, b_n$ defined by our sub-procedures holds the next proposition.

Proposition 18. *For every positive integer $k \in [n-1]$, and for every equence of elements $a_0, \dots, a_k, a_{k+1}$ in $[2^n]$ such that $a_i \neq a_j$ for all $0 \leq i < j \leq k+1$, if the sequence of elements $b_0, \dots, b_k, b_{k+1}$ that are produced by $\beta_k(a_0, \dots, a_k, a_{k+1})$ are not distinct under $\sim_E$ (i.e., there is a pair of elements b_i and b_j such that $b_i \sim_E b_j$), then the algorithm* CheckSolutions$(b_0, \dots, b_k, b_{k+1})$ *returns* True.

Proof. We prove this by induction. In the case where $k = 1$, the statement is trivial.

Let k be a positive integer in $[n-1]$, and let $a_0, \dots, a_k, a_{k+1}$ denote a sequence of elements in $[2^n]$ such that $a_i \neq a_j$ for all $0 \leq i < j \leq k+1$. Also, we write $b_0, \dots, b_k, b_{k+1}$ for the corresponding sequence of elements. Suppose that the statement holds for every $i < k$; that is, the algorithm CheckSolutions$(b_0, \dots, b_i, b_{i+1})$ returns True if we have two elements b_{j^1} and b_{j^2} with $0 \leq j^1 < j^2 \leq i+1$ such that $b_{j^1} \sim_E b_{j^2}$. We will show that the statement also follows for the index k.

We assume that there exist two elements b_i and b_j with $0 \leq i < j \leq k+1$ such that $b_i \sim_E b_j$. From the inductive supposition, we consider the case where we have an element b_j such that $b_j \sim_E b_{k+1}$. By the constrcution of the sub-procedure β_k, the algorithm CheckSolutions$(b_0, \dots, b_k, a_k)$ returns True because if not, the element b_{k+1} is defined to be distinct with other elements under $\sim_E$. We set it to be $b_{k+1} := a_{k+1}$, and thus, the algorithm CheckSolutions$(b_0, \dots, b_k, b_{k+1})$ returns True. □

We move on to constructing the predicates $P_0, P_1, \dots, P_{n-2}$, where $P_i : ([2^n])^{i+2} \to \{0,1\}$ for each $i = 0, 1, \dots, n-2$. Our structure is straightforward: After applying our sub-procedures, we apply the predictions defined by Pasarkar, Papadimitriou, and Yannakakis [PPY23]. For the self-containment, we now describe the definition of a sequence of finite sets $B_0, \dots, B_i, \dots$ and $F_0, \dots, F_i, \dots$. We proceed with an inductive definition.

In the base case, we define $B_0 := [2^n] \setminus \{v^*\}$. Since $a_0 := v^*$ and $C(v^*) \not\sim_E v^*$, the unfilled set $\mathrm{unfill}_C(B_0 \mid a_0)$ has size $2^n - 2$. We define $F_0 := B_0[\kappa]$, where $\kappa = 2^{n-1} - 1$ if $C(a_0) > 2^{n-1} - 1$; otherwise $\kappa = 2^{n-1}$. Then, the finite set F_0 has size $2^{n-1} - 1$.

Suppose that we have a sequence of elements $a_0, \dots, a_k$ such that $a_0 = v^*$, and B_i and F_i are defined for every $i < k$. Here, we denote by $b_0, \dots, b_k$ the sequence of elements that are outputs of $\beta_{k-1}(a_0, \dots, a_k)$. We first define the finite set B_k using the following rules:

(I) If B_{k-1} is a singleton, then $B_k = F_k = B_{k-1}$.

(II) Otherwise, if $C(b_k)$ belongs to F_{k-1}, then $B_k = F_{k-1}$. If $C(b_k)$ is not in F_{k-1}, then $B_k = B_{k-1} \setminus F_{k-1}$.

Finally, we define F_k. Let κ denote the smallest integer such that

$$|\mathrm{unfill}_C(B_k \mid b_0, \dots, b_k)[\kappa]| = \left\lceil \frac{|\mathrm{unfill}_C(B_k \mid b_0, \dots, b_k)|}{2} \right\rceil.$$

We define $F_k = B_k[\kappa]$.

From the above construction, we can see that $B_0 \supseteq B_1 \supseteq \cdots \supseteq B_i \supseteq \cdots \supseteq B_n$ and $F_i \subseteq B_i$ for every $i \in \{0, 1, \dots, n-1\}$.

To complete constructing the Constrained Long Choice instance, we define the predicate function P_k for each index k in $\{0, 1, \dots, n-2\}$ as follows:

$$P_k(a_0, \dots, a_k, x) = \begin{cases} 1 & \text{if } C(b_{k+1}) \in F_k \\ 0 & \text{if } C(b_{k+1}) \notin F_k, \end{cases}$$

where $(b_0, \dots, b_{k+1}) = \beta_k(a_0, \dots, a_k, x)$.

We now obtain the reduced Constrained Long Choice instance $\langle P_0, P_1, \ldots, P_{n-2}, v^* \rangle$. What remains is to prove that a feasible sequence for our instance allows us to recover a solution to the original Quotient Pigeon instance $\langle C, E, v^* \rangle$.

Let $a_0, a_1, \ldots, a_n$ be a feasible sequence; that is, it holds that $a_0 = v^*$, $a_i \neq a_j$ for all $0 \leq i < j \leq n$, and for each index $i \in \{0, \ldots, n-2\}$, $P_i(a_0, \ldots, a_i, a_j)$ are the same for every $j > i$. Let $b_0, \ldots, b_{n-1}, b_n$ denote the elements that are outputs of the sub-procedure $\beta_{n-1}(a_0, \ldots, a_{n-1}, a_n)$. It suffices to show that $\mathrm{CheckSolutions}(b_0, \ldots, b_{n-1}, b_n)$ returns True.

For the sake of contradiction, we suppose that $\mathrm{CheckSolutions}(b_0, \ldots, b_{n-1}, b_n)$ returns False. From Proposition 18, it satisfies that $b_i \not\sim_E b_j$ for all $0 \leq i < j \leq n$. That is, the elements $b_0, \ldots, b_n, b_{n+1}$ are distinct. Also, we have no element x such that $C(x) \neq v^*$ from our assumption. Furthermore, the following two properties hold.

Proposition 19 (Pasarkar, Papadimitriou, and Yannakakis [PPY23])**.** *(1) For each $i \in \{0, \ldots, n\}$, $C(b_j) \in B_i$ for every $j > i$. (2) For every $i \in \{0, \ldots, n-2\}$, $|\mathrm{unfill}_C(B_i \mid b_0, \ldots, b_i)| = 2^{n-i} - 2$.*

From Item (2) of Proposition 19, we have that $|\mathrm{unfill}_C(B_{n-2} \mid b_0, \ldots, b_{n-2})| = 2$. Recall the definition of the subset F_{n-2}. It satisfies that $|\mathrm{unfill}_C(F_{n-2} \mid b_0, \ldots, b_{n-2})| = 1$ and $|\mathrm{unfill}_C(B_{n-2} \setminus F_{n-2} \mid b_0, \ldots, b_{n-2})| = 1$. Since the $P_{n-2}(a_0, \ldots, a_{n-2}, a_{n-1})$ and $P_{n-2}(a_0, \ldots, a_{n-2}, a_n)$ are the same, exactly one of the following holds: (i) $C(b_{n-1})$ and $C(b_n)$ are in F_{n-2}; and (ii) $C(b_{n-1})$ and $C(b_n)$ are in $B_{n-2} \setminus F_{n-2}$. Therefore, we have a collision, which contradicts from $\mathrm{CheckSolutions}(b_0, \ldots, b_{n-1}, b_n)$ returns False.

4 Conclusion and Open Questions

This short paper has investigated the computational aspects of Interactive Bipartition Stone-Picking Game. We have shown the PLS-hardness of Long Choice, a TFNP formulation of Interactive Bipartition Stone-Picking Game. Our result implies that the complexity class PLC also contains the class PLS. Furthermore, we have introduced the new TFNP problem Quotient Pigeon that is PPP- and PLS-hard as a by-product.

This short paper has left the following open questions:

(i) Does PLC contain PPA? Thus, does PLC unify traditional TFNP subclasses?

(ii) Is the problem Quotient Pigeon PLC-hard? In other words, is Quotient Pigeon PLC-complete?

- As a matter of course, a natural PLC-complete problem is still unknown.
- We are also interested in the relationship between Quotient Pigeon and the problem Unary Long Choice, a TFNP formulation of the non-interactive variant of Interactive Bipartition Stone-Picking Game [PPY23, Section 2].

Finally, we remark that the concept of our TFNP problem Quotient Pigeon has come from [Ish21] (and also [HG18]). Ishizuka [Ish21] has shown the robustness of End of Potential Line. To prove this, he has constructed a reduction by regarding several nodes of the original instance as one node of the reduced instance[4]. Such an approach can be viewed as a quotient from the mathematical perspective. This short paper formulates a search problem over a quotient set by extending their ideas. The primal purpose of this work has been to characterize the complexity of the variants with super-polynomially many known sources of End of Line and End of Potential Line. It is still open whether such variants are also PPAD- and EOPL-complete, respectively. We believe that a search problem on a quotient set helps us advance our understanding of TFNP problems. For example, a computational problem related to the Chevalley-Warning theorem [Göö+20] is one of TFNP problems on quotient sets.

[4]Previously, Hollender and Goldberg [HG18] have proven the robustness of End of Line using the similar approach.

Acknowledgment

This work was partially supported by JST, ACT-X, Grant Number JPMJAX2101. I would like to appreciate Pavel Hubácek for taking the time to talk about TFNP problems privately between September and October 2023. During the discussion, I identified the future direction of my study; before that, I had been uncertain about the direction of my study.

References

[Bur06] Joshua Buresh-Oppenheim. "On the TFNP complexity of factoring." In: (2006). URL: https://www.cs.toro

[CDT09] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. "Settling the complexity of computing two-player Nash equilibria." In: *J. ACM* 56.3 (2009), 14:1–14:57. DOI: 10.1145/1516512.1516516.

[DFM22] Argyrios Deligkas, John Fearnley, and Themistoklis Melissourgos. "Pizza Sharing Is PPA-Hard." In: *Thirty-Sixth AAAI Conference on Artificial Intelligence*. AAAI Press, 2022, pp. 4957–4965. DOI: 10.1609/AAAI.V36I5.20426.

[DGP09] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. "The complexity of computing a Nash equilibrium." In: *Commun. ACM* 52.2 (2009), pp. 89–97. DOI: 10.1145/1461928.1461951.

[DP11] Constantinos Daskalakis and Christos H. Papadimitriou. "Continuous Local Search." In: *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2011, pp. 790–804. DOI: 10.1137/1.9781611973082.62.

[Fea+20] John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. "Unique end of potential line." In: *Journal of Computer and System Sciences* 114 (2020), pp. 1–35. DOI: 10.1016/J.JCSS.2020.05.007.

[FG18] Aris Filos-Ratsikas and Paul W. Goldberg. "Consensus halving is PPA-complete." In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 2018, pp. 51–64. DOI: 10.1145/3188745.3188880.

[GHH23] Paul W. Goldberg, Kasper Høgh, and Alexandros Hollender. "The Frontier of Intractability for EFX with Two Agents." In: *Algorithmic Game Theory - 16th International Symposium*. Vol. 14238. Lecture Notes in Computer Science. Springer, 2023, pp. 290–307. DOI: 10.1007/978-3-031-43254-5_17.

[Göö+20] Mika Göös, Pritish Kamath, Katerina Sotiraki, and Manolis Zampetakis. "On the Complexity of Modulo-q Arguments and the Chevalley - Warning Theorem." In: *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*. Vol. 169. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 19:1–19:42. DOI: 10.4230/LIPICS.CCC.2020.19.

[Göö+22] Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. "Further Collapses in TFNP." In: *37th Computational Complexity Conference*. Vol. 234. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 33:1–33:15. DOI: 10.4230/LIPICS.CCC.2022.33.

[GP18] Paul W. Goldberg and Christos H. Papadimitriou. "Towards a unified complexity theory of total functions." In: *Journal of Computer and System Sciences* 94 (2018), pp. 167–192. DOI: 10.1016/J.JCSS.2017.12.003.

[HG18] Alexandros Hollender and Paul W. Goldberg. "The Complexity of Multi-source Variants of the End-of-Line Problem, and the Concise Mutilated Chessboard." In: *Electron. Colloquium Comput. Complex.* TR18-120 (2018). ECCC: TR18-120. URL: https://eccc.weizmann.ac.il/report/2018/12

[HV21] Pavel Hubácek and Jan Václavek. "On Search Complexity of Discrete Logarithm." In: *46th International Symposium on Mathematical Foundations of Computer Science*. Vol. 202. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 60:1–60:16. DOI: 10.4230/LIPICS.MFCS.2021.60.

[Ish21] Takashi Ishizuka. "The complexity of the parity argument with potential." In: *Journal of Computer and System Sciences* 120 (2021), pp. 14–41. DOI: 10.1016/J.JCSS.2021.03.004.

[Jer16] Emil Jerábek. "Integer factoring and modular square roots." In: *Journal of Computer and System Sciences* 82.2 (2016), pp. 380–394. DOI: 10.1016/J.JCSS.2015.08.001.

[JPY88] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. "How Easy is Local Search?" In: *Journal of Computer and System Sciences* 37.1 (1988), pp. 79–100. DOI: 10.1016/0022-0000(88)90046-3.

[MP91] Nimrod Megiddo and Christos H. Papadimitriou. "On Total Functions, Existence Theorems and Computational Complexity." In: *Theoretical Computer Science* 81.2 (1991), pp. 317–324. DOI: 10.1016/0304-3975(91)90200-L.

[Pap94] Christos H. Papadimitriou. "On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence." In: *Journal of Computer and System Sciences* 48.3 (1994), pp. 498–532. DOI: 10.1016/S0022-0000(05)80063-7.

[PPY23] Amol Pasarkar, Christos H. Papadimitriou, and Mihalis Yannakakis. "Extremal Combinatorics, Iterated Pigeonhole Arguments and Generalizations of PPP." In: *14th Innovations in Theoretical Computer Science Conference*. Vol. 251. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, 88:1–88:20. DOI: 10.4230/LIPICS.ITCS.2023.88.

[Pud15] Pavel Pudlák. "On the complexity of finding falsifying assignments for Herbrand disjunctions." In: *Arch. Math. Log.* 54.7-8 (2015), pp. 769–783. DOI: 10.1007/s00153-015-0439-6.

[SZZ18] Katerina Sotiraki, Manolis Zampetakis, and Giorgos Zirdelis. "PPP-Completeness with Connections to Cryptography." In: *59th IEEE Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 2018, pp. 148–158. DOI: 10.1109/FOCS.2018.00023.

A Procedures

Algorithm 1 The sub-procedure β_k for $k \in \{0, 1, \dots, n\}$

```
Input: a sequence $(a_0, \dots, a_k, a_{k+1})$ on $[2^n]$ such that $a_i \neq a_j$ for all $0 \leq i < j \leq k+1$
Output: a sequence $(b_0, \dots, b_k, b_{k+1})$ on $[2^n]$
 1: $b_0 \leftarrow a_0$
 2: Suppose that we have a sequence $(b_0, \dots, b_k)$ by using the sub-procedures $\beta_0, \dots, \beta_{k-1}$ inductively.
 3: if Exists a pair $b_i, b_j$ such that $b_i \sim_E b_j$ then
 4:     $b_{k+1} \leftarrow a_{k+1}$
 5: else if $a_{k+1} \not\sim_E b_i$ for every $i \in \{0, 1, \dots, k\}$ then
 6:     $b_{k+1} \leftarrow a_{k+1}$
 7: else if CheckSolutions$(b_0, \dots, b_k, a_{k+1})$ returns TRUE then
 8:     $b_{k+1} \leftarrow a_{k+1}$
 9: else
10:     Find the smallest positive integer $\ell$ such that $C^\ell(a_0) \not\sim_E b_i$ for every $i \in \{0, 1, \dots, k\}$
11:     $b_{k+1} \leftarrow C^\ell(a_0)$
12: end if
```

Algorithm 2 The algorithm CheckSolutions that decides whether a solution to QUOTIENT PIGEON exists

```
Input: a sequence of elements $\xi_0, \dots, \xi_k, \xi_{k+1}$ in $[2^n]$
Output: Ether TRUE or FALSE
 1: if There exist two elements $\xi_i$ and $\xi_j$ such that $\xi_i \sim_E \xi_j$ and $C(\xi_i) \sim_E C(\xi_j)$ then
 2:     return TRUE
 3: else if There is an element $\xi \in \{\xi_0, \dots, \xi_k, \xi_{k+1}\}$ such that $C(\xi) \sim_E v^*$ then
 4:     return TRUE
 5: else if There are two elements $\xi, \eta \in \{\xi_0, \dots, \xi_k, \xi_{k+1}\}$ such that $\xi \sim_E \eta$ and $C(\xi) \not\sim_E C(\eta)$ then
 6:     return TRUE
 7: else if There exists an element $\xi \in \{\xi_0, \dots, \xi_k, \xi_{k+1}\}$ such that $E(\xi, \xi) = 0$ then
 8:     return TRUE
 9: else if There are two elements $\xi, \eta \in \{\xi_0, \dots, \xi_k, \xi_{k+1}\}$ such that $E(\xi, \eta) = E(\eta, \xi)$ then
10:     return TRUE
11: else if There exist distinct three elements $\xi, \eta, \zeta \in \{\xi_0, \dots, \xi_k, \xi_{k+1}\}$ such that $\xi \sim_E \eta$, $\eta \sim_E \zeta$, and
    $\xi \not\sim_E \zeta$ then
12:     return TRUE
13: else
14:     return FALSE
15: end if
```