Journal of Computer and System Sciences 120 (2021) 14–41

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

The complexity of the parity argument with potential

Takashi Ishizuka

Graduate School of Mathematics, Kyushu University, Japan

ARTICLE INFO

Article history:
Received 27 October 2019
Received in revised form 18 February 2021
Accepted 14 March 2021
Available online 23 March 2021

Keywords:
Computational complexity
Search problems
PPA
PLS
EOPL

ABSTRACT

The parity argument principle states that every finite graph has an even number of odd degree vertices. We consider the problem whose totality is guaranteed by the parity argument on a graph with potential. In this paper, we show that the problem of finding an unknown odd-degree vertex or a local optimum vertex on a graph with potential is polynomially equivalent to EndOfPotentialLine if the maximum degree is at most three. However, even if the maximum degree is 4, such a problem is $\mathtt{PPA} \cap \mathtt{PLS}$-complete. To show the complexity of this problem, we provide new results on multiple-source variants of EndOfPotentialLine, which is the canonical problem for EOPL. This result extends the work by Goldberg and Hollender; they studied similar variants of EndOfLine.

1. Introduction

We are interested in the complexity class TFNP: a set of total search problems belonging to FNP [21]. Every problem in TFNP satisfies the following two properties: every instance always has a solution, and we can verify that a solution is correct in polynomial time. The class TFNP contains some of the most fundamental, elegant, and intriguing computational problems. The most famous examples are factoring, computing a Nash equilibrium, and finding a local optimum. These problems have no known polynomial-time algorithms, but it seems to be not NP-hard either. In fact, if TFNP has an NP-hard problem, then we have $\mathtt{NP} = \mathtt{coNP}$ [21].

Unfortunately, TFNP is a *"semantic"* class. Generally, such classes seem to have no complete problem like language classes RP, ZPP, and BPP [22]. Thus, we study *"syntactic"* subclasses of TFNP. The best-well known such subclasses are PLS, PPP, PPA, and PPAD [22,16,17]. These are classes of search problems whose totality is guaranteed by the corresponding mathematical lemma. The subclasses of TFNP have been studied productively and extensively, and have been shown to have many interesting search problems as complete problems. Section 2.3 gives details of some complexity classes contained in TFNP.

The above subclasses of TFNP are usually defined in terms of the corresponding search problem on an exponentially-large graph, which is represented by functions. Note that a graph in a problem is presented via a function that computes the neighbors of a given vertex on the graph,[1] not an adjacency matrix. Hence, we need exponential effort to get the structure of the entire graph.

For example, the class PPA is defined as a set of problems that reduce to Leaf: given a neighborhood function on an undirected graph on 2^n vertices whose degree at most two and a known leaf, find an unknown leaf. It is known that PPA has

E-mail address: ishizuka.takashi.664@s.kyushu-u.ac.jp.

[1] If a given graph is a digraph, it is represented via two functions, a function that computes outgoing arcs and a function that computes incoming arcs. We call the former a successor function/circuit and the latter a predecessor function/circuit.

https://doi.org/10.1016/j.jcss.2021.03.004
0022-0000/© 2021 Elsevier Inc. All rights reserved.

the following complete problems: ConsensusHalving, NecklaceSplitting, DiscreteHamSandwich, and OctahedralTucker [13,14,9,1].

Similarly, the class PPAD is defined as a set of problems that reduce to EndOfLine: given a successor circuit and a predecessor circuit on a digraph on 2^n vertices whose in-degree/out-degree at most one and a standard source, find a sink or a non-standard source. PPAD includes a search problems that seems easier than PPA. The most famous PPAD-complete problem is the problem of finding a Nash equilibrium [6,22].

Fearnley et al. [12] studied the search problems on digraphs with potential and produced the new computational complexity class EOPL. Although the natural EOPL-complete problem is still unknown, it contains some fascinating search problems, e.g., solving a Linear Complementarity Problem for P-matrices, solving parity games, and solving simple stochastic games [12].

A natural question worth considering is how easier the problem is by adding a potential condition to a problem known as a known PPA-complete problem. We consider the hardness of finding an unknown odd-degree vertex or a local optimum vertex when we are given a known odd-degree vertex on the graph with potential. To show the complexity of such a problem, we provide the new results on multiple-source variants of EndOfPotentialLine, which is a canonical problem for EOPL.

Recently, Goldberg and Hollender [15] have studied the robustness of the classification by EndOfLine. They considered combinatorial principles related to PPAD, leading to the following problems, on digraphs with degree at most two:

- given k sources and $l \neq k$ sinks, find another sink or source;
- given k sources and $l < k$ sinks, find $k - l$ other sinks; and
- given k sources, find k sinks or k other sources.

They proved that these above problems are also PPAD-complete. Moreover, they showed that the problem Imbalance, in which given a digraph and an unbalanced vertex, i.e., a vertex with in-degree $\neq$ out-degree, find another unbalanced vertex, is also PPAD-complete. These facts imply that the classification by EndOfLine is very robust.

Hollender and Goldberg [18] left an open question: Is a multiple-source variant of EndOfPotentialLine also EOPL-complete? We resolve this question in this paper, and we show that the classification of the class EOPL based on EndOfPotentialLine is robust.

In this paper, we extend Goldberg and Hollenders' [15] results to EndOfPotentialLine. In this problem, given a successor circuit, a predecessor circuit, a potential function, and one standard source, the objective is to find one of a sink, a non-standard source, and a non-increasing arc. We first consider combinatorial principles related to EOPL, leading to the following problems, on graphs with potential with degree at most two:

- given k sources, find another degree-1 vertex or a non-increasing arc; and
- given k sources, find k distinct vertices that are at least one of a sink, other source, and a non-increasing arc.

We show that these variants of EndOfPotentialLine can be classified in terms of the original problem, that is, these problems are also EOPL-complete. Furthermore, we consider the problem of generalizing EndOfPotentialLine to higher degree digraphs. In this problem, given a digraph with potential and one unbalanced vertex, the objective is to find another unbalanced vertex or a non-increasing arc. Naturally, this problem also belongs to EOPL.

Finally, we consider a problem: the goal is to seek an unknown odd-degree vertex or a local optimum vertex on a given graph with potential on 2^n vertices with one known odd-degree vertex. We prove that if the maximum degree of a given graph is at most three, this problem is EOPL-complete. However, this problem is not always EOPL-complete, even if the maximum degree of a given graph is 4.

1.1. *Our contribution*

An overview of our results and the known relationship of complexity classes is depicted in Fig. 1. In this figure, each arrow $\alpha \to \beta$ denotes that there is a polynomial-time reduction from α to β.

In Section 2.4, we introduce the notion called the normalization to simplify arguments in this paper. Some search problems require that every instance has several properties. The best-known example of such a problem is Brouwer. In this problem, the function given by the instance is required Lipschitz continuous. Generally, it seems hard to decide whether a given function is Lipschitz continuous. However, given a witness, it is easy to check it. We often add every witness as a solution called a violation. Informally speaking, our normalization is to transform from an instance which has violations to another instance satisfying the required conditions. The formal definition is given in Section 2.4.

Directed graphs with potential. In Section 3, we extend the elegant results by Goldberg and Hollender [15] to EndOfPotentialLine, and we show the robustness of EOPL. We introduce the new variant of EndOfPotentialLine. We call this problem Multiple-Source EndOfPotentialLine. We prove that this problem is also EOPL-complete. Furthermore, we introduce the new problem generalizing EndOfPotentialLine to higher degree digraphs, which is called PotentialImbalance. We also prove that this problem is EOPL-complete.

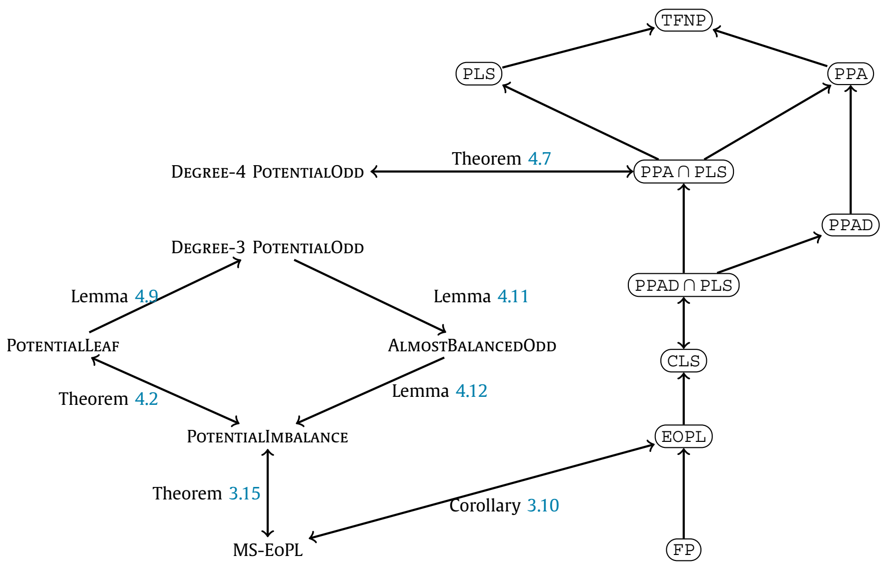

Fig. 1. The relationship of search problems and complexity classes.

Undirected graphs with potential. In Section 4, to study the hardness of the parity argument with potential, we introduce the complexity class PPA ∩ PLS. This class consists of all search problems belonging to both PPA and PLS (see Section 2.3.6). We define PotentialOdd; this problem is a generalization of PotentialImbalance to an undirected graph with potential. We show that PotentialOdd is, generally, PPA ∩ PLS-complete. Specifically, if the maximum degree on a given graph is at most three, then PotentialOdd belongs to EOPL, that is, this problem is EOPL-complete. However, even if the maximum degree on a given degree is 4, then PotentialOdd is PPA ∩ PLS-complete.

1.2. Related work

Daskalakis and Papadimitriou [7] introduced the new complexity class CLS. This class is defined as a set of all search problems that reduce to ContinuousLocalOPT[2] in polynomial time. They showed that this class includes some interesting search problems: finding an approximate fixed point of a contraction map, finding a stationary point of a polynomial, and finding a mixed Nash equilibrium on congestion games [7]. In particular, the problem of finding a pure Nash equilibrium on congestion games is PLS-complete [10]. Daskalakis et al. [8] proved that the problem of finding an approximate fixed point of a metric contraction map is CLS-complete. This is the first natural CLS-complete problem. Most recently, Fearnley et al. [11] showed that CLS = PPAD ∩ PLS. They proved that finding a KKT point is CLS-complete. Babichenko and Rubinstein [2] showed that finding a mixed Nash equilibrium on explicit congestion games is PPAD ∩ PLS-complete; this implies that CCLS = PPAD ∩ PLS, where the class CCLS is another subclass of PPAD ∩ PLS introduced by Daskalakis and Papadimitriou [7]. Therefore, three classes PPAD ∩ PLS, CLS, and CCLS are equivalent.

Hubáček and Yogev [19] introduced the new search problem called EndOfMeteredLine, and showed that this problem belongs to CLS. Fearnley et al. [12] defined the class EOPL by using the problem EndOfPotentialLine. Furthermore, they showed that EndOfPotentialLine is equivalent to EndOfMeteredLine in polynomial time. That is, EOPL is a subclass of CLS.

Beame et al. [3] defined two types of search problems, which are a directed analog of Odd. One is called Excess, and the other is called Imbalance. Specifically, Excess is a generalization of a Sink[3] to higher degree digraphs. The problem Excess is defined as: given a digraph and a vertex satisfying that in-degree $<$ out-degree, find a vertex satisfying that in-degree $>$ out-degree on the given graph. We can easily see that this problem is PPADS-complete. On the other hand, Imbalance is a generalization of EndOfLine to higher degree digraphs. Obviously, this problem is PPAD-hard; however, the PPAD-completeness of Imbalance has been open for two decades. Goldberg and Hollender [15] proved that Imbalance is PPAD-complete. Furthermore, they applied the PPAD-completeness of Multiple-Source EndOfLine to show that k-D-HairyBall is PPAD-complete for all even $k \geq 2$.

[2] In ContinuousLocalOPT, when we are given two arithmetic circuits computing functions $f : [0,1]^3 \to [0,1]^3$ and $p : [0,1]^3 \to [0,1]$ and two constants $\varepsilon, \lambda > 0$, the objective is to find a point $x \in [0,1]^3$ such that $p(f(x)) \geq p(x) - \varepsilon$ or two points on $[0,1]^3$ that violate the λ-Lipschitz continuity of either f or p.

[3] In Sink, when we are given a digraph whose vertex has in-degree and out-degree at most one, and a source, the objective is to find a sink on the given graph.

Daskalakis et al. [6] proved the PPAD-completeness for finding a Nash equilibrium on graphical games, which is called polymatrix games, and three-players forms games. Daskalakis et al. [4] showed that we can compute a Nash equilibrium on the zero-sum polymatrix game by using a liner programming, that is, we can find it in polynomial time. Chen et al. [5] showed that finding a Nash equilibrium in a two-players game is PPAD-hard.

2. Preliminaries

2.1. Notation

For each positive integer n, we define $[n] := \{1, 2, \dots, n\}$. For each finite set X, we denote by $|X|$ the number of elements contained in X. We denote by Σ the finite set of symbols. Each finite sequence of symbols in Σ is called a string. That is, for each string s with respect to Σ, there exists a positive integer n such that $s = s_1 s_2 \dots s_n$ where $s_i \in \Sigma$ for all $i \in [n]$. Here, we define by Σ^* a set of all finite strings, and define by Σ^n a set of all strings of length n. Throughout this paper, let $\Sigma = \{0, 1\}$. The Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. For each pair of strings x, y in Σ^n, we denote by $\|x - y\|$ the Hamming distance between x and y. A language is defined as a subset of Σ^*, and a relation is defined as a subset of $\Sigma^* \times \Sigma^*$.

An instance of a search problems typically contains a few functions. Specifically, a problem on a graph is presented via a function that computes the neighbors of a given vertex. Here, we consider mostly the white-box model; this is an explicit computational model in which each function $f : \Sigma^n \to \Sigma^m$ is given explicitly by a Boolean circuit with n inputs and m outputs. Throughout this paper, we assume that each function given by an instance is represented by a Boolean circuit, and each Boolean circuit is encoded by a binary representation.

2.2. Implicit graphs

We consider a graph produced by some Boolean circuits that compute the neighborhood of a given vertex. In an implicit graph, we can effortlessly obtain only local property around each given vertex, and it takes probably exponential effort to obtain the whole structure of the graph.

We consider mostly graphs with potential in this paper. Each vertex in such a graph has a non-negative value as a potential. A function computing the potential of a given vertex is called a potential function.

Directed graphs. We consider the given two functions $S, P : \Sigma^n \to \Sigma^{dn}$, where d and n are positive integers. Here, for each vertex x in Σ^n, we view as $S(x) = \{y_1, \dots, y_d\}$, where each string y_i is in Σ^n, similarly P do. Thus, two functions S and P output the set of at most d vertices.[4] Then we consider the implicit digraph $G(S, P) = (\Sigma^n, E)$. Each element of Σ^n is called a vertex, and the set Σ^n is called a vertex set. The functions S and P are called a successor function and a predecessor function, respectively. For each pair of vertices (x, y) in $\Sigma^n \times \Sigma^n$, (x, y) is called a valid arc if it holds that $y \in S(x)$ and $x \in P(y)$. Furthermore, a vertex x in Σ^n is called a self-loop if $S(x) = \{x\}$ and $P(x) = \{x\}$. The set E consists of all valid arcs with respect to S and P, i.e., we define $E := \{(x, y) \in \Sigma^n \times \Sigma^n; x \neq y,\ \ y \in S(x),\ \text{and } x \in P(y)\}$. For each vertex v in Σ^n, an arc (u, v) in E is called an incoming arc of v, and an arc (v, w) in E is called an outgoing arc of v. Furthermore, we denote by $E^-(v)$ the set of all incoming arcs of v, and denote by $E^+(v)$ the set of all outgoing arcs of v. For every vertex v in Σ^n, we define the in-degree of v as $\delta^-(v) = |E^-(v)|$, the out-degree of v as $\delta^+(v) = |E^+(v)|$, and the degree of v as $\delta(v) = \delta^-(v) + \delta^+(v)$. Finally, we define the degree of the implicit graph $G(S, P)$ as $\deg(G) = \max_{v \in \Sigma^n} \delta(v)$.

In particular, when $d = 1$, the implicit graph $G(S, P)$ is a directed graph with in-degree/out-degree at most one. Then every valid arc (x, y) in E satisfies that $S(x) = y$ and $P(y) = x$. Furthermore, a vertex s in Σ^n is called a source if it satisfies that $S(P(s)) \neq s$, and a vertex t in Σ^n is called a sink if it satisfies that $P(S(t)) \neq t$.

Undirected graphs. When we are given a function $N : \Sigma^n \to \Sigma^{dn}$, we consider the implicit graph $G(N) = (\Sigma^n, E)$, where d and n are positive integers. For each string x in Σ^n, we view as $N(x) = \{y_1, \dots, y_d\}$, where each string y_i is contained in Σ^n. The function N is called the neighborhood function, and the set Σ^n is called the vertex set. A set $\{x, y\} \subseteq \Sigma^n$ satisfying that $x \neq y$ is called a valid edge if it holds that $y \in N(x)$ and $x \in N(y)$. A vertex x in Σ^n is called an isolated vertex if it holds that $N(x) \subseteq \{x\}$. Then the set E consists of all valid edges with respect to N, i.e., we define $E := \{\{x, y\} \subseteq \Sigma^n; x \neq y, y \in N(x),\ \text{and } x \in N(y)\}$. For each vertex v in Σ^n, the degree of x is defined as $\deg_G(x) = |\{y \in \Sigma^n; \{x, y\} \in E\}|$. Furthermore, the degree of the graph G is defined as $\deg(G) = \max_{v \in \Sigma^n} \deg(v)$.

In particular, when $d = 2$, the implicit graph $G(N)$ is an undirected graph with degree at most two. Then the vertex x with degree one is called a leaf. It is well known that the total number of all leaves on $G(N)$ is always even [22].

2.3. Search problem

A computational search problem is defined by using a relation. Let $R \subseteq \Sigma^* \times \Sigma^*$ be a relation. R is said to be polynomial-time computable if we can decide whether $(x, y) \in R$ in polynomial time for each pair of strings $(x, y) \in \Sigma^* \times \Sigma^*$. We call

[4] We suppose that when a vertex has degree strictly less than d, two functions S and P output the remaining points as itself without loss of generality.

that R is a polynomial-balanced relation if there exists a non-negative integer k such that for each pair of strings $(x, y) \in R$, $|y| \leq |x|^k$. Throughout this paper, we assume that every relation is polynomial-time computable and a polynomial-balanced relation. We define a search problem with respect to a relation R as follows [22].

Definition 2.1 *(A search problem with respect to R).* When we are given a string x in Σ^n, we return a string y in Σ^* such that (x, y) is in R if such a y exists, and return a string "*no*" otherwise.

Here, to simplify notation, we refer to a search problem with respect to R as a problem R. The class of all search problems is called the class $\texttt{FNP}$. We denote by $\texttt{FP}$ the subclass of $\texttt{FNP}$ that contains all search problems that can be solved in polynomial time. The class $\texttt{TFNP}$ is the subclass of $\texttt{FNP}$ containing all search problems which have totality. We say that a search problem R has totality if there always exists a string $y \in \Sigma^*$ such that $(x, y) \in R$ for every instance $x \in \Sigma^*$. It is clear that $\texttt{FP} \subseteq \texttt{TFNP} \subseteq \texttt{FNP}$ [22]. However, it is still open whether these inclusions are strict. Megiddo and Papadimitriou [21] showed that there is an $\texttt{FNP}$-complete problem in $\texttt{TFNP}$ if and only if $\texttt{NP} = \texttt{coNP}$.

A polynomial-time reduction from problem A to problem B is a polynomial-time computable function f which maps an instance x of A to an instance $f(x)$ of B, plus another polynomial-time computable function g which maps every solution y of $f(x)$ to a solution $g(y, x)$ of x.

Generally, every search problem has a solution set $\mathcal{O}(\mathcal{I})$ for each instance $\mathcal{I}$. In this paper, we consider that every problem has two types of solutions (possibly empty). One is called a regular solution, and the other is called violating a solution. Some search problems require their instance to satisfy some conditions. While it is easy to decide whether a given instance is valid, it is hard to check whether a given instance satisfies the required conditions most of the time. However, it is effortless to verify whether a given instance violates the required conditions if we obtain a witness. We often add a witness as a solution, and such a solution is called a violating solution. On the other hand, a regular solution means a desired solution to the search problem originally. For instance, ContinuousLocalOPT (see [7]) requires that two functions f and p are λ-Lipschitz continuous. It seems to be hard that we efficiently verify whether f and p are λ-Lipschitz continuous. Therefore, we treat the violation as a solution. Note that this configuration ensures the totality of ContinuousLocalOPT.

Formally, when we are given an instance $\mathcal{I}$ of the search problem R that has violating solutions, the solution set $\mathcal{O}(\mathcal{I})$ to $\mathcal{I}$ is denoted as follows.

$$\mathcal{O}(\mathcal{I}) = \mathcal{O}_R(\mathcal{I}) \cup \mathcal{O}_V(\mathcal{I}),$$

where $\mathcal{O}_R(\mathcal{I})$ is a set of all regular solutions and $\mathcal{O}_V(\mathcal{I})$ is a set of all violating solutions. For instance, the set of solutions for ContinuousLocalOPT consists of all ε-approximate local optima and all witnesses of λ-Lipschitz continuity for given functions. The set $\mathcal{O}_R(\cdot)$ contains all ε-approximate local optima, and the set $\mathcal{O}_V(\cdot)$ contains all witnesses of non-λ-Lipschitz continuity for given functions.

The following subsections introduce several subclasses of $\texttt{TFNP}$. See Fig. 1 for their relationship.

2.3.1. Class PLS

The complexity class $\texttt{PLS}$ is the class of all search problems whose totality is proved by local search algorithms. This class is introduced by Johnson et al. [20]. Formally, this class is defined as the set of all search problems that are reducible to LocalOPT in polynomial time.

Definition 2.2 *(*LocalOPT*).* A valid instance of LocalOPT consists of two positive integers n and m, two Boolean circuits $f : \Sigma^n \to \Sigma^n$ and $p : \Sigma^n \to \{0, 1, \dots, 2^m - 1\}$. This problem is defined as: given a valid instance (n, m, f, p), find a string x in Σ^n such that $p(x) \geq p(f(x))$.

Definition 2.3 *(Class $\texttt{PLS}$).* The complexity class $\texttt{PLS}$ consists of all search problems that are reducible to LocalOPT in polynomial time.

It is well known that the class $\texttt{PLS}$ contains many essential and fascinating search problems. In particular, the following two problems, MinFlip and MaxFlip, are the first $\texttt{PLS}$-complete problems [20].

Definition 2.4 *(*MinFlip*).* A valid instance of MinFlip consists of two positive integers n and m and a Boolean circuit C with n inputs and m outputs that computes a function $f : \Sigma^n \to \{0, 1, \dots, 2^m - 1\}$. This problem is defined as: given a valid instance (n, m, C), find a string x in Σ^n such that for every $y \in \Sigma^n$ with $\|x - y\| = 1$, $C(x) \leq C(y)$.

Definition 2.5 *(*MaxFlip*).* A valid instance of MaxFlip consists of two positive integers n and m and Boolean circuit C with n inputs and m outputs that computes a function $f : \Sigma^n \to \{0, 1, \dots, 2^m - 1\}$. Then this problem is defined as: given a valid instance (n, m, C), find a string x in Σ^n such that for every $y \in \Sigma^n$ with $\|x - y\| = 1$, $C(x) \geq C(y)$.

Theorem 2.6 *(Johnson et al. [20]). MinFlip and MaxFlip are $\texttt{PLS}$-complete.*

2.3.2. Class PPA

Papadimitriou [22] introduced the complexity class PPA. This class consists of all search problems whose totality is guaranteed by the parity argument. Formally, we define the class PPA as the set of all search problems that are reducible to Leaf in polynomial time.

Definition 2.7 *(*Leaf*).* A valid instance of Leaf consists of a positive integer n, an implicit graph $G(N) = (\Sigma^n, E)$, and a known leaf $\pi \in \Sigma^n$ on $G(N)$, where the implicit graph $G(N)$ is produced by a Boolean circuit $N : \Sigma^n \to \Sigma^{2n}$. This problem is defined as: given a valid instance (n, N, π), find another leaf x on the implicit graph $G(N)$, i.e., it satisfies that $\deg_G(x) = 1$ and $x \neq \pi$.

Definition 2.8 *(Class* PPA*).* The complexity class PPA consists of all search problems that are reducible to Leaf in polynomial time.

Let $G(N)$ be an implicit graph given by a valid instance of Leaf. Notice that every vertex on the graph $G(N)$ has degree at most two. The parity argument guarantees that there must exist an unknown leaf when we have a known leaf. This principle also works for higher degree graphs, that is, a graph which has a known odd-degree vertex must have at least one unknown odd-degree vertex. We consider the search problem Odd, which is a natural generalization of Leaf. Furthermore, Papadimitriou [22] showed that Odd is also PPA-complete.

Definition 2.9 *(*Odd*).* A valid instance of Odd consists of two positive integers n and d, an implicit graph $G(N) = (\Sigma^n, E)$, and an odd-degree vertex π on $G(N)$, where the implicit graph $G(N)$ is produced by a Boolean circuit $N : \Sigma^n \to \Sigma^{dn}$. This problem is defined as: given a valid instance (n, d, N, π), find another odd-degree vertex on the implicit graph $G(N)$, i.e., it satisfies that $x \neq \pi$ and $\deg_G(x) = 2k - 1$ for some $k \in \mathbb{N}$.

Theorem 2.10 *(Papadimitriou [22]).* Odd *is* PPA*-complete.*

2.3.3. Class PPAD

The complexity class PPAD is the class of all search problems whose totality is guaranteed by the parity argument for digraphs. This class is also introduced by Papadimitriou [22].

Definition 2.11 *(*EndOfLine*).* A valid instance of EndOfLine consists of a positive integer n, two Boolean circuits $S, P : \Sigma^n \to \Sigma^n$, a standard source $\pi \in \Sigma^n$, i.e., $P(\pi) = \pi \neq S(\pi)$. This problem is defined as: given a valid instance (n, S, P, π), find a vertex x in Σ^n satisfying at least one of the following:

(R1) $P(S(x)) \neq x$, i.e., x is a sink, and
(R2) $S(P(x)) \neq x \neq \pi$, i.e., x is a non-standard source.

Definition 2.12 *(Class* PPAD*).* The complexity class PPAD consists of all search problems that can be reduced to EndOfLine in polynomial time.

By definition, it is easy to see that every instance of EndOfLine has at least one sink. This implies that the class PPAD is a subclass of TFNP. Furthermore, it is not hard to see that PPAD is a subclass of PPA.

For each valid instance of EndOfLine, we are given a digraph $G(S, P)$ with in-degree/out-degree at most one and a standard source $\pi \in \Sigma^n$, i.e., $S(P(\pi)) \neq \pi$. Then, the parity argument for digraph states that there must exist at least one sink on $G(S, P)$. This principle also works for higher degree digraphs, that is, an implicit digraph which has an unbalanced vertex, i.e., a vertex with in-degree $\neq$ out-degree, must have another unbalanced vertex. Such a problem is called Imbalance, and Goldberg and Hollender [15] showed that Imbalance is polynomially equivalent to EndOfLine. Thus, Imbalance is PPAD-complete.

Definition 2.13 *(*Imbalance*).* A valid instance of Imbalance consists of two positive integers n and d, an implicit digraph $G(S, P)$, and an unbalanced vertex π in Σ^n such that $\delta^+(\pi) \neq \delta^-(\pi)$, where the implicit digraph $G(S, P)$ is produced by a successor circuit $S : \Sigma^n \to \Sigma^{dn}$ and a predecessor circuit $P : \Sigma^n \to \Sigma^{dn}$. This problem is defined as: given a valid instance (n, d, S, P, π), find another unbalanced vertex x on the implicit digraph $G(S, P)$, i.e., it satisfies that $\delta^+(x) \neq \delta^-(x)$ and $x \neq \pi$.

Theorem 2.14 *(Goldberg and Hollender [15]).* Imbalance *is* PPAD*-complete.*

2.3.4. Class EOPL

The complexity class EOPL is introduced by Fearnley et al. [12]. This class consists of all search problems that are reducible to EndOfPotentialLine.

Definition 2.15 *(*EndOfPotentialLine*).* A valid instance of EndOfPotentialLine consists of two positive integers n and m, two Boolean circuits $S, P : \Sigma^n \to \Sigma^n$, one Boolean circuit $V : \Sigma^n \to \{0, 1, \dots, 2^m - 1\}$, and a standard source π in Σ^n such that $P(x) = x \neq S(x)$ and $V(\pi) = 0$. This problem is defined as: given a valid instance (n, m, S, P, V, π), find a vertex x in Σ^n satisfying at least one of the following:

(R1) $P(S(x)) \neq x$, i.e., x is a sink,
(R2) $S(P(x)) \neq x \neq \pi$, i.e., x is a non-standard source, and
(V1) $S(x) \neq x$, $P(S(x)) = x$, and $V(S(x)) - V(x) \leq 0$.

Definition 2.16 *(Class* EOPL*).* The complexity class EOPL consists of all search problems that can be reduced to EndOfPotentialLine in polynomial time.

Notice that EndOfPotentialLine has violating solutions. Let (n, m, S, P, V, π) be a valid instance of EndOfPotentialLine, and let $G(S, P) = (\Sigma^n, E)$ be the implicit digraph produced by S and P. Each vertex x on $G(S, P)$ has a potential assigned by the potential function V. Then this problem requires that for every valid arc (x, y) in E, it holds that $V(x) < V(y)$, i.e., every valid arc on $G(S, P)$ is an increasing arc. Therefore, any non-increasing valid arc is a violation.

2.3.5. The problem: EITHER SOLUTION($\mathcal{A}, \mathcal{B}$)

In this section, we describe the problem EitherSolution($\mathcal{A}, \mathcal{B}$) to formulate the complexity class PPA ∩ PLS (see Section 2.3.6 for details). This search problem is introduced by Daskalakis and Papadimitriou [7] to consider the complexity of the class PPAD ∩ PLS.[5] Informally speaking, in this problem, given two instances that one is an instance of $\mathcal{A}$, and the other is an instance of $\mathcal{B}$, then find a solution for either $\mathcal{A}$ or $\mathcal{B}$.

Definition 2.17 *(*EitherSolution($\mathcal{A}, \mathcal{B}$)*).* Fix two different search problems $\mathcal{A}$ and $\mathcal{B}$. A valid instance of EitherSolution($\mathcal{A}$, $\mathcal{B}$) consists of a valid instance $I_{\mathcal{A}}$ of $\mathcal{A}$ and a valid instance $I_{\mathcal{B}}$ of $\mathcal{B}$. Then this problem is defined as: given a valid instance $(I_{\mathcal{A}}, I_{\mathcal{B}})$, find either a solution to $I_{\mathcal{A}}$ or a solution to $I_{\mathcal{B}}$.

Daskalakis and Papadimitriou [7] showed that if $\mathcal{A}$ is a PPAD-complete problem and $\mathcal{B}$ is a PLS-complete problem, then the problem EitherSolution($\mathcal{A}, \mathcal{B}$) is PPAD ∩ PLS-complete, and thus, they proved the following theorem.

Theorem 2.18 *(Daskalakis and Papadimitriou [7]).* EitherSolution(EndOfLine, LocalOPT) *is a* PPAD ∩ PLS*-complete problem.*

Now, we present a more general statement than the result of Daskalakis and Papadimitriou [7] mentioned above. Specifically, we show that EitherSolution($\mathcal{A}, \mathcal{B}$) is an $\mathcal{C}_A \cap \mathcal{C}_B$-complete problem if A is $\mathcal{C}_A$-complete, and B is $\mathcal{C}_B$-complete.

Theorem 2.19. *Let $\mathcal{C}_A$ and $\mathcal{C}_B$ be subclasses of* FNP *which have complete problems.* EitherSolution(A, B) *is $\mathcal{C}_A \cap \mathcal{C}_B$-complete if A is a $\mathcal{C}_A$-complete problem, and B is a $\mathcal{C}_B$-complete problem.*

Proof. First, we show that EitherSolution(A, B) belonging to both $\mathcal{C}_A$ and $\mathcal{C}_B$. We assume that we are given a valid instance $\mathcal{I}_A$ of A and a valid instance $\mathcal{I}_B$ of B. Immediately, the problem A belongs to $\mathcal{C}_A$; we can obtain a polynomial-time reduction to A. Similarly, we can construct a polynomial-time reduction to B. Therefore, EitherSolution(A, B) is contained in $\mathcal{C}_A$ and $\mathcal{C}_B$.

To show the hardness, we consider any problem C belonging to $\mathcal{C}_A \cap \mathcal{C}_B$. There is a polynomial-time reduction f from C to A since A is a $\mathcal{C}_A$-complete problem. Thus, we can generate a valid instance $f(x)$ of A from a given instance of C. Similarly, there is a polynomial-time reduction g from C to B, that is, we can produce a valid instance $g(x)$ of B from a given instance of C. Hence, the tuple $(f(x), g(x))$ is a valid instance of EitherSolution(A, B), that is, we have a polynomial-time reduction from C to EitherSolution(A, B). □

From Theorem 2.19, the following statement straightforwardly follows.

Lemma 2.20. EitherSolution(MaxFlip, MinFlip) *is* PLS*-complete.*

[5] The complexity class PPAD ∩ PLS is defined as a set of all search problems belonging to both PPAD and PLS.

In the problem EITHERSOLUTION(MAXFLIP, MINFLIP), when we are given two Boolean circuits C and D, we find either a local maximum of C or a local minimum of D. A natural question worth considering is the complexity when $C = D$. That is, we consider the problem, which is called FLIP, of finding a local maximum or a local minimum for a given Boolean circuit C. Furthermore, we show that FLIP is also PLS-complete. We formally define FLIP as follows.

Definition 2.21 *(*FLIP*).* Given a Boolean circuit C with n inputs and m outputs computing a function $f : \Sigma^n \to \{0, 1, \ldots, 2^m - 1\}$, find a bit-string $x \in \Sigma^n$ satisfying at least one of the following:

(R1) $C(x) \geq C(y)$ for every $y \in \Sigma^n$ with $\|x - y\| = 1$, and
(R2) $C(x) \leq C(y)$ for every $y \in \Sigma^n$ with $\|x - y\| = 1$.

Theorem 2.22. FLIP *is a* PLS*-complete problem.*

Proof. Assume that two Boolean circuits C and D as an instance of EITHERSOLUTION(MAXFLIP, MINFLIP), where C and D compute functions $f : \Sigma^{n_1} \to \{0, 1, \ldots, 2^{m_1} - 1\}$ and $g : \Sigma^{n_2} \to \{0, 1, \ldots, 2^{m_2} - 1\}$, respectively. Furthermore, the Boolean circuit C with n_1 inputs and m_1 outputs is an instance of MAXFLIP, and the Boolean circuit D with n_2 inputs and m_2 outputs is an instance of MINFLIP.

Now, we define the new Boolean circuit $E : \Sigma^{n_1+n_2} \to \Sigma^{m_1+m_2}$ as follows. We consider the first n_1-bits of input x in $\Sigma^{n_1+n_2}$ to E as input to C, and the remaining n_2-bits as input to D. Furthermore, we define the first n_1-bits as x_1 and the remaining n_2-bits as x_2, and thus, denote x as (x_1, x_2). Similarly, we denote y in $\Sigma^{m_1+m_2}$ as (y_1, y_2), where y_1 is the first m_1-bits of y and y_2 is the remaining m_2-bits of y. Then for every bit-string $x = (x_1, x_2)$ in $\Sigma^{n_1+n_2}$, we define as $E(x_1, x_2) = (C(x_1), D(x_2))$. In particular, the circuit E computes $2^{m_2} f(x_1) + g(x_2)$ for every bit-string $x = (x_1, x_2)$ in $\Sigma^{n_1+n_2}$. It is easy to see that E is constructed in polynomial time. The Boolean circuit E is a valid instance of FLIP.

What remains is to show that we can obtain an original solution from a solution for E. In the rest of this proof, we denote by $N(x_1, x_2) = N_1(x_1, x_2) \cup N_2(x_1, x_2)$ the neighborhood of $x = (x_1, x_2) \in \Sigma^{n_1+n_2}$, where $N_1(x_1, x_2) = \{(y_1, x_2) \in \Sigma^{n_1+n_2}; \|x_1 - y_1\| = 1\}$ and $N_2(x_1, x_2) = \{(x_1, y_2) \in \Sigma^{n_1+n_2}; \|x_2 - y_2\| = 1\}$. Here, the function N_1 outputs the neighbors of a given bit-string x_1 in Σ^{n_1}, and the function N_2 outputs the neighbors of a given bit-string x_2 in Σ^{n_2}. Notice that for every bit-string $x = (x_1, x_2)$ in $\Sigma^{n_1+n_2}$, the neighbors of x agree with the union of $N_1(x_1)$ and $N_2(x_2)$.

First, we consider when we obtain a local maximum solution $(x_1, x_2) \in \Sigma^{m_1+m_2}$ for E, i.e., it satisfies that $E(x_1, x_2) \geq E(y_1, y_2)$ for every $(y_1, y_2) \in N(x_1, x_2)$. Then x_1 is a local maximum solution for C because it holds that $C(x_1) \geq C(y_1)$ for every $y_1 \in N(x_1, x_2)$. Second, we consider when we obtain a local minimum solution $(x_1, x_2) \in \Sigma^{m_1+m_2}$ for E, i.e., it satisfies that $E(x_1, x_2) \leq E(y_1, y_2)$ for every $(y_1, y_2) \in N(x_1, x_2)$. Then x_2 is a local minimum solution for D because it holds that $D(x_2) \leq D(y_2)$ for every $y_2 \in N(x_1, x_2)$. Hence, we can extract a solution of the original instance from a solution of the new instance.

From the above arguments, we complete the polynomial-time reduction from EITHERSOLUTION(MAXFLIP, MINFLIP) to FLIP. Therefore, the theorem follows. □

2.3.6. Class PPA ∩ PLS

In this section, we introduce the computational complexity class PPA ∩ PLS. This class is defined as the set of all search problems that are contained in PPA and PLS.

Definition 2.23 *(Class* PPA ∩ PLS*).* The complexity class PPA ∩ PLS consists of all search problems that belongs to both PPA and PLS.

Clearly, this class has following relationship with other well-known classes:

(a) EOPL ⊆ PPAD ∩ PLS ⊆ PPA ∩ PLS.
(b) PPA ∩ PLS ⊆ PPA.
(c) PPA ∩ PLS ⊆ PLS.
(d) PPAD ⊆ PPA ∩ PLS if and only if PPAD ⊆ PLS.

Furthermore, the next theorem straightforwardly follows from Theorem 2.19.

Theorem 2.24. EITHERSOLUTION(ODD, FLIP) *is* PPA ∩ PLS*-complete.*

2.4. Normalization

In this section, we introduce a normalization of search problems. Informally speaking, a normalization of a search problem that has both regular solutions and violating solutions is defined as a transformation from a given valid instance to another valid instance that satisfies the required conditions. Formally, it is defined as follows.

Definition 2.25 *(Normalization).* A search problem R has both regular solutions and violating solutions. A normalization of the problem R is a polynomial-time computable function f which maps an instance $\mathcal{I}$ of R to another instance $f(\mathcal{I})$ of R which has no violations, i.e., $\mathcal{O}_V(f(\mathcal{I}))$ is empty, plus another polynomial-time computable function g which maps every solution y of $f(\mathcal{I})$ to a solution $g(y,\mathcal{I})$ of $\mathcal{I}$.

The rest of this section gives us an example of a normalizable search problem. We show that EndOfPotentialLine can be normalized.

Proposition 2.26. EndOfPotentialLine *can be normalized.*

Proof. Let $\mathcal{I} = (n, m, S, P, V, \pi)$ be a valid instance of EndOfPotentialLine. Without loss of generality, we assume that the standard source π is not a solution of $\mathcal{I}$ because we can check it in polynomial time. Now, we construct another instance $\mathcal{J} = (n, m, S', P', V', \pi)$ that has no violations.

First, we construct a successor circuit $\bar{S}$ and a predecessor circuit $\bar{P}$ such that they always compute valid arcs or self-loops. For each vertex x in Σ^n, we define

$$\bar{S}(x) := \begin{cases} S(x) & \text{if } S(x) \neq x \text{ and } P(S(x)) = x, \\ x & \text{otherwise,} \end{cases}$$

and

$$\bar{P}(x) := \begin{cases} P(x) & \text{if } P(x) \neq x \text{ and } S(P(x)) = x, \\ x & \text{otherwise.} \end{cases}$$

From the above construction, if $y = \bar{S}(x) \neq x$, then $\bar{P}(y) = x$ for each $x \in \Sigma^n$, similarly, if $y = \bar{P}(x) \neq x$, then $\bar{S}(y) = x$ for each $x \in \Sigma^n$.

Next, we construct a successor circuit S' and a predecessor circuit P'. For each vertex x in Σ^n, we define

$$S'(x) := \begin{cases} \bar{S}(x) & \text{if } \bar{S}(x) \neq x \text{ and } V(\bar{S}(x)) > V(x), \\ x & \text{otherwise,} \end{cases}$$

and

$$P'(x) := \begin{cases} \bar{P}(x) & \text{if } \bar{P}(x) \neq x \text{ and } V(\bar{P}(x)) < V(x), \\ x & \text{otherwise.} \end{cases}$$

By the definitions of S' and P', if $S'(x) \neq x$, then $S'(x) = \bar{S}(x) \neq x$ and $V(\bar{S}(x)) > V(x)$. This implies that it satisfies that $\bar{P}(\bar{S}(x)) = x$. Since $V(S'(x)) = V(\bar{S}(x)) > V(x)$, it holds that $P'(S'(x)) = P'(\bar{S}(x)) = \bar{P}(\bar{S}(x)) = x$. Similarly, if $P'(x) \neq x$, then it holds that $S'(P'(x)) = x$ and $V(x) > V(\bar{P}(x)) = V(P'(x))$. Therefore, every valid arc with respect to S' and P' is always a strictly increasing arc.

Finally, we define the new potential function V' as follows.

$$V'(x) := \begin{cases} 0 & \text{if } S'(x) = x = P'(x), \\ V(x) & \text{otherwise.} \end{cases}$$

From the above constructions, it is easy to see that the standard source π in Σ^n satisfies that $P'(\pi) = \pi \neq S'(x)$ and $V'(\pi) = 0$. Therefore, the tuple $\mathcal{J} = (n, m, S', P', V', \pi)$ is a valid instance of EndOfPotentialLine. Furthermore, it is not hard to see that $\mathcal{J}$ has no violating solutions. In the rest of this proof, we show that when we obtain a solution of $\mathcal{J}$, we can convert to a solution of $\mathcal{I}$ in polynomial time.

We first consider when we obtain a solution x such that $P'(S'(x)) \neq x$. Then it must satisfy that $S'(x) = x$. Therefore, it holds that $P'(S'(x)) = P'(x) \neq x$. If $\bar{S}(x) \neq x$, then $V(\bar{S}(x)) \leq V(x)$ by the definition of S'. Hence, x is a violating solution for the original instance since $V(\bar{S}(x)) = V(S(x))$. On the other hand, if $\bar{S}(x) = x$, then $\bar{P}(\bar{S}(x)) \neq x$ since $P'(x) \neq x$. This implies that x satisfies $P(S(x)) \neq x$, and thus, x is a sink of the original instance.

Next, we consider when we obtain a solution x such that $S'(P'(x)) \neq x \neq \pi$. Then it must satisfy that $P'(x) = x$. Therefore, it holds that $S'(P'(x)) = S'(x) \neq x$. If $\bar{P}(x) \neq x$, then $V(x) \leq V(\bar{P}(x))$. Therefore, it satisfies that $S(P(x)) \neq P(x)$, $P(S(P(x))) = P(x)$, and $V(x) \leq V(P(x))$ since $\bar{P}(x) = P(x) \neq x$. This implies that the vertex $\bar{P}(x)$ is a violating solution for the original instance. On the other hand, if $\bar{P}(x) = x$, then $\bar{S}(\bar{P}(x)) \neq x$. Therefore, it holds that $S(P(x)) \neq x \neq \pi$. This implies that x is a non-standard source of the original instance.

From the above arguments, we complete the normalization of EndOfPotentialLine. □

By definition, any normalization is polynomial-time computable. Without loss of generality, we assume that every given instance is already normalized if it can be normalized.

Applying Proposition 2.26, without loss of generality, we assume that every instance of EndOfPotentialLine satisfies the following properties:

- if $S(x) \neq x$, then $P(S(x)) = x$ and $V(S(x)) > V(x)$;
- if $P(x) \neq x$, then $S(P(x)) = x$ and $V(x) > V(P(x))$.

3. Multiple source problems

In this section, we discuss the new variants of EndOfPotentialLine. The most typical modification worth considering is perhaps the following: what if the implicit digraph associated EndOfPotentialLine has two or more standard sources instead of one. The objective remains the same: find one of a sink, a non-standard source, and a non-increasing arc. The existence of at least two standard sources implies that there must exist at least two sinks. Such a problem has more candidate solutions than the original problem. Hence, a new problem might seem more effortless.

As mentioned in the introduction, Goldberg and Hollender [15] studied and showed that the multiple-source variants of EndOfLine also belong to PPAD. Surprisingly, these problems are PPAD-complete. Thus, the classification by EndOfLine is robust. The natural question worth considering is whether a similar statement holds for EndOfPotentialLine. Throughout this section, we prove that this claim is mostly correct.

3.1. Given k sources, then find one solution

We consider the following variant of EndOfPotentialLine that the goal is to find one solution when we are given k standard sources, where k is some constant.

Definition 3.1 *(kS-EndOfPotentialLine).* Let k be a positive integer. A valid instance of kS-EndOfPotentialLine, abbreviated kS-EoPL, consists of two positive integers n and m, two Boolean circuits $S, P : \Sigma^n \to \Sigma^n$, one Boolean circuit $V : \Sigma^n \to \{0, 1, \dots, 2^m - 1\}$, and a set of k standard sources $\Pi = \{\pi_1, \dots, \pi_k\}$ such that $P(\pi) = \pi \neq S(\pi)$ and $V(\pi) = 0$ for each π in Π. This problem is define as: given a valid instance (n, m, S, P, V, Π), find a vertex x in Σ^n satisfying at least one of the following:

(R1) $P(S(x)) \neq x$,
(R2) $S(P(x)) \neq x \notin \Pi$, and
(V1) $S(x) \neq x$, $P(S(x)) = x$, and $V(S(x)) - V(x) \leq 0$.

Immediately, we can see that kS-EoPL is EOPL-hard. For every given instance of EndOfPotentialLine, we create k copies it. Therefore, the following lemma straightforwardly follows.

Lemma 3.2. *kS-EoPL is EOPL-hard.*

Next, we prove that kS-EoPL belongs to EOPL. To prove this, we construct a polynomial-time reduction from kS-EoPL to EndOfPotentialLine. Our reduction is via an instance of EndOfMeteredLine as an intermediate instance. The problem EndOfMeteredLine is introduced by Hubáček and Yogev [19]. Every EndOfMeteredLine instance has an excellent property: the potential increase exactly one along each valid arc. In other words, the potential on a vertex means the distance from some source. This property plays a crucial role in our reduction.

To construct a polynomial-time reduction from kS-EoPL to EndOfPotentialLine, we have to settle the following two issues:

(a) the bundling of k standard sources into one standard source;
(b) computing potential of each point efficiently.

In the multiple-source variant of EndOfLine, we only had to resolve the issue (a). Goldberg and Hollender [15] resolved this problem and showed the containment of Multiple-Source EndOfLine in PPAD. They bundled k standard sources into one source by regarding a set of some vertices as one vertex. Recall that the digraph given by an instance has an exponential number of vertices; it seems that we can not efficiently determine whether vertices that make up a new vertex are on the same path. Furthermore, not all paths have the same length. For example, look at the path on the left of Fig. 2; the length is different for each path. To settle this problem, Goldberg and Hollender [15] simulated k paths by retracing the paths some times backward. However, their exciting technique did not establish that we can efficiently compute the potential because the turn back the path became an obstacle.

The straightforward way to settle the above two issues (a) and (b) is to make all paths the same length and ensure each vertex that makes up a new vertex is always at the same distance from its source. It is possible in the kS-EoPL because the

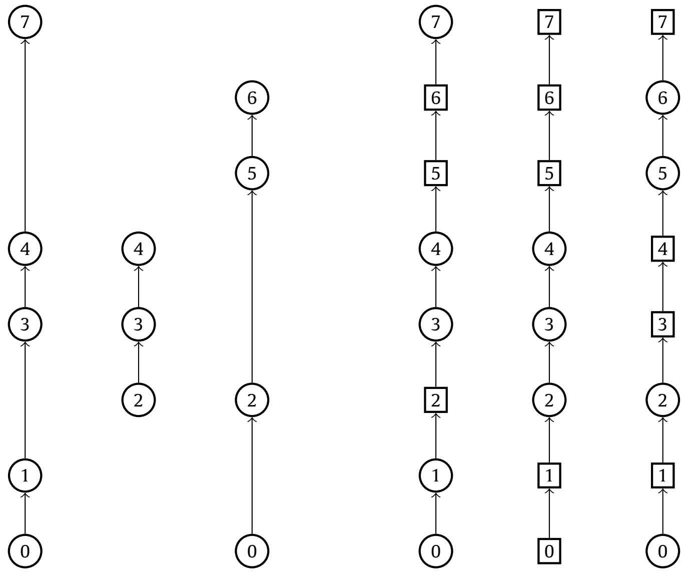

Fig. 2. The left digraph is the original instance. The right digraph is a metered line based on the left digraph. A vertex of a rectangle represents a dummy. Note that each source and each sink on the right digraph has the same potential 0 and 7, respectively.

instance has a potential function. Recall that every instance of EndOfMeteredLine has an excellent property: the potential increase exactly one along each path. Fearnley et al. [12] gave us a polynomial-time reduction from EndOfPotentialLine to EndOfMeteredLine. We realize a polynomial-time reduction from kS-EoPL to EndOfPotentialLine via an EndOfMetered-Line instance. Moreover, we add some additional vertices to make the length of all paths the same.

The outline of our reduction as follows. This reduction consists of two steps. In the first step, we construct an intermediate instance like EndOfMeteredLine, see Fig. 2. In the second step, we create a valid instance of EndOfPotentialLine from the intermediate instance. We regard a list of k distinct vertices as a single vertex. Specifically, all k vertices that make up a valid vertex on the EndOfPotentialLine instance have the same potential. This configuration allows us to compute the potential of the new vertex efficiently. The new standard source is the set of all k standard sources of the intermediate instance. This completes the illustration of our reduction from kS-EoPL to EndOfPotentialLine. We can efficiently extract a solution from a solution to the new instance. Details are given in the proof of Lemma 3.3.

Lemma 3.3. *kS-EoPL is reducible to* EndOfPotentialLine *in polynomial time.*

Proof. Let $\mathcal{I} = (n, m, S, P, V, \Pi)$ be a valid instance of kS-EoPL. Applying Proposition 2.26, we assume that this instance is already normalized without loss of generality, that is, $\mathcal{I}$ satisfies the following conditions:

- if $S(x) \neq x$ then $P(S(x)) = x$ and $V(S(x)) > V(x)$;
- if $P(x) \neq x$ then $S(P(x)) = x$ and $V(x) > V(P(x))$.

In this proof, we denote the set of k distinct standard sources as $\Pi = \{\pi_1, \pi_2, \dots, \pi_k\}$.

First, we construct an intermediate instance $\mathcal{I}_M = (2n + h, h, S', P', V', \Pi')$, where $h = 2n + m$. Let $\mathcal{U} := \Sigma^n \times \Sigma^n \times \{0, 1, \dots, 2^h - 1\}$ be the new vertex set. The intermediate instance $\mathcal{I}_M$ is also an valid instance of kS-EoPL satisfying the following properties:

- if $S'(u) \neq u$, then $P'(S'(u)) = x$ and $V'(S'(u)) = V'(u) + 1$;
- if $P'(u) \neq u$, then $S'(P'(u)) = x$, and $V'(u) = V'(P'(u)) + 1$;
- every self-loop vertex u, i.e., $S'(u) = u = P'(u)$ has $V'(u) = 0$;

where u is a vertex in $\mathcal{U}$. Furthermore, we can efficiently convert a solution to $\mathcal{I}$ from a solution to $\mathcal{I}_M$.

We show the algorithms of the successor circuit $S' : \mathcal{U} \to \mathcal{U}$ and the predecessor circuit $P' : \mathcal{U} \to \mathcal{U}$, in Algorithm 1, and Algorithm 2, respectively. Our algorithm is based on the reduction from EndOfPotentialLine to EndOfMeteredLine by Fearnley et al. [12]. Here, we add additional processing to make the length of all paths the same.

Finally, for each vertex $(x, y, z) \in \mathcal{U}$, we define

$$V'(u) := \begin{cases} 0 & \text{if } S'(u) = (u) = P'(u), \\ z & \text{otherwise.} \end{cases}$$

Algorithm 1 Compute $S'(x, y, z)$.

```
1: If $x = y$ then
    1. if $S(x) \neq x$ and $P(S(x)) = x$ then
        (a) if $V(S(x)) - 1 > V(x) = z$ then return $(x, S(x), z + 1)$,
        (b) if $V(S(x)) - 1 = V(x) = z$ then return $(S(x), S(x), z + 1)$,
    2. if $S(P(x)) \neq x$, i.e., $x$ is a source, then
        (a) if $z < V(x)$ then return $(x, x, z + 1)$.
    3. if $P(S(x)) \neq x$, i.e., $x$ is a sink, then
        (a) if $2^h - 1 > z \geq V(x)$ then return $(x, x, z + 1)$.
2: If $x \neq y$ then
    1. if $S(x) = y$, $P(y) = x$, and $V(x) < V(y)$ then
        (a) if $V(x) + 1 \leq z < V(y) - 1$ then return $(x, y, z + 1)$,
        (b) if $V(x) < z = V(y) - 1$ then return $(y, y, z + 1)$.
3: In all other cases, then return $(x, y, z)$.
```

Algorithm 2 Compute $P'(x, y, z)$.

```
1: If $x = y$ then
    1. if $P(y) \neq y$ and $S(P(y)) = y$ then
        (a) if $V(P(y)) + 1 < V(y) = z$ then return $(P(y), y, z - 1)$,
        (b) if $V(P(y)) + 1 = V(y) = z$ then return $(P(y), P(y), z - 1)$,
    2. if $S(P(y)) \neq y$, i.e., $x$ is a source, then
        (a) if $V(y) \geq z > 0$ then return $(y, y, z - 1)$.
    3. if $P(S(y)) \neq y$, i.e., $x$ is a sink, then
        (a) if $V(y) < z$ then return $(y, y, z - 1)$
2: If $x \neq y$, $S(x) = y$, and $P(y) = x$ then
    1. if $V(x) + 1 < z \leq V(y) - 1$ then return $(x, y, z - 1)$,
    2. if $V(x) + 1 = z < V(y)$ then return $(x, x, z - 1)$.
3: In all other cases, then return $(x, y, z)$.
```

It is easy to see that the circuits S', P', and V' are constructed in polynomial time, and they can be polynomial-time computable. Now, we denote the new set of k distinct standard sources by $\Pi' := \{(\pi_i, \pi_i, 0) \in \mathcal{U}; i = 1, 2, \ldots, k\}$. We complete the construction of the intermediate instance $\mathcal{I}_M$.

Before completing the first step, we verify that we can extract a solution to $\mathcal{I}$ from a solution to $\mathcal{I}_M$ in polynomial time. First, we show that each valid arc on S' and P' is increased by exactly one.

Claim 3.4. *Let $u = (x, y, z) \in \mathcal{U}$. If $S'(u) \neq u$, then it satisfies that $P'(S'(u)) = u$ and $V'(S'(u)) = V'(u) + 1$. Similarly, if $P'(u) \neq u$, then it satisfies that $S'(P'(u)) = u$ and $V'(u) = V'(P'(u)) + 1$.*

Proof. It is clear from our construction. If a vertex $u = (x, y, z)$ satisfies that $S'(u) \neq u$, then $V'(u) = z$. Therefore, we have $V'(S'(u)) = z + 1$. Similarly, it holds that $V'(u) = V'(P'(u)) + 1$ when $P'(u) \neq u$. □

Next, we show that if a vertex $u = (x, y, z) \in \mathcal{U}$ is a sink or a source, $x = y$ holds. The next claim states the contraposition of this.

Claim 3.5. *For each vertex $u = (x, y, z) \in \mathcal{U}$, if $x \neq y$ then $S'(P'(u)) = u$ and $P'(S'(u)) = u$.*

Proof. We only consider when $S(x) = y$ and $P(y) = x$ hold because x is a self-loop in all other cases. From Algorithm 2, we have

$$P'(u) = \begin{cases} (x, y, z - 1) & \text{if } V(x) + 1 < z < V(y) - 1, \\ (x, x, z - 1) & \text{if } V(x) + 1 = z < V(y). \end{cases}$$

Hence, $S'(P'(u)) = u$ holds. Similarly, we can see that $P'(S'(u)) = u$ holds if $x \neq y$. Therefore, the claim follows. □

So far, we have shown the properties that the intermediate instance $\mathcal{I}_M$ satisfies. We prove that we can efficiently convert a solution to the original instance $\mathcal{I}$ from a solution to the intermediate instance $\mathcal{I}_M$. First, we show that every sink $u = (x, y, z) \in \mathcal{U}$ on S' and P' includes an original solution; specifically, x is a sink for $\mathcal{I}$. This fact appears in the proof of the next claim. The following claim is useful in the second step; a vertex u is a sink if and only if u has potential $2^h - 1$.

Claim 3.6. *Let $u = (x, y, z) \in \mathcal{U}$. A vertex u is a sink on S' and P', i.e., $P'(S'(u)) \neq u$, if and only if $V'(u) = z = 2^h - 1$.*

Proof. First, we prove that $V'(u) = z = 2^h - 1$ when a vertex u is a sink on S' and P'. From Claim 3.5 and Claim 3.4, it satisfies that $x = y$ and $S'(u) = u \neq P'(u)$. We show that $P(S(x)) \neq x$ holds. For the sake of contradiction, we suppose that

$P(S(x)) = x$. It is clear that $S'(u) = u = P'(u)$ if $S(x) = x$. This is contradiction from $P'(u) \neq u$. Therefore, we have $S(x) \neq x$ and $P(S(x)) = x$. It satisfies that $V(x) \neq z$ since $S'(u) = u$. This implies that $P'(u) = u$, which contradicts. Hence, $P(S(x)) \neq x$ holds, and thus, x is a sink on the original instance. Notice that $P'(u) \neq u$, we have $V(x) < z$, and it satisfies that $z = 2^h - 1$ by $S'(u) = u$.

Conversely, we prove that if $V'(u) = z = 2^h - 1$, then $P'(S'(u)) \neq u$ holds. It is easy to see that at least one of $S'(u) \neq u$ and $P'(u) \neq u$ since $V'(u) = z = 2^h - 1 > 0$. By definition, every $u \in \mathcal{U}$ has $V'(u) < 2^h$. This implies that $S'(u) = u$ and $P'(u) \neq u$ hold. □

Next, we show that we can efficiently take an original solution from each source $u = (x, y, z) \in \mathcal{U}$ on S' and P'. This fact appears in the proof of Claim 3.7. In this claim, we show that if a vertex u is a source, u has potential 0. Conversely, if a vertex u has a potential 0 and it is not self-loop, u is a source. Notice that assuming $u \notin \Pi'$ in Claim 3.7, we obtain a non-standard source on S and P, that is, we get a solution to $\mathcal{I}$ efficiently.

Claim 3.7. *We assume that a vertex $u = (x, y, z) \in \mathcal{U}$ is not a self-loop for S' and P'. The vertex u is a source, i.e., $S'(P'(u)) \neq u$, if and only if $V'(u) = z = 0$.*

Proof. From Claim 3.5, we have $x = y$. Applying Claim 3.4, it satisfies that $P'(u) = u \neq S'(u)$. First, we prove that $S(P(x)) \neq x$ holds. For the contradiction, we suppose that $S(P(x)) = x$. We obviously see that $S'(u) = u$ if $P(x) = x$. This is a contradiction from $S'(u) \neq u$. Therefore, we have $P(x) \neq x$ and $S(P(x)) \neq x$. Since $P'(u) = u$, it satisfies that $V(x) \neq z$, and thus, we have $S'(u) = u$, which contradicts. Hence, $S(P(x)) \neq x$ holds, i.e., x is a source on the original instance. Notice that $S'(u) \neq u$, we have $V(x) > z$, and it satisfies that $z = 0$ by $P'(u) = u$.

Conversely, we prove that if $V'(u) = z = 0$, then $S'(P'(u)) \neq u$. Notice that u is not a self-loop for S' and P', we have at least one of $S'(u) \neq u$ and $P'(u) \neq u$. Furthermore, for every vertex in $\mathcal{U}$, the potential function V' has non-negative value. It must hold that $P'(u) = u$ from Claim 3.4, and thus we have $S'(u) \neq u$. This implies that the vertex u is a source for S' and P'. □

We complete the first step, construction of the intermediate instance $\mathcal{I}_M$. Note that this instance is also a valid instance of kS-EoPL. From now on, we start the second step. In the second step, we construct a polynomial-time reduction from kS-EoPL to EndOfPotentialLine. Specifically, we transform $\mathcal{I}_M$ to the instance $\mathcal{J}$ of EndOfPotentialLine in polynomial time. Note that we can extract an original solution from a solution to $\mathcal{J}$ if we efficiently extract an intermediate solution from a solution to $\mathcal{J}$ since we can convert an original solution from a solution to the intermediate solution in polynomial time.

First, we define the set $\Gamma_k := \bigcup_{\alpha=0}^{2^h-1} \Gamma_k(\alpha)$, where

$$\Gamma_k(\alpha) := \{\{u_1 \ldots, u_k\}; \forall i \neq j, u_i \neq u_j, \forall i, V'(u_i) = \alpha, \text{ and } u_i \in \mathcal{U}\}.$$

It is easy to see that we can decide whether a given string $v = \{u_1, \ldots, u_k\}$ is in Γ_k in polynomial time. Moreover, the bit-length of $v = \{u_1, \ldots, u_k\}$ in Γ_k is bounded by some polynomial in n. We need $(2n + h)$-bits to represent each u_i, so $k(2n + h)$-bits are sufficient to represent v. Notice that k is a constant number, this is a polynomial in n.

For each vertex $v = \{u_1, \ldots, u_k\}$ in Γ_k, we define the successor circuit S_k and the predecessor circuit P_k as follows. If there exists an element u_i such that $S'(u_i) = u_i = P'(u_i)$, then $S_k(v) = v$, otherwise $S_k(v) = \{S'(u_1), \ldots, S'(u_k)\}$. Similarly, if there exists an element u_i such that $S'(u_i) = u_i = P'(u_i)$, then $P_k(v) = v$, otherwise $P_k(v) = \{P'(u_1), \ldots, P'(u_k)\}$.

It satisfies that $S_k(v)$ is also in Γ_k for each vertex $v = \{u_1, \ldots, u_k\}$ in Γ_k. It is clear that when there is an element u_i such that $S'(u_i) = u_i = P'(u_i)$. Therefore, we show when there are no such elements. It suffices to prove that $S'(u_i) \neq S'(u_j)$ for each pair of u_i and u_j with $u_i \neq u_j$. If u_i is a sink on S' and P', i.e., $S'(u_i) = u_i \neq P(u_i)$, then $V'(u_i) = 2^h - 1$. From Claim 3.6, it follows that $V'(u_j) = 2^h - 1$ for every j, and thus, u_j is also a sink. We have that $S'(u_i) = u_i \neq u_j = S'(u_j)$. Next, we show when u_i is not a sink on S' and P'. Note that u_j is also not a sink. Suppose for the sake of contradiction that $S'(u_i) = S'(u_j)$. From Claim 3.4, it satisfies that $P'(S'(u_i)) = u_i$. We get the following $u_i = P'(S'(u_i)) = P'(S'(u_j)) = u_j$, getting a contradiction. Finally, we can easily see that $V'(S'(u_i)) = V'(S'(u_j))$ for each pair of u_i and u_j with $u_i \neq u_j$ from Claim 3.4. Similarly, we can see that $P'(v)$ is also in Γ_k for each vertex $v = \{u_1, \ldots, u_k\}$ in Γ_k.

We define the new potential function V_k as follows.

$$V_k(v) = \begin{cases} 0 & \text{if } S_k(v) = v = P_k(v), \\ V(u_1) & \text{otherwise.} \end{cases}$$

By definition, the above function S_k, P_k, and V_k can be constructed in polynomial time, and they are polynomial-time computable. Finally, the new standard source is defined as $\pi^* = \{(\pi_1, \pi_1, 0), (\pi_2, \pi_2, 0), \ldots, (\pi_k, \pi_k, 0)\} \in \Gamma_k$. It is easy to see that $P_k(\pi^*) = \pi^* \neq S_k(\pi^*)$ and $V_k(\pi^*) = 0$ hold. Therefore, the tuple $\mathcal{J} = (k(2n + h), h, S_k, P_k, V_k, \pi^*)$ is a valid instance of EndOfPotentialLine. In the rest of this proof, we show that when we obtain a solution of $\mathcal{J}$, we can convert to an original solution in polynomial time.

First, we show that there are no violating solutions for $\mathcal{J}$. Suppose that $S_k(v) \neq v = \{u_1, \ldots, u_k\}$. Then v is not a self-loop with respect to S_k and P_k. Furthermore, if there is an element u_i in v such that $S'(u_i) = u_i \neq P'(u_i)$, then it satisfies that $z_i = 2^h - 1$ from Claim 3.6. This implies that $z_j = 2^h - 1$ for every $j \in [k]$. This is a contradiction from $S_k(v) \neq v$ because it satisfies that $S'(u_j) = (u_j)$ for every $j \in [k]$. Therefore, it holds that $S'(u_i) \neq u_i$ for every $i \in [k]$. Therefore, we obtain that $P'(S'(u_i)) = u_i$ by the definitions of S' and P'. From our normalization assumption for $\mathcal{I}$, it satisfies that $V(S(x)) = V(x) + 1$ unless $S(x) = x$. This implies that $V'(S'(x)) > V'(x)$ holds, and thus, there are no violating solutions for $\mathcal{J}$.

Second, we consider that we obtain a solution $v = \{u_1, \ldots, u_k\}$ satisfying that $P_k(S_k(v)) \neq v$. Then v has no elements which are self-loops. Hence, it holds that

$$\{P'(S'(u_1)), \ldots, P'(S'(u_k))\} \neq \{u_1, \ldots, u_k\}.$$

Therefore, there exists at least one element u_i such that $P'(S'(u_i)) \neq u_i$ holds. From Claim 3.6, it must hold that $V'(u_i) = 2^h - 1$, and thus, $V'(u_1) = \cdots = V'(u_k) = 2^h - 1$ since $v \in \Gamma_k$. For every element $u_j = (x_j, y_j, z_j)$ in v, it holds that $P'(S'(u_j)) \neq u_j$. This implies that it satisfies that $P(S(x_j)) \neq x_j$ for every x_j. Therefore, we can extract one solution for the original instance in polynomial time.

Finally, we consider that we obtain a solution $v = \{u_1, \ldots, u_k\}$ satisfying that $S_k(P_k(v)) \neq v \neq \pi^*$. Then v has no elements which are self-loops. Hence, it holds that

$$\{S'(P'(u_1)), \ldots, S'(P'(u_k))\} \neq \{u_1, \ldots, u_k\}.$$

Therefore, there exists at least one element u_i such that $S'(P'(u_i)) \neq (u_i)$ holds. From Claim 3.7, it holds that $V'(u_i) = 0$. Hence, $V'(u_1) = \cdots = V'(u_k) = 0$ since $v \in \Gamma_k$. Note that for every $j \in [k]$, $u_j = (x_j, y_j, z_j)$ is not a self-loop, and thus, this implies that $S'(P'(u_j)) \neq u_j$ holds. Therefore, it satisfies that $S(P(x_j)) \neq x_j$ for every x_j. Since $v \neq \pi^*$, there exists at least one string $x_{j'}$ such that $x_{j'} \notin \Pi$. Such a string $x_{j'}$ corresponds to an original solution. Therefore, we can extract one solution for the original instance in polynomial time.

From the above arguments, we complete the polynomial-time reduction from kS-EoPL to EndOfPotentialLine, and this implies that the problem kS-EoPL is a member of the class EOPL. □

The following theorem follows from Lemma 3.2 and Lemma 3.3.

Theorem 3.8. kS-EoPL *is* EOPL*-complete.*

Note that the proof of Lemma 3.3 works even if k is not constant. Indeed, our proof works for a multiple-source variant that has at most polynomially numbers of sources. We consider the following problem, and we immediately see that this problem is an EOPL-complete problem.

Definition 3.9 *(*Multiple-Source EndOfPotentialLine*).* A valid instance of Multiple-Source EndOfPotentialLine, abbreviated MS-EoPL, consists of a three positive integers n, m, and k, two Boolean circuits $S, P : \Sigma^n \to \Sigma^n$, one Boolean circuit $V : \Sigma^n \to \{0, 1, \ldots, 2^m - 1\}$, and a set of k standard sources $\Pi = \{\pi_1, \ldots, \pi_k\}$ such that $P(\pi) = \pi \neq S(\pi)$ and $V(\pi) = 0$ for each π in Π. This problem is define as: given a valid instance (n, m, k, S, P, V, Π), find a vertex x in Σ^n satisfying at least one of the following:

(R1) $P(S(x)) \neq x$,
(R2) $S(P(x)) \neq x \notin \Pi$, and
(V1) $S(x) \neq x$, $P(S(x)) = x$, and $V(S(x)) - V(x) \leq 0$.

Corollary 3.10. MS-EoPL *is* EOPL*-complete.*

Remark 1. Another variant of EndOfPotentialLine worth considering is finding a non-standard source, a sink, or a non-increasing arc when we are given k standard sources and l standard sinks, where $l < k$. In the case of EndOfLine, it can be easily reduced to $(k - l)$S-EoL in polynomial time [18]. We can add a valid arc from each of l standard sinks to some corresponding known source, and thus, we obtain a valid instance with $(k - l)$ standard sources and no standard sinks. Unfortunately, this simple technique does not work for a variant of EndOfPotentialLine. Recall that every source/sink given by our reduction from kS-EoPL to EndOfPotentialLine contains k sources/sinks. Furthermore, the vertex consisting of k standard sources is a standard source of the new instance. Therefore, we can extract a non-standard source from an obtained non-standard source. Notice that $l < k$, we can extract a non-standard sink from an obtained sink for the new instance. That is, this variant is also EOPL-complete.

3.2. Higher degree problem: IMBALANCE with potential

Up to this point, we have only considered implicit graphs where every vertex has in-degree/out-degree at most one. However, the principle that guarantees the totality of EndOfPotentialLine can be generalized to higher degree implicit graphs. If we are given an implicit digraph with potential and an "unbalanced" vertex, i.e., a vertex with in-degree $\neq$ out-degree, then there must exist another unbalanced vertex or a non-increasing arc.

In this section, we consider the new variant of the problem EndOfPotentialLine that corresponds to a higher degree digraph. We define this problem as follows.

Definition 3.11 *(*PotentialImbalance*).* A valid instance of PotentialImbalance consists of three positive integers n, m, and d, an implicit graph $G(S, P) = (\Sigma^n, E)$, a Boolean circuit $V : \Sigma^n \to \{0, 1, \ldots, 2^m - 1\}$, and an unbalanced vertex $\pi \in \Sigma^n$ such that $\delta^+(\pi) > \delta^-(\pi)$, where the implicit graph $G(S, P, V)$ is produced by a successor circuit $S : \Sigma^n \to \Sigma^{dn}$ and a predecessor circuit $P : \Sigma^n \to \Sigma^{dn}$. This problem is defined as: given a valid instance (n, m, d, S, P, V, π), find a vertex $x \in \Sigma^n$ satisfying at least one of the following:

(R1) $\delta^+(x) \neq \delta^-(x)$ and $x \neq \pi$, and
(V1) there exists a valid arc $(x, y) \in E$ such that $V(y) \leq V(x)$.

By definition, it is not hard to see that the problem PotentialImbalance is EOPL-hard. Surprisingly, this problem also belongs to EOPL, and thus PotentialImbalance is EOPL-complete. To prove this, we show that there is a polynomial-time reduction from PotentialImbalance to MS-EoPL in Lemma 3.12.

Lemma 3.12. PotentialImbalance *is reducible to* MS-EoPL *in polynomial time.*

Proof. Our proof relies on the proof in Goldberg and Hollender [15]. The reduction constructed by this proof is the same as their technique, except for defining the potential function. Indeed, our proof is completed simply by adding the construction of a natural potential function to a polynomial-time reduction from Imbalance to MS-EoPL constructed by Goldberg and Hollender [15].

Let (n, m, d, S, P, V, π) be a valid instance of PotentialImbalance, and let $G(S, P) = (\Sigma^n, E)$ be the implicit digraph produced by S and P, where E is a set of all valid arcs with respect to S and P.

To construct a reduction to MS-EoPL, we apply "chessplayer algorithm" used to prove that Odd reduces to Leaf by Papadimitriou [22]. We fix a label function $\lambda_x^+ : E^+(x) \to [\delta^+(x)]$ and $\lambda_x^- : E^-(x) \to [\delta^-(x)]$ for each vertex $x \in \Sigma^n$. The function $\lambda_x^+(y)$ is the index of y in the successor list of x, ordered in lexicographically. Similarly, the function $\lambda_x^-(z)$ corresponds to the index of z in the predecessor list of x, ordered in lexicographically. Note that both functions λ_x^+ and λ_x^- are bijective and polynomial-time computable. This means that there exists an algorithm computing the label $\lambda_x^+((x, y))$ for a given vertex x in Σ^n and a valid arc (x, y) in $E^+(x)$ in polynomial time. Similarly, there exists an algorithm computing the label $\lambda_x^-((y, x))$ for a given vertex x in Σ^n and a valid arc (y, x) in $E^-(x)$ in polynomial time.

To show the reduction to MS-EoPL, we consider the following vertex set,

$$\mathcal{K} = \bigcup_{x \in \Sigma^n} \{(i, x); i \in [d(x)]\},$$

where $d(x) = \max\{\delta^+(x), \delta^-(x)\}$. Furthermore, we define a successor S' and a predecessor P' over $\mathcal{K}$ as follows.

For each vertex $(i, x) \in \mathcal{K}$, if $i > \delta^+(x)$, then define $S'(i, x) = (i, x)$, else define $S'(i, x) = (j, y)$, where $y \in \Sigma^n$ satisfies that $\lambda_x^+((x, y)) = i$ and the label $j = \lambda_y^-((x, y))$. Since the label function is a bijection, such a string y and a label j are unique for each vertex (i, x).

Similarly, for each vertex $(i, x) \in \mathcal{K}$, if $i > \delta^-(x)$, then $P'(i, x) = (i, x)$, else define $P'(i, x) = (j, y)$, where $y \in \Sigma^n$ satisfies that $\lambda_x^-((y, x)) = i$ and the label $j = \lambda_y^+((y, x))$. Since the label function is a bijection, such a string y and a label j are unique for each (i, x).

It is easy to see that the functions S' and P' can be constructed in polynomial time. Straightforwardly, the following two claims hold.

Claim 3.13. *For each vertex $(i, x) \in \mathcal{K}$, if $S'(i, x) \neq (i, x)$, then $P'(S'(i, x)) = (i, x)$.*

Proof. Since $S'(i, x) \neq (i, x)$, it satisfies that $i \leq \delta^+(x)$. We can uniquely write $S'(i, x) = (j, y)$ since the label function is a bijection. Now, it satisfies that $(\lambda_x^+)^{-1}(i) = (x, y) \in E$ and $j = \lambda_y^-((x, y))$. Therefore, $P'(j, y) = (i, x)$ holds, and thus, it satisfies that $P'(S'(i, x)) = (i, x)$. $\square$

Claim 3.14. *For each vertex $(i, x) \in \mathcal{K}$, if $P'(i, x) \neq (i, x)$, then $S'(P'(i, x)) = (i, x)$.*

Proof. Since $P'(i, x) \neq (i, x)$, it satisfies that $i \leq \delta^-(x)$. We can uniquely write $P'(i, x) = (j, y)$. Now, it satisfies that $(\lambda_x^-)^{-1}(i) = (y, x) \in E$ and $j = \lambda_y^+((y, x))$. Therefore, we obtain that $S'(j, y) = (i, x)$, and thus, this implies that $S'(P'(i, x)) = (i, x)$ holds. $\square$

Furthermore, we define by $\Pi' = \{(i,\pi); \delta^-(\pi) < i \le \delta^+(\pi)\}$ the set of all standard sources. It is easy to see that every element of Π' is a souse. For every vertex (i,x), we define

$$V'(i,x) := \begin{cases} 0 & \text{if } (i,x) \in \Pi', \\ V(x) & \text{otherwise.} \end{cases}$$

From the above arguments, the tuple $(m+n, m, S', P', V', \Pi')$ is a valid instance of MS-EoPL, and these construction can be in polynomial time. In the rest of this proof, we show that when we obtain a solution of $(m+n, m, S', P', V', \Pi')$, we can convert to an original solution in polynomial time.

First, we consider the case that we obtain a solution $(i,x) \in \mathcal{K}$ satisfying that $P'(S'(i,x)) \neq (i,x)$. Then it holds that $S'(i,x) = (i,x)$ from Claim 3.13. Hence, it satisfies that $\delta^+(x) < i$. Since $S'(i,x) = (i,x)$, it holds that $P'(S'(i,x)) = P'(i,x) \neq (i,x)$. Therefore, it satisfies that $i \le \delta^-(x)$, and thus, $\delta^+(x) < \delta^-(x)$ holds. This implies that x is a solution for the original instance.

Second, we consider the case that we obtain a solution $(i,x) \in \mathcal{K}$ satisfying that $S'(P'(i,x)) \neq (i,x) \notin \Pi'$. Then it holds that $\delta^-(x) < i$ from the definition of P'. This implies that $x \neq \pi$ since $(i,x) \notin \Pi'$. Furthermore, it holds that $S'(P'(i,x)) = S'(i,x) \neq (i,x)$ since $P'(i,x) = (i,x)$, and thus, $i \le \delta^+(x)$ holds. Hence, it satisfies that $\delta^-(x) < \delta^+(x)$, and this implies that x is a solution for the original instance.

Finally, we consider the case that we obtain a solution (i,x) satisfying that $S'(i,x) \neq (i,x)$, $P'(S'(i,x)) = (i,x)$, and $V'(S(i,x)) \le V'(i,x)$. Since $S'(i,x) \neq (i,x)$, there is a valid arc $(x,y) \in E$ such that $S'(i,x) = (j,y)$. Furthermore, it holds that $V(y) = V'(S'(i,x)) \le V'(i,x) = V(x)$ by the definition of V'. Hence, x is a solution to the original instance.

We complete the construction of the polynomial-time reduction form PotentialImbalance to MS-EoPL. □

Straightforwardly, we obtain the following theorem from the above lemma.

Theorem 3.15. PotentialImbalance *is* EOPL*-complete.*

Remark 2. In the definition of the problem PotentialImbalance (see Definition 3.11), we assume that the standard unbalanced vertex π satisfies that $\delta^+(\pi) > \delta^-(\pi)$ on the implicit digraph $G(S,P)$ with the potential function V. In fact, the proof of Lemma 3.12 works even if we assume that the standard vertex π satisfies that $\delta^+(\pi) < \delta^-(\pi)$ on the implicit digraph $G(S,P)$ with the potential function V by making a simple modification. Specifically, we define the new potential function as $(2^m - 1) - V(x)$ for each vertex x in Σ^n, and change the direction of every valid arc. Therefore, without loss of generality, we assume that the standard unbalanced vertex π given by an instance of PotentialImbalance satisfies that $\delta^+(\pi) \neq \delta^-(\pi)$.

3.3. *Looking for multiple solutions*

As mentioned in the Introduction, Goldberg and Hollender [15] also proved that the problem, in which given k sources, find k distinct sources or sinks, is also PPAD-complete. Now, we consider a generalized problem of EndOfPotentialLine that similar to this problem. The principle that guarantees the existence of solutions for EndOfPotentialLine implies that if there are k sources, then there must exist at least k distinct sinks. Clearly, this problem is EOPL-hard. In this section, we show that this problem also belongs to EOPL. The new variant of the problem EndOfPotentialLine is defined as follows.

Definition 3.16 *(k-EndOfPotentialLine).* Let k be a positive integer. A valid instance of k-EndOfPotentialLine, abbreviated k-EoPL, consists of two positive integers n and m, two Boolean circuits $S, P : \Sigma^n \to \Sigma^n$, one Boolean circuit $V : \Sigma^n \to \{0, 1, \ldots, 2^m - 1\}$, and a set of k standard sources $\Pi = \{\pi_1, \ldots, \pi_k\}$ such that $P(\pi) = \pi \neq S(\pi)$ and $V(\pi) = 0$ for each π in Π. This problem is define as: given a valid instance (n, m, S, P, V, Π), find distinct k vertices $x_1, \ldots, x_k$ such that for each $i \in [k]$, x_i satisfies at least one of the following:

(R1) $x_i \neq P(S(x_i))$,
(R2) $S(P(x_i)) \neq x_i \notin \Pi$, and
(V1) $S(x_i) \neq x_i$ and $P(S(x_i)) = x_i$, and $V(S(x_i)) - V(x_i) \le 0$.

By definition, it is easy to see that k-EoPL is EOPL-hard. For every given instance of EndOfPotentialLine, we create k copies it.

Lemma 3.17. *k-EoPL is* EOPL*-hard.*

Surprisingly, the complexity is the same as in EOPL; that is, this problem is reducible to EndOfPotentialLine in polynomial time. We show this result in Lemma 3.18. Our technique is an extension of the technique by Hollender and Goldberg

[18] to EndOfPotentialLine. Their technique has a weak point: it possibly can not efficiently compute a potential for each vertex. Our method shown in Lemma 3.3 gets over this weak point, i.e., we can efficiently compute a potential for each vertex.

Lemma 3.18. *k-EoPL is reducible to* EndOfPotentialLine *in polynomial time.*

Proof. Let $\mathcal{I} = (n, m, S, P, V, \Pi)$ be a valid instance k-EoPL. First, we assume that $\mathcal{I}$ is already normalized by applying the method Proposition 2.26. Notice that our method also works for multiple-source variants. Hence, in this proof, we assume that the following properties holds:

- if $S(x) \neq x$ then $P(S(x)) = x$ and $V(S(x)) > V(x)$;
- if $P(x) \neq x$ then $S(P(x)) = x$ and $V(x) > V(P(x))$.

Note that we might be able to extract strictly fewer than k solutions to the original instance when we are given k solutions to the normalized instance. In the normalization method shown in Proposition 2.26, we remove a non-increasing arc, and we make two new solutions; a sink and a non-standard source. If we get these two solutions, we can only extract a single solution to the original instance.

Our reduction from k-EoPL to EndOfPotentialLine constructed in this proof avoids this. A solution to the new instance consists of either k' distinct sinks or k' distinct sources for the normalized instance, where $k' \geq k$. Notice that every solution of the new instance does not mix the sinks and sources of the normalized instance. Therefore, we always extract the k distinct solutions to the original instance.

Let $\mathcal{U} = \Sigma^n \times \Sigma^n \times \{0, 1, \dots, 2^h - 1\}$ and $h = 2n + m$. First, we construct three polynomial-time computable and constructive circuits $S', P' : \mathcal{U} \to \mathcal{U}$ and $V' : \mathcal{U} \to \{0, 1, \dots, 2^h - 1\}$ that are constructed in the proof of Lemma 3.3.

Hereafter, the same construction as Lemma 3.3 is continued. However, only the vertex set is different. For each $l = 0, 1, \dots, k$ and each $\alpha = 0, 1, \dots, 2^h - 1$, we define $\Gamma_{k+l} := \bigcup_{\alpha=0}^{2^h-1} \Gamma_{k+l}(\alpha)$, where

$$\Gamma_{k+l}(\alpha) := \{\{u_1, \dots, u_{k+l}\}; \forall i \neq j, u_i \neq u_j \text{ and } \forall i, V'(u_i) = \alpha, u_i \in \mathcal{U}\}.$$

Notice that k is constant. For every given string $v = \{u_1, \dots, u_{k+l}\}$, we can check whether v is in Γ_{k+l} in polynomial time. Moreover, the bit-length of string $v = \{u_1, \dots, u_{k+l}\} \in \Gamma_{k+l}$ is bounded by polynomial in n. We need $(2n + h)$-bits to represent each u_i, and thus, $(2k + 1)(2n + h)$-bits are sufficient to represent v.

For each $l = 0, 1, \dots, k$, we define a successor S_{k+l} and a predecessor P_{k+l} over Γ_{k+l} as follows.

- If l is even, then for each $v = \{u_1, \dots, u_{k+l}\}$, we define

$$S_{k+l}(v) := \begin{cases} \{u_1, \dots, u_{k+l}\} & \text{if there exists } u_i \text{ that is a self-loop,} \\ \{S'(u_1), \dots, S'(u_{k+l})\} & \text{otherwise,} \end{cases}$$

  and

$$P_{k+l}(v) := \begin{cases} \{u_1, \dots, u_{k+l}\} & \text{if there exists } u_i \text{ that is a self-loop,} \\ \{P'(u_1), \dots, P'(u_{k+l})\} & \text{otherwise.} \end{cases}$$

- If l is odd, then for each $v = \{u_1, \dots, u_{k+l}\}$, we define

$$S_{k+l}(v) := \begin{cases} \{u_1, \dots, u_{k+l}\} & \text{if there exists } u_i \text{ that is a self-loop,} \\ \{P'(u_1), \dots, P'(u_{k+l})\} & \text{otherwise,} \end{cases}$$

  and

$$P_{k+l}(v) := \begin{cases} \{u_1, \dots, u_{k+l}\} & \text{if there exists } u_i \text{ that is a self-loop,} \\ \{S'(u_1), \dots, S'(u_{k+l})\} & \text{otherwise.} \end{cases}$$

Note that the direction of the valid arc is different when l is even and odd. Furthermore, we define the potential function V_{k+l} depending on the parity of l as follows. For each vertex $v = \{u_1, \dots, u_{k+l}\}$ in Γ_{k+l}, if v is a self-loop with respect to S' and P', then we define $V_{k+l}(v) = 0$, else we define

$$V_{k+l}(v) := \begin{cases} V'(u_1) & \text{if } l \text{ is even,} \\ (2^h - 1) - V'(u_1) & \text{if } l \text{ is odd.} \end{cases}$$

Finally, we define by $\Pi' = \{(\pi_1, \pi_1, 0), \dots, (\pi_k, \pi_k, 0)\}$ the standard source, where π_i is in Π for each $i \in [k]$.

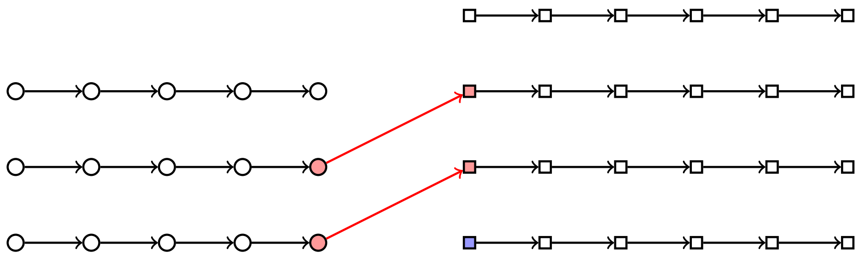

Fig. 3. The paths consisting of rounded vertices are paths where l is odd. On the other hand, the paths consisting of square vertices are paths where l is even. The red vertices are bad solutions. The blue vertex is a standard source. The red arcs represent the new arc introduced by the method of Hollender and Goldberg [18]. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

We denote by $G_{k+l}(S_{k+l}, P_{k+l}) = (\Gamma_{k+l}, E_{k+l})$ the implicit digraph that are derived from S_{k+l}, P_{k+l} for each $l = 0, 1, \dots, k$. Moreover, the implicit graph $G_{k+l}(S_{k+l}, P_{k+l})$ has the potential function V_{k+l} for each $l = 0, 1, \dots, k$. To simply the notation, we denote by $G_{k+l}(S_{k+l}, P_{k+l}, V_{k+l})$ this implicit digraph with potential. It is easy to see that $G_{k+l}(S_{k+l}, P_{k+l}, V_{k+l})$ has the following sinks and sources:

- Let $t_1, \dots, t_{k+l}$ be distinct $k+l$ sinks with respect to S' and P'. Then the vertex $\{t_1, \dots, t_{k+l}\}$ in Γ_{k+l} is a sink of G_{k+l} if l is even, and $\{t_1, \dots, t_{k+l}\}$ in Γ_{k+l} is a non-standard source of G_{k+l} if l is odd. Therefore, we can extract at least k original solutions.
- Let $s_1, \dots, s_{k+l}$ be distinct $k+l$ sources with respect to S' and P'. Then the vertex $\{s_1, \dots, s_{k+l}\}$ in Γ_{k+l} is a source of G_{k+l} if l is even, and $\{s_1, \dots, s_{k+l}\}$ in Γ_{k+l} is a sink of G_{k+l} if l is odd. However, it might contain strictly less than k non-standard sources of the original instance, and thus, we can not extract k distinct original solutions. In this case, we call $\{s_1, \dots, s_{k+l}\}$ a bad solution.

To complete our reduction from k-EoPL to EndOfPotentialLine, we need to remove every bad solution. Hollender and Goldberg [18] established the method of removing every bad solution from a sink and a non-standard source of G_{k+l}. They made a one-to-one correspondence between bad solutions that are sinks and bad solutions that are sources and connected two bad solutions. Thereby, their technique removed bad solutions from sinks and non-standard sources. See Fig. 3. Notice that a bad solution is a sink when l is odd, and a bad solution is a source when l is even, we can efficiently compute the potential without generating violating solutions.

Our reduction heavily relies on the elegant technique of Hollender and Goldberg [18]; in the following, we apply their method to construct the new successor circuit $\hat{S}$ and the new predecessor circuit $\hat{P}$.

First, we fix a strict order on the set of standard sources Π. Here, we suppose that an order such $\pi_1 \prec \pi_2 \prec \cdots \prec \pi_k$ is given. Then we consider the set $\hat{\Gamma} = \bigcup_{l=0}^{k} \hat{\Gamma}_{k+l}$, where $\hat{\Gamma}_{k+l} := \{(v, (a_1, \dots, a_l));\ v \in \Gamma_{k+l}, a_i \in \Pi \text{ for all } i \in [l], \text{ and } a_1 \preceq a_2 \preceq \cdots \preceq a_l\}$. In particular, for $l = 0$, the elements of $\hat{\Gamma}_{k+l}$ are the form $(v, ())$, where $()$ denotes the empty tuple. Furthermore, the bit-length of a string $(v, (a_1, \dots, a_l)) \in \hat{\Gamma}$ is bounded by some polynomial in n. We need at most $(2k+1)(2n+h)$-bits to represent v, $(2n+h)$-bits to represent each a_l, and $(\log_2(l)+1)$-bits to express the value l. Hence, $((2k+1)(2n+h) + k(2n+h) + \log_2(k) + 1)$-bits are sufficient to represent $(v, (a_1, \dots, a_l))$. This is a polynomial in n since k is a constant number.

We define the new successor circuit $\hat{S}$ and the new predecessor circuit $\hat{P}$ for this vertex set $\hat{\Gamma}$ as follows. First, we consider the case of that l is even. For each vertex $(v, (a_1, \dots, a_l))$ in $\hat{\Gamma}$, if v is not a source, then we define $\hat{S}(v, (a_1, \dots, a_l)) := (S_{k+l}(v), (a_1, \dots, a_l))$ and $\hat{P}(v, (a_1, \dots, a_l)) := (P_{k+l}(v), (a_1, \dots, a_l))$. On the other hand, if v is a source, i.e., it satisfies that $S_{k+l}(P_{k+l}(v)) \neq v$, then we can write $v = K \cup U$, where $K \subseteq \Pi'$ and $U \subseteq \Gamma_{k+l} \setminus \Pi'$. Note that $K \cap U = \emptyset$ holds. If $|U| \geq k$, then we can extract k distinct original solutions, and thus, we define $\hat{S}(v, (a_1, \dots, a_l)) := (S_{k+l}(v), (a_1, \dots, a_l))$ and $\hat{P}(v, (a_1, \dots, a_l)) := (P_{k+l}(v), (a_1, \dots, a_l))$. However, if $|U| < k$, then we can not extract k distinct original solutions. Hence, we introduce the new valid arc to remove the vertex $(v, (a_1, \dots, a_l))$ from a non-standard source.

Here, let $\bar{K} = \Pi' \setminus K$, and define $\max(X) := \arg\max_{\prec} X$ for every subset $X \subseteq \Pi'$; specifically, let $a_1 \succ \max(\emptyset)$. For vertex $v = K \cup U$, if $a_l \succ \max(\bar{K})$, then we define

$$\hat{S}(v, (a_1, \dots, a_l)) := (S_{k+l}(v), (a_1, \dots, a_l))$$

and

$$\hat{P}(v, (a_1, \dots, a_l)) := ((K \setminus \{a_l\} \cup U), (a_1, \dots, a_{l-1})).$$

On the other hand, if $a_l \preceq \max(\bar{K}) =: j$ and $l > 0$, then we define

$$\hat{S}(v, (a_1, \dots, a_l)) := (S_{k+l}(v), (a_1, \dots, a_l))$$

and

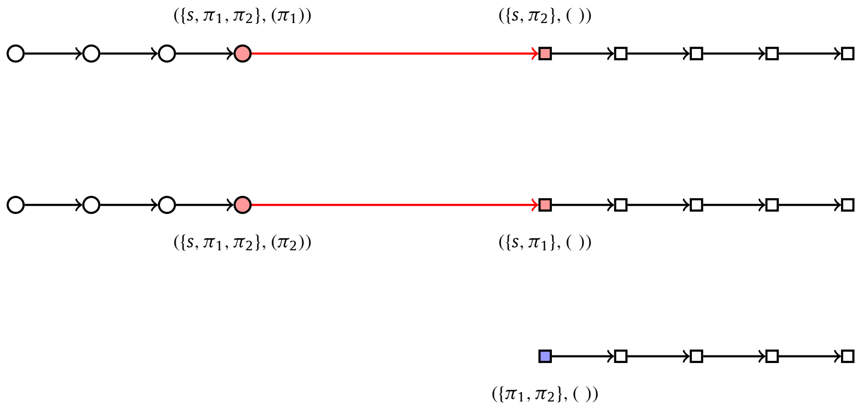

Fig. 4. An example of removing bad solutions when $k=2$. An element s is any source, and π_i is a standard source for each $i=1,2$. The paths consisting of rounded vertices are paths where $l=1$. On the other hand, the paths consisting of square vertices are paths where $l=0$. The red vertices are bad solutions. The blue vertex is a standard source. Note that when $l=2$, every sink and every source are solution that we can extract two original solutions.

$$\hat{P}(v,(a_1,\dots,a_l)) := ((K\cup\{j\})\cup U),(a_1,\dots,a_l,j)),$$

else if $a_l \preceq \max(\bar{K}) =: j$ and $l=0$, then define as follows.

$$\hat{S}(v,()) := (S_{k+l}(v),())$$

and

$$\hat{P}(v,()) := \begin{cases} ((K\cup\{j\})\cup U),(j)) & \text{if } |U|>0, \\ (v,()) & \text{if } |U|=0. \end{cases}$$

Next, we consider the case that l is odd. For each vertex $(v,(a_1,\dots,a_l))\in\hat{\Gamma}$, if v is not a sink, then $\hat{S}(v,(a_1,\dots,a_l)) = (S_{k+l}(v),(a_1,\dots,a_l))$ and $\hat{P}(v,(a_1,\dots,a_l)) = (P_{k+l}(v),(a_1,\dots,a_l))$. On the other hand, if v is a sink, i.e., it satisfies that $P'(S'(v)) \neq v$, then we can write $v = K\cup U$, where $K\subseteq \Pi'$ and $U\subseteq \Gamma_{k+l}\setminus\Pi'$ with $K\cap U=\emptyset$. If $|U|\geq k$, then we can extract k distinct original solutions. Hence, we define $\hat{S}(v,(a_1,\dots,a_l)) = (S_{k+l}(v),(a_1,\dots,a_l))$ and $\hat{P}(v,(a_1,\dots,a_l)) = (P_{k+l}(v),(a_1,\dots,a_l))$. However, if $|U|<k$, then we can not extract k distinct original solutions. Hence, we introduce the new valid arc to remove the vertex $(v,(a_1,\dots,a_l))$ from a sink.

For vertex $v=K\cup U$, if $a_l \succ \max(\bar{K})$, then we define

$$\hat{S}(v,(a_1,\dots,a_l)) := ((K\setminus\{a_l\})\cup U),(a_1,\dots,a_{l-1}))$$

and

$$\hat{P}(v,(a_1,\dots,a_l)) := (P_{k+l}(v),(a_1,\dots,a_l)).$$

On the other hand, if $a_l \preceq \max(\bar{K}) =: j$ and $l>0$, then we define

$$\hat{S}(v,(a_1,\dots,a_l)) := ((K\cup\{j\})\cup U),(a_1,\dots,a_l,j))$$

and

$$\hat{P}(v,(a_1,\dots,a_l)) := (P_{k+l}(v),(a_1,\dots,a_l)).$$

From the above definitions $\hat{S}$ and $\hat{P}$, the every bad solution is removed from a sink and a standard source. This completes the application of the technique by Hollender and Goldberg [18]. We can immediately see that their method removes every bad solution. In Fig. 4, we show an example of their method for $k=2$. What remains is to define the potential function.

Finally, we define the new potential function $\hat{V}$ to complete the reduction. For each vertex $(v,(a_1,\dots,a_l))$, we define

$$\hat{V}(v,(a_1,\dots,a_l)) := 2^h\chi_{even}(l) + V_{k+l}(v),$$

where $\chi_{even}(l)$ is 1 if l is even, and is 0 if l is odd. Exceptionally, we define $\hat{V}(\Pi',())=0$.

It is easy to see that this construction is polynomial-time computable. The tuple $(\hat{S},\hat{P},\hat{V},\Pi')$ is a valid instance of EndOfPotentialLine. Moreover, we can efficiently extract an original solution from a sink and a non-standard source on $\hat{S}$ and $\hat{P}$.

What remains is to verify that each valid arc added to remove a bad solution is an increasing arc. We prove that each bad solution is not a violating solution for $(\hat{S}, \hat{P}, \hat{V}, \Pi')$, and we complete the polynomial-time reduction from k-EndOfPotentialLine to EndOfPotentialLine.

First, if l is even, then every bad solution $(v, (a_1, \dots, a_l))$ is a non-standard source for S_{k+l} and P_{k+l}. In particular, it satisfies that $V_{k+l}(v) = 0$. Therefore, the vertex $(v, (a_1, \dots, a_l))$ is assigned the potential $\hat{V}(v, (a_1, \dots, a_l)) = 2^h$. Now, let $(w, \tau) = \hat{P}(v, (a_1, \dots, a_l))$. By the definition of $\hat{V}$, it holds that $\hat{V}(w, \tau) = 2^h - 1 < 2^h = \hat{V}(v, (a_1, \dots, a_l))$. Moreover, it satisfies that $(w, \tau) = \hat{P}(v, (a_1, \dots, a_l))$ and $\hat{S}(w, \tau) = (v, (a_1, \dots, a_l))$. That is, the vertex $(v, (a_1, \dots, a_l))$ is not a solution.

Next if l is odd, the every bad solution $(v, (a_1, \dots, a_l))$ is a sink for S_{k+l} and P_{k+l}. In particular, it satisfies that $V_{k+l}(v) = 2^h - 1$. Therefore, it holds that $\hat{V}(v, (a_1, \dots, a_l)) = 2^h - 1$. Now, let $(w, \tau) = \hat{S}(v, (a_1, \dots, a_l))$. By the definition of $\hat{V}$, it holds that $\hat{V}(w, \tau) = 2^h > 2^h - 1 = \hat{V}(v, (a_1, \dots, a_l))$. Moreover, it satisfies that $(w, \tau) = \hat{S}(v, (a_1, \dots, a_l))$ and $\hat{P}(w, \tau) = (v, (a_1, \dots, a_l))$. Hence, the vertex $(v, (a_1, \dots, a_l))$ is not a solution. □

Immediately, we get EOPL-completeness of k-EndofPotentialLine by combining the above lemmata.

Theorem 3.19. *k-EoPL is EOPL-complete.*

4. The hardness of parity argument with potential

In this section, we study the complexity of the parity argument with potential. As mentioned in Section 2.3.4, the class EOPL is characterized by EndOfPotentialLine. Up to this point, we show that the classification by this problem is robust. Recall the definition of EndOfPotentialLine. This problem can be viewed as the problem that each instance of EndOfLine relaxed by a potential condition, in which every valid arc is an increasing arc. Incidentally, every undirected graph with potential can be introduced to a natural orientation by its potential. The following natural question arises. How hard is a PPA-complete problem which is given a potential condition, e.g., the problem PotentialLeaf (see Definition 4.1)? Also, is the classification based on such a problem robust? In Section 4.1, we show that problem of finding an unknown leaf or a local optimum vertex on a given graph with potential whose vertex has degree at most two is EOPL-complete. However, the problem of finding an unknown odd-degree vertex or a local optimum vertex on a given graph with potential is much harder than EOPL; specifically, this problem is PPA ∩ PLS-complete (see Section 4.2 for details).

4.1. The problem: POTENTIAL LEAF

In this section, we introduce the new problem called PotentialLeaf. This problem is the most simple generalization of EndOfPotentialLine. Informally speaking, this problem is defined as: given an implicit undirected graph that every vertex has degree at most two, a potential function, and a known leaf, find at least one of another leaf and a local optimum solution. Intuitively, this problem is EOPL-hard. We can construct a polynomial-time reduction by replacing every valid arc on an instance of EndOfPotentialLine by an undirected edge. Not surprisingly, the problem PotentialLeaf is polynomially equivalent to the problem EndOfPotentialLine; we can introduce a natural direction to each edge on an instance of PotentialLeaf.

Definition 4.1 *(*PotentialLeaf*).* A valid instance of PotentialLeaf consists of two positive integers n and m, an implicit graph with potential $G(N, V) = (\Sigma^n, E)$, and a vertex $\pi \in \Sigma^n$ such that $\deg_G(\pi) = 1$ and $V(\pi) = 0$, where $G(N, V)$ is produced by a neighborhood circuit $N : \Sigma^n \to \Sigma^{2n}$ and a potential circuit $V : \Sigma^n \to \{0, 1, \dots, 2^m - 1\}$. This problem is defined as: given a valid instance (n, m, N, V, π), find a non-isolated vertex satisfying at least one of the following:

(R1) $x \neq \pi$ and $\deg_G(x) = 1$,
(R2) $x \neq \pi$ and $V(x) \leq V(y)$ for every valid edge $\{x, y\}$ in E, and
(R3) $V(x) \geq V(y)$ for every valid edge $\{x, y\}$ in E.

Theorem 4.2. PotentialLeaf *is an EOPL-complete problem.*

Proof. By the definition of PotentialLeaf, it is straightforward to see that this problem is EOPL-hard. Therefore, it suffices to prove that PotentialLeaf is polynomial-time reducible to EndOfPotentialLine.

Let (n, m, N, V, π) be a valid instance of PotentialLeaf. Without loss of generality, we assume that the neighborhood function N always computes valid edges, i.e., it satisfies that for each pair of vertices x and y in Σ^n, $y \in N(x)$ if and only if $x \in N(y)$.

We construct the successor circuit S and the predecessor circuit P as follows. Let $N(\pi) = \{\rho\}$, where $\rho \neq \pi$. We first define $S(\pi) := \rho$ and $P(\pi) = \pi$. For every vertex $x \in \Sigma^n$ with $x \neq \pi$, we define $S(x) := \arg\max\{y \in N(x); V(y) > V(x)\}$ and $P(x) := \arg\max\{y \in N(x); V(y) < V(x)\}$. However, select exactly one that is lexicographically small if there are two vertices that have the same potential. Furthermore, if $\arg\max\{y \in N(x); V(y) > V(x)\}$ is empty, then the successor circuit S outputs x, similarly if $\arg\max\{y \in N(x); V(y) < V(x)\}$ is empty, then the successor circuit P outputs x.

From the above definitions, it holds that $P(\pi) = \pi \neq S(\pi)$ and $V(\pi) = 0$. Therefore, the tuple (n, m, S, P, V, π) is a valid instance of EndOfPotentialLine. In the rest of this proof, we show that when we obtain a solution of (n, m, S, P, V, π), we can convert to an original solution in polynomial time.

First, we consider when we obtain a solution $x \in \Sigma^n$ satisfying that $P(S(x)) \neq x$. If $S(x) = x$, then for every $y \in N(x)$, it holds that $V(y) \leq V(x)$ by definition. Thus, x is a local maximum solution. Otherwise, i.e., $S(x) \neq x$, then it satisfies that $V(S(x)) > V(x)$. Since $P(S(x)) \neq x$, there is a vertex $z \in N(S(x))$ such that $V(z) \leq V(x) < V(S(x))$ and $z \neq x$. Note that $x \in N(S(x))$, and thus, $N(S(x)) = \{x, z\}$. Hence, $S(x)$ is a local maximum solution since $V(S(x)) > V(x)$ and $V(S(x)) > V(z)$.

Second, we consider when we obtain a solution x in Σ^n satisfying that $S(P(x)) \neq x \neq \pi$. If $P(x) = x$, then for every $y \in N(x)$, it holds that $V(y) \geq V(x)$ by definition. This implies that x is a local minimum solution. Otherwise, i.e., $P(x) \neq x$, then it satisfies that $V(P(x)) < V(x)$. Furthermore, there exists a vertex $z \in N(P(x))$ such that $V(P(x)) < V(x) \leq V(z)$ and $z \neq x$ since $S(P(x)) \neq x$. Note that $x \in N(P(x))$, and thus, $N(P(x)) = \{x, z\}$. Hence, $P(x)$ is a local minimum solution since $V(P(x)) < V(x)$ and $V(P(x)) < V(z)$.

Finally, it is easy to see that there are no violating solutions for the instance (n, m, S, P, V, π). From the above arguments, we complete the polynomial-time reduction from PotentialLeaf to EndOfPotentialLine. Therefore, the problem PotentialLeaf is an EOPL-complete problem. □

4.2. The problem: POTENTIAL ODD

In Section 4.1, we have only considered graphs where every vertex has degree at most two. Now, we generalize the problem PotentialLeaf to a new search problem on higher degree graphs, called PotentialOdd. This problem is defined as follows.

Definition 4.3 *(PotentialOdd).* A valid instance of PotentialOdd consists of three positive integers n, m, and d, an implicit graph with potential $G(N, V) = (\Sigma^n, E)$, and an odd-degree vertex $\pi \in \Sigma^n$ on $G(N, V)$, where $G(N, V)$ is produced by a neighborhood circuit $N : \Sigma^n \to \Sigma^{dn}$ and a potential circuit $V : \Sigma^n \to \{0, 1, \dots, 2^m - 1\}$. This problem is defined as: given a valid instance (n, m, d, N, V, π), find a non-isolated vertex $x \in \Sigma^n$ satisfying at least one of the following:

(R1) $x \neq \pi$ and $\deg_G(x) = 2k + 1$ for some non-negative integer k,
(R2) $V(x) \geq V(y)$ for every $y \in N(x)$ with $\{x, y\} \in E$, and
(R3) $V(x) \leq V(y)$ for every $y \in N(x)$ with $\{x, y\} \in E$.

It is clear to see that PotentialOdd belongs to PPA ∩ PLS. Recall that the parity argument guarantees that every finite graph has an even number of odd-degree vertices, that is, at least one unknown odd-degree vertex exists when we have a known odd-degree vertex. Naturally, this principle works for every finite graph with potential. Moreover, we can find a local minimum or maximum vertex by applying a local search method.

Therefore, in the rest of this paper, we focus on the hardness of this problem. First, we show that PotentialOdd is, generally, PPA ∩ PLS-hard, and thus, this problem is a PPA ∩ PLS-complete problem.

Theorem 4.4. PotentialOdd *is a* PPA ∩ PLS*-complete problem.*

Proof. We first show that PotentialOdd belongs to PPA ∩ PLS. Let $\mathcal{I} = (n, m, d, N, V, \pi)$ be a valid instance of PotentialOdd. Then, the tuple (n, d, N, π) is a valid instance of Odd. Furthermore, every solution of (n, d, N, π) is also a solution to $\mathcal{I}$. Hence, PotentialOdd is in PPA.

We prove that PotentialOdd also belongs to PLS. To prove this, we construct a polynomial-time reduction from PotentialOdd to LocalOPT. Without loss of generality, we assume that the known odd-degree vertex π has positive potential, i.e., $V(\pi) > 0$. If $V(\pi) = 0$, then π is obviously a solution to $\mathcal{I}$. We consider the implicit graph with potential $G(N, V) = (\Sigma^n, E)$ produced by N and V, where E is a set of all valid edges for N. Then, the function $f : \Sigma^n \to \Sigma^n$ is defined as follows. For every vertex $x \in \Sigma^n$,

$$f(x) = \arg\max\{V(y); y \in N(x) \text{ and } \{x, y\} \in E\}.$$

Specifically, select exactly one lexicographically small if there are some vertices with the same potential. If a vertex $x \in \Sigma^n$ is an isolated vertex, we define $f(x) = \pi$ and replace its potential by 0, i.e., $V(x) = 0$. Note that an isolated vertex x is not a local maximum for f from our assumption that $V(\pi) > 0$. The tuple (f, V) is a valid instance of LocalOPT. Every solution $x \in \Sigma^n$ to (f, V) satisfies that $V(f(x)) \leq V(x)$. We have that $V(y) \leq V(x)$ for each $y \in N(x)$ with $\{x, y\} \in E$, and thus, x is also a solution to $\mathcal{I}$. Therefore, PotentialOdd belongs to PLS.

From the above arguments, PotentialOdd is in PPA ∩ PLS.

To prove that PotentialOdd is PPA ∩ PLS-hard, we show that EitherSolution(Odd, Flip) is reducible to PotentialOdd in polynomial time. Let $\mathcal{I} = (n_1, d, N, \pi, n_2, m_1, C)$ be a valid instance of EitherSolution(Odd, Flip). Then, the tuple (n_1, d, N, π) is a valid instance of Odd. Here, we denote by $G(N) = (\Sigma^{n_1}, E)$ the implicit graph produced by N, and let

k_0 be a non-negative integer such that $\deg_G(\pi) = 2k_0 + 1$. The tuple (n_2, m_1, C) is a valid instance of Flip, where the given Boolean circuit C computes a polynomial-time computable function $f : \Sigma^{n_2} \to \{0, 1, \ldots, 2^{m_1} - 1\}$. Without loss of generality, we assume that n_2 is even.

From now on, we construct a valid instance $\mathcal{J} = (n_1 + n_2, m_1, n_2 + d, N', V', \pi')$ of PotentialOdd. We first denote by $U = \Sigma^{n_1} \times \Sigma^{n_2}$ the new vertex set. For each $x = (x_1, x_2) \in U$, we define

$$N'(x) := \{(x_1, y_2) \in U; \|x_2 - y_2\| = 1\} \cup \{(y_1, x_2) \in U; x_2 = 0^{n_2} \text{ and } \{x_1, y_1\} \in E\}.$$

Note that $|N'(x)| \leq n_2 + \deg_G(x) \leq n_2 + d$ for every vertex $x \in U$, and thus, it is bounded by some polynomial in n_1 and n_2. This implies that the function N' is polynomial-time computable. Furthermore, we define the potential function as $V'(x_1, x_2) = C(x_2)$ for every $(x_1, x_2) \in U$, and let $\pi' = (\pi, 0^{n_2})$. We denote by $H(N', V') = (U, F)$ the implicit graph with potential induced by N' and V', where F is a set of all valid edges defined by N'. Then, it satisfies that $\deg_H(\pi') = n_2 + 2k_0 + 1$. Since n_2 is even and k_0 is a non-negative integer, π' is an odd-degree vertex on the graph H. Therefore, the tuple $\mathcal{J} = (n_1 + n_2, m_1, n_2 + d, N', V', \pi')$ is a valid instance of PotentialOdd. What remains is to prove that we can extract a solution to $\mathcal{I}$ in polynomial time when we obtain a solution to $\mathcal{J}$.

We first consider when we obtain a vertex $x = (x_1, x_2) \in U \setminus \{\pi'\}$ satisfying that $\deg_H(x) = 2k + 1$, where k is a non-negative integer. For every vertex $z = (z_1, z_2) \in U$ satisfying that $z_2 \neq 0^{n_2}$, it holds that $\deg_H(z) = n_2$ since $N'(x) = \{(z_1, y_2) \in U; \|z_2 - y_2\| = 1\}$. Assuming that n_2 is even, z is an even-degree vertex. Therefore, it must hold that $x_2 = 0^{n_2}$. Then, it satisfies that $\deg_H(x) = n_2 + \deg_G(x_1)$, and thus, $\deg_G(x_1)$ is odd since n_2 is even and $\deg_H(x)$ is odd. This implies that the vertex x_1 has odd degree. Furthermore, it holds that $x_1 \neq \pi$ since $x = (x_1, 0^{n_2}) \neq \pi' = (\pi, 0^{n_2})$. Hence, the vertex x_1 is a solution to $\mathcal{I}$.

Next, we consider when we obtain a vertex $x = (x_1, x_2) \in U$ satisfying that $V'(x) \leq V'(y)$ for every vertex $y = (y_1, y_2) \in N'(x)$. By the definition of V' and N', it holds that $C(x_2) \leq C(y_2)$ for every y_2 with $\|x_2 - y_2\| = 1$. Hence, x_2 is a solution for Flip, and it is a solution to $\mathcal{I}$. Similarly, when we obtain a vertex $x = (x_1, x_2) \in U$ satisfying that $V'(x) \geq V'(y)$ for every vertex $y = (y_1, y_2) \in N'(x)$, the string x_2 is a solution to $\mathcal{I}$.

From the above arguments, we complete the polynomial-time reduction from EitherSolution(Odd, Flip) to PotentialOdd. Therefore, PotentialOdd is $\mathtt{PPA} \cap \mathtt{PLS}$-hard. □

4.3. Variants of ODD with potential

In this section, we scrutinize the complexity classification of these problems in more detail. In the previous sections, we show that PotentialLeaf, which is associated with graphs whose every vertex has degree at most two, is an $\mathtt{EOPL}$-complete problem. However, PotentialOdd, which is associated with higher degree graphs, is a $\mathtt{PPA} \cap \mathtt{PLS}$-complete problem.

The most natural question is where the boundaries of the complexity of PotentialOdd lie. We focus on the maximum degree of a given implicit graph. Now, we consider the constant degree variant of PotentialOdd. In this problem, the maximum degree of the implicit graph is bounded by some constant. We prove that even if the maximum degree is 4, then PotentialOdd is $\mathtt{PPA} \cap \mathtt{PLS}$-complete, but if the maximum degree is at most 3, then PotentialOdd is $\mathtt{EOPL}$-complete.

Definition 4.5 *(*Degree-d PotentialOdd*).* Let d be a positive integer. A valid instance of Degree-d PotentialOdd consists of two positive integers n and m, an implicit graph with potential $G(N, V) = (\Sigma^n, E)$, and an odd-degree vertex $\pi \in \Sigma^n$ on $G(N, V)$, where $G(N, V)$ is produced by a neighborhood function $N : \Sigma^n \to \Sigma^{dn}$ and a potential function $V : \Sigma^n \to \{0, 1, \ldots, 2^m - 1\}$. Then this problem is defined as: given a valid instance (n, m, N, V, π), find a non-isolated vertex $x \in \Sigma^n$ satisfying at least one of the following:

(R1) $x \neq \pi$ and $\deg_G(x) = 2k + 1$ for some non-negative integer k,
(R2) $V(x) \geq V(y)$ for every $y \in N(x)$ with $\{x, y\} \in E$, and
(R3) $V(x) \leq V(y)$ for every $y \in N(x)$ with $\{x, y\} \in E$.

In order to simplify the following arguments, we prove the following proposition.

Proposition 4.6. *Without loss of generality, we assume that for every instance of* PotentialOdd*, each vertex has a different potential.*

Proof. To prove this, we show that when we are given a valid instance $\mathcal{I} = (n, m, d, N, V, \pi)$ of PotentialOdd, we can construct another instance $\mathcal{I}' = (n, m + n, N, V', \pi)$ that satisfies that $V'(x) \neq V'(y)$ for each pair of vertices $x, y \in \Sigma^n$ with $x \neq y$. Moreover, we can convert from a solution from $\mathcal{I}'$ to a solution of $\mathcal{I}$ in polynomial time.

For every string x in Σ^n, we define

$$V'(x) := 2^n V(x) + \sum_{i=1}^{n} 2^{i-1} x_i,$$

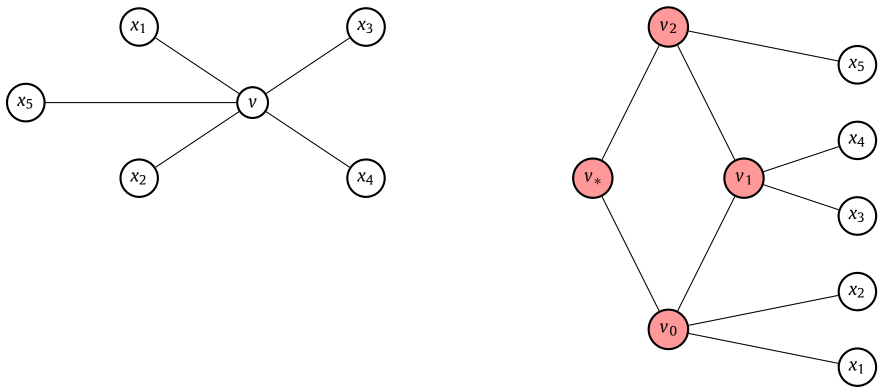

Fig. 5. Illustration of construction.

where let $x = x_1x_2 \dots x_n$. Clearly, it satisfies that $V'(x) \neq V'(y)$ for each pair of vertices $x, y \in \Sigma^n$ with $x \neq y$. Furthermore, for each pair of $x, y \in \Sigma^n$, if $V'(x) < V'(y)$ then $V(x) \leq V(y)$. Therefore, if a vertex x in Σ^n satisfying that $V'(x) \geq V'(y)$ for every $y \in N(x)$ with $\{x, y\} \in E$, then x is a solution of $\mathcal{I}$. Similarly, if a vertex x in Σ^n satisfying that $V'(x) \leq V'(y)$ for every $y \in N(x)$ with $\{x, y\} \in E$, then x is a solution of $\mathcal{I}$. Also, a vertex x is an odd-degree vertex for $\mathcal{I}'$ if and only if x is an odd-degree vertex for $\mathcal{I}$. □

Notice that our proof of Proposition 4.6 also works for Degree-d PotentialOdd. Without loss of generality, we assume that for every instance of Degree-d PotentialOdd, every vertex on the given implicit graph has a different potential.

First, we show that Degree-4 PotentialOdd is PPA ∩ PLS-complete.

Theorem 4.7. Degree-4 PotentialOdd *is* PPA ∩ PLS*-complete.*

Proof. It is easy to see that Degree-4 PotentialOdd belongs to PPA ∩ PLS. Therefore, it suffices to prove that Degree-4 PotentialOdd is PPA ∩ PLS-hard. To prove this, we show that there exists a polynomial-time reduction from PotentialOdd to Degree-4 PotentialOdd.

The fundamental idea of our proof is that we simulate every vertex with degree > 4 by some vertices with degree ≤ 4. Before beginning our proof, we illustrate the main idea. See Fig. 5 for an illustration of the construction. In this figure, the vertex v has degree > 4 on the left graph, and we assume that the original potential function V satisfies that $V(x_1) < V(x_2) < V(x_4) < V(x_5)$. We construct the right graph from the left one. The red vertices on the right graph simulate the vertex v. All adjacent vertices of v are placed adjacent to v_0, v_1, and v_2 in order of increasing potential. We define the new potential function V' such that $V'(v_0) < V'(v_1) < V'(v_2)$ and $V'(v_0) < V'(v_*) < V'(v_2)$. Furthermore, V' follows that $V'(v_j) > V'(x_i)$ if and only if $V(v) > V(x_i)$ for all i and j. From our construction, it is easy to see that the following three properties hold:

- v is a local minimum if and only if v_0 is a local minimum,
- v is a local maximum if and only if v_2 is a local maximum, and
- v has an odd degree if and only if v_2 has an odd degree.

We do this for all vertices on the original graph. Notice that every red vertex has degree at most 4. This construction allows us to simulate a vertex with degree > 4 by some vertices with degree ≤ 4. In the rest of this proof, we show that the above construction is polynomial-time computable.

Let $\mathcal{I} = (n, m, d, N, V, \pi)$ be a valid instance of PotentialOdd. Without loss of generality, we assume that every vertex of the instance $\mathcal{I}$ has a different potential, i.e., $V(x) \neq V(y)$ for each pair of vertices $x, y \in \Sigma^n$ with $x \neq y$. We denote by $G(N, V) = (\Sigma^n, E)$ the implicit graph with potential produced by N and V, where E is a set of all valid edges defined by N.

We consider the following vertex set

$$U := \hat{U} \cup \{(*, x); x \in \Sigma^n \text{ and } \deg_G(x) \geq 3\},$$

where $*$ is a special symbol, and the set $\hat{U}$ is defined as follows:

$$\hat{U} := \{(i, x); x \in \Sigma^n \text{ and } 1 \leq i \leq \left\lceil \frac{\deg_G(x)}{2} \right\rceil\}.$$

We construct the new neighborhood function $N' : U \to U^{4d}$ and the new potential function $V' : U \to \{0, 1, \dots, 2^{2m+d}\}$. Before we construct these two functions, we define the following three functions. First, for each vertex x in Σ^n and each valid edge $\{x, y\}$ in E, we define

$$rank_x(\{x, y\}) := 1 + |\{\{x, z\} \in E; V(z) < V(y)\}|.$$

Note that this function is bijective. Second, for each vertex x in Σ^n and each vertex y in Σ^n with $\{x, y\}$ in E, we define

$$\lambda_x(y) = \left\lceil \frac{rank_x(\{x, y\})}{2} \right\rceil.$$

Finally, for each vertex (i, x) in $\hat{U}$, we define

$$\hat{N}(i, x) := \{(j_1, y_1), (j_2, y_2)\},$$

where it satisfies that $\lambda_x(y_1) = \lambda_x(y_2) = i$ and $j_k = \lambda_{y_k}(x)$ for each $k = 1, 2$. If there is only one vertex y in Σ^n such that $\lambda_x(y) = i$, then we define $\hat{N}(i, x) := \{(j, y)\}$, where it satisfies that $j = \lambda_y(x)$. If there are no vertices y in Σ^n such that $\lambda_x(y) = i$, we define that $\hat{N}(i, x)$ is empty. However, for each $(*, x)$ in U, we define $\hat{N}(*, x) := \emptyset$. By using the function $\hat{N}$, for each vertex (i, x) in U, the neighborhood function N' is defined as

$$\begin{aligned} N'(i, x) = \hat{N}(i, x) &\cup \{(i+1, x); i < M_x\} \cup \{(i-1, x); i > 0\} \\ &\cup \{(*, x); i \in \{1, M_x\}\} \cup \{(1, x), (M_x, x); M_x \neq 1, i = *\}, \end{aligned}$$

where $M_x = \left\lceil \frac{\deg_G(x)}{2} \right\rceil$. Next, for each (i, x) in U, we define

$$V'(i, x) := \begin{cases} 2^{2m} V(x) + 2(i-1) & \text{if } i \in [M_x], \\ 2^{2m} V(x) + 1 & \text{if } i = *. \end{cases}$$

It is easy to see that given a string (i, x) in $[\deg(G)] \times \Sigma^n$, we can check whether (i, x) is in U in polynomial time. Hence, the neighborhood function N' and the potential function V' are polynomial-time computable and constructed. Furthermore, it is not hard to see that every vertex (i, x) in U has degree at most 4, i.e., it satisfies that $|N'(i, x)| \leq 4$.

From the construction of N', the following claim holds.

Claim 4.8. *A vertex (i, x) in U is an odd-degree vertex on H if and only if it satisfies that $i = M_x$ and x is an odd-degree vertex on G.*

Proof. We first show that if a vertex (i, x) in U is an odd-degree vertex on the graph H, then x is an odd-degree vertex on G by using a contradiction. Assume that x has even degree on G, i.e., there is a non-negative integer k such that $\deg_G(x) = 2k$. Then for each $i' \in [k]$, there exist two distinct vertices z_1 and z_2 such that $rank_x(\{x, z_1\}) = 2i' - 1$ and $rank_x(\{x, z_2\}) = 2i'$. Hence, it holds that $N'(i', x) = \{(j_1, z_1), (j_2, z_2)\}$, where $\lambda_{z_l}(x) = j_l$ for each $l = 1, 2$. This is a contradiction from (i, x) has odd degree on H.

Therefore, x is an odd-degree vertex on G, and thus, there is a non-negative integer k' such that $\deg_G(x) = 2k' + 1$. Then it satisfies that $M_x = k' + 1$. Assuming that $i < M_x$, there exist two distinct vertices z_1 and z_2 such that $rank_x(\{x, z_1\}) = 2i - 1$ and $rank_x(\{x, z_2\}) = 2i$. This implies that (i, x) has even degree on H. This is a contradiction. Hence, it satisfies that $i = M_x$. Then there is only one vertex z such that $rank_x(\{x, z\}) = 2k' + 1$.

Next, we show that if $i = M_x$ and a vertex x in Σ^n is an odd-degree vertex on G, then a vertex (M_x, x) in U is an odd-degree vertex on H. Since x is an odd-degree vertex on G, there is a non-negative integer k'' such that $\deg_G(x) = 2k'' - 1$. Therefore, it holds that

$$\left\lceil \frac{(2k'' - 1) - 1}{2} \right\rceil < \left\lceil \frac{2k'' - 1}{2} \right\rceil = k'' = M_x.$$

Therefore, there is only one vertex y in Σ^n such that $\lambda_x(y) = k''$. This implies that the vertex (M_x, x) is an odd-degree vertex on H. □

Immediately, it is easy to see that $\pi' := (M_\pi, \pi)$ has odd degree on the graph H from the above claim. Hence, the tuple $\mathcal{J} = (n + d + 1, 2m + d, N', V', \pi')$ is a valid instance of Degree-4 PotentialOdd. In the rest of this proof, we show that when we obtain a solution of $\mathcal{J}$, we can extract an original solution.

We first consider when we obtain an odd-degree vertex $(i, x) \neq \pi'$ on the graph H. From the above claim, it is easy to see that $i = M_x$, $x \neq \pi$, and x is an odd-degree vertex on the graph G. Therefore, x is a solution for the original instance $\mathcal{J}$.

Next, we consider when we obtain a vertex (i, x) that is a local minimum solution on H, i.e., it satisfies that $V'(i, x) \leq V'(j, y)$ for every vertex $(j, y) \in U$ with $\{(i, x), (j, y)\} \in F$. Note that $i \neq *$ since it satisfies that $V'(1, x) <$

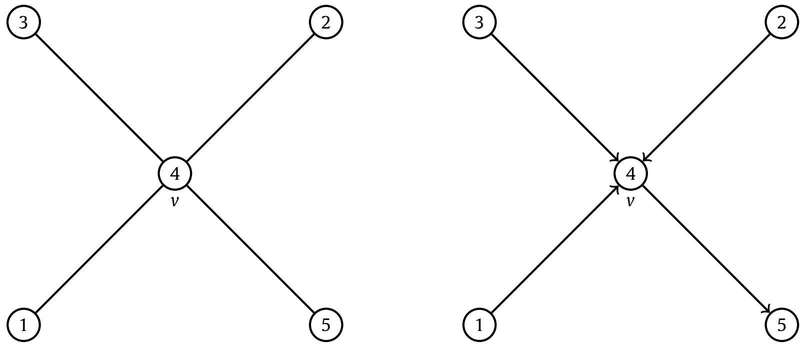

Fig. 6. Left: An original undirected graph. Right: A directed graph which naturally induced from the left one. This figure shows that x is an even-degree vertex on the left graph but x is an unbalanced vertex on the right graph.

$V'(*, x) < V'(M_x, x)$ and $N'(*, x) = \{(1, x), (M_x, x)\}$. Similarly, we can see that $i = 1$. We denote by y_{min} the vertex in U that has the minimum potential among vertices satisfying that $\lambda_x(y) = 1$. Since the vertex (i, x) is a local minimum solution on H, it holds that $V'(i, x) \leq V'(j_{min}, y_{min})$. Thus, $V(x) \leq V(y_{min})$ holds. Therefore, it holds that $V(x) \leq V(y_{min}) \leq V(y)$ for every vertex y in Σ^n with $\{x, y\} \in E$ since y_{min} has the minimum potential among neighbors of x on the graph G. This implies that the vertex x is a solution of the original instance.

From the similar argument, we show that if a vertex (i, x) is a local maximum solution on H, i.e., it satisfies that $V'(i, x) \geq V'(j, y)$ for every vertex $(j, y) \in U$ with $\{(i, x), (j, y)\} \in F$, then the vertex x is a local maximum solution on G, and thus, x is a solution of the original instance.

Therefore, we complete the polynomial-time reduction from PotentialOdd to Degree-4 PotentialOdd. Since PotentialOdd is a $\mathtt{PPA} \cap \mathtt{PLS}$-complete problem, this implies that Degree-4 PotentialOdd is $\mathtt{PPA} \cap \mathtt{PLS}$-hard. Hence, this problem is also $\mathtt{PPA} \cap \mathtt{PLS}$-complete. □

Next, we prove that Degree-3 PotentialOdd is EOPL-complete. We firstly show EOPL-hardness of this problem. Our proof gives a polynomial-time reduction from PotentialLeaf to Degree-3 PotentialOdd. Although every instance of PotentialLeaf is also an instance of Degree-3 PotentialOdd, the known leaf π given an instance of PotentialLeaf is a local minimum vertex, i.e., π is a solution for Degree-3 PotentialOdd. Therefore, it is necessary to avoid that the known leaf becomes a solution for Degree-3 PotentialOdd. For this, we make three copies of the original instance, and we add a vertex with degree three as a known odd-degree vertex.

Lemma 4.9. PotentialLeaf *is reducible to* Degree-3 PotentialOdd *in polynomial time.*

Proof. To prove EOPL-hardness, we construct a polynomial-time reduction from PotentialLeaf to Degree-3 PotentialOdd. Our reduction is simple. Let $\mathcal{I} = (n, m, N, V, \pi)$ be a valid instance of PotentialLeaf. Without loss of generality, we assume that the neighborhood function N always computes valid edges, i.e., it satisfies that for each pair of vertices x and y in Σ^n, $y \in N(x)$ if and only if $x \in N(y)$.

We make three copies of the instance $\mathcal{I}$; $\mathcal{I}_1$, $\mathcal{I}_2$, and $\mathcal{I}_3$, respectively. We denote by π_j the known leaf of $\mathcal{I}_j$, where $j = 1, 2, 3$. We invert the potential of all vertices on $\mathcal{I}_1$, i.e., we define it as $(2^m - 1) - V(x)$ for every vertex x. In the other two instances, we redefine the potential as $2^{m+2} \cdot V(x)$ for every vertex x. Moreover, we add a vertex π^*. The potential of this vertex is 2^{m+1}. The vertex π^* is adjacent to the known leaves of each copied instance π_1, π_2, and π_3. That is, π^* is a vertex with degree three. Note that the known leaf on the original instance is not a solution. Furthermore, π^* is a known odd-degree vertex on the new instance, and it is not a local optimum. It is easy to see that we can extract an original solution from a solution to the new instance. □

It is easy to see that Degree-3 PotentialOdd belongs to EOPL; we can apply the same technique used in the proof of Theorem 4.2. We naturally define every valid arc depending on the potential function. Note that, in the case of degree three, each unbalanced vertex is an odd-degree vertex, and thus, our simple technique works well. Unfortunately, this simple technique does not work to show that Degree-4 PotentialOdd belongs to EOPL. See Fig. 6. The digraph in the right of this figure is naturally induced from the left one. In the right graph, the vertex v is an unbalanced vertex, that is, v is a solution for PotentialImbalance. However, v has an even degree on the left one, that is, v is not a solution for PotentialOdd. Therefore, new ideas seem necessary to prove that Degree-4 PotentialOdd belongs to EOPL.

We prove the containment of Degree-3 PotentialOdd in EOPL via a reduction to a more general variant of PotentialOdd, although we can straightforwardly prove it by the same way used in Theorem 4.2. Thereby, we provide a potential condition for which PotentialOdd belongs to EOPL even through a graph given by an instance with degree ≥ 4.

Informally speaking, the potential condition introduced in the following means that we can well-pair the neighbors of every even-degree vertex that is not local optimum. This condition gives us a limitation that the natural orientation works well for proving that the problem on undirected graphs with potential belongs to EOPL.

Let $G(N, V) = (\Sigma^n, E)$ be an implicit graph with potential produced by a neighborhood function N and a potential function V. We define $E_V^+(x) := \{y \in U; \{x, y\} \in E \text{ and } V(y) > V(x)\}$ and $E_V^-(x) := \{y \in U; \{x, y\} \in E \text{ and } V(y) < V(x)\}$ for each vertex x in Σ^n. A vertex x in Σ^n is said to be well balanced if x satisfies that

$$|E_V^+(x)| = |E_V^-(x)|.$$

We say that the graph $G(N, V)$ is almost balanced if every even-degree vertex that is not local optimum on $G(N, V)$ is well balanced. In Fig. 6, the vertex v is not well balanced. We consider the problem involving such vertices as violations. Hence, the problem called AlmostBalancedOdd requires that the implicit graph with potential given by an instance is almost balanced. This problem is defined as follows.

Definition 4.10 *(AlmostBalancedOdd).* A valid instance of AlmostBalancedOdd consists of three positive integers n, m, and d, an implicit graph with potential $G(N, V) = (\Sigma^n, E)$, and an odd-degree vertex π on $G(N, V)$, where $G(N, V)$ is produced by a neighborhood function $N : \Sigma^n \to \Sigma^{dn}$ and a potential function $V : \Sigma^n \to \{0, 1, \ldots, 2^m - 1\}$. This problem is defined as: given a valid instance (n, m, d, N, V, π), find a non-isolated vertex x in Σ^n satisfying one of the following:

(R1) $x \neq \pi$ and there is a non-negative integer k such that $\deg_G(x) = 2k - 1$,
(R2) $V(x) \geq V(y)$ for every valid edge $\{x, y\} \in E$,
(R3) $V(x) \leq V(y)$ for every valid edge $\{x, y\} \in E$, and
(V1) $|E_V^+(x)| \neq 0 \neq |E_V^-(x)| \neq |E_V^+(x)|$, and $\deg_G(x) = 2k$ for some integer k.

We shall prove that AlmostBalancedOdd is EOPL-complete. First, in Lemma 4.11, we show that a polynomial-time reduction from Degree-3 PotentialOdd to this problem, that is, we show the EOPL-hardness of AlmostBalancedOdd. After that, in Lemma 4.12, we show that AlmostBalancedOdd belongs to EOPL by constructing a polynomial-time reduction to PotentialImbalance. Of course, our proof implies that Degree-3 PotentialOdd is also EOPL-complete.

Lemma 4.11. Degree-3 PotentialOdd *is reducible to* AlmostBalancedOdd *in polynomial time.*

Proof. Each valid instance $\mathcal{I} = (n, m, N, V, \pi)$ of Degree-3 PotentialOdd is also a valid instance of AlmostBalancedOdd. We denote by $G(N, V) = (\Sigma^n, E)$ the implicit graph with potential produced by N and V. We can easily see that every odd-degree vertex and every local optimum vertex on $G(N, V)$ are a solution to $\mathcal{I}$. Therefore, it is sufficient to show that $\mathcal{I}$ has no violating solutions. We consider an even-degree vertex x that is not local optimum on $G(N, V)$. It satisfies that $|E_V^+(x)| \neq 0 \neq |E_V^-(x)|$. Notice that x has even degree and $\mathcal{I}$ is an instance of Degree-3 PotentialOdd, we have $|E_V^+(x)| + |E_V^-(x)| = 2$, and thus, $|E_V^+(x)| = 1 = |E_V^-(x)|$. Hence, the vertex x is a well-balanced vertex. □

Lemma 4.12. AlmostBalancedOdd *is reducible to* PotentialImbalance *in polynomial time.*

Proof. Assume that we are given an instance $\mathcal{I} = (n, m, d, N, V, \pi)$ of AlmostBalancedOdd. We denote by $G(N, V) = (\Sigma^n, E)$ the implicit graph with potential that is produced by N and V. For every vertex x in Σ^n, we define $S(x) := E_V^+(x)$ and $P(x) := E_V^-(x)$. Furthermore, we denote $\delta^+(x) = |E_V^+(x)|$ and $\delta^-(x) = |E_V^-(x)|$ for each vertex x in Σ^n. Then the vertex π satisfies that $\delta^+(\pi) \neq \delta^-(\pi)$ since π has odd degree on the implicit graph with potential $G(N, V)$. Thus, the tuple $\mathcal{J} = (n, m, d, S, P, V, \pi)$ is a valid instance of PotentialImbalance.

By definition, there are no violating solutions to $\mathcal{J}$. Therefore, we only obtain a solution x in Σ^n satisfying that $\delta^+(x) \neq \delta^-(x)$ and $x \neq \pi$. If $\delta^+(x) = 0$, then it holds that $V(x) \geq V(y)$ for every vertex valid edge $\{x, y\} \in E$. This implies that the vertex x is a local maximum solution of the original instance $\mathcal{I}$. Similarly, if $\delta^-(x) = 0$, then the vertex x is a local minimum solution of $\mathcal{I}$.

In the following, we consider the case where $\delta^+(x) \neq 0 \neq \delta^-(x)$. If x is an odd-degree vertex on the original instance, it is obviously a solution to $\mathcal{I}$. Otherwise, we obtain a violating solution to $\mathcal{I}$ because $|E_V^+(x)| \neq 0 \neq |E_V^-(x)| \neq |E_V^+(x)|$ and x is an even-degree vertex. □

Immediately, the following two theorems follow.

Theorem 4.13. Degree-3 PotentialOdd *is* EOPL*-complete.*

Theorem 4.14. AlmostBalancedOdd *is* EOPL*-complete.*

Finally, we show that AlmostBalancedOdd can be normalized.

Proposition 4.15. AlmostBalancedOdd *is normalizable.*

Proof. Let $\mathcal{I} = (n, m, d, N, V, \pi)$ is a valid instance of AlmostBalancedOdd. For each vertex $x \in \Sigma^n$, we add d additional vertices (x, i) where $i = 1, 2, \ldots, d$.

First, we define the new neighborhood function N' as follows. If a vertex $x \in \Sigma^n$ is a violating solution, $N'(x) = N(x) \cup \{(x, i); i = 1, 2, \ldots, \big||E_V^+(x)| - |E_V^+(x)|\big|\}$. Otherwise, we define $N'(x) = N(x)$. Furthermore, for each additional vertex (x, i), we define $N'(x, i) = \{x\}$ if $i \le \big||E_V^+(x)| - |E_V^+(x)|\big|$, otherwise $N'(x, i)$ is empty. Next, we define the new potential function V' as follow. For each original vertex $x \in \Sigma^n$, we define $V'(x) = V(x)$. For each additional vertex (x, i), we define

$$V'(x, i) = \begin{cases} V(x) + 1 & \text{if } |E_V^+(x)| < |E_V^-|, \\ V(x) - 1 & \text{if } |E_V^+(x)| > |E_V^-|, \\ V(x) & \text{if } |E_V^+(x)| = |E_V^-|. \end{cases}$$

Every violating solution x is adjacent to $\big||E_V^+(x)| - |E_V^+(x)|\big|$ additional vertices. Furthermore, the vertex x is well balanced from the definition of the new potential function V'. Therefore, the new instance has no violating solutions. Every additional vertex adjacent to x is an odd-degree vertex, and thus, we can extract a solution to the original instance efficiently. Moreover, each regular solution for the original instance is also a regular solution for the new instance. Hence, we get a normalization of AlmostBalancedOdd. □

Remark 3. Recall the definition of AlmostBalancedOdd. We require that every even-degree vertex which is not local optimum is well balanced. From this requirement, we can remove the non-local optimum condition, that is, we require that every even-degree vertex is well balanced. Notice that the solution set of the new search problem is same the original problem. Therefore, the complexity of this problem is also EOPL-complete. Note that every regular solution to this new variant is an odd-degree vertex.

5. Conclusions and open problems

We have studied the complexity of several variants of EndOfPotentialLine based on previous exciting work by Goldberg and Hollender [15]. Their technique can be extend to EndOfPotentialLine. We have shown that several variants of EndOfPotentialLine are also EOPL-complete. Our results imply that the classification of search problems based on EndOfPotentialLine is robust.

We have extended this argument to a similar problem on an undirected graph with potential. We have proved that the undirected variant of EndOfPotentialLine is generally not EOPL-complete. Specifically, Degree-3 PotentialOdd is EOPL-complete, but Degree-4 PotentialOdd is PPA ∩ PLS-complete. These facts leave an intriguing issue about the relationship between EOPL and PPA ∩ PLS. Are EOPL and PPA ∩ PLS separated? We conjecture that true.

There are some open problems left in this paper. The question worth considering is equivalence with PPA ∩ PLS and EOPL. Specifically, if PPA ∩ PLS = EOPL, then EOPL = CLS. Hence, that would resolve an important open question presented by Fearnley et al. [12]. Furthermore, we should also consider the relationship between PPA ∩ PLS and PPAD. Naturally, if PPAD ⊆ PPA ∩ PLS, then PPAD is a subset of PLS; this is an important open question that has been unsolved for a quarter-century. On the other hand, the containment of PPA ∩ PLS in PPAD implies that PPA ∩ PLS = PPAD ∩ PLS.

Another question worth considering is to verify whether UniqueEndOfPotentialLine can be normalized. The problem UniqueEndOfPotentialLine is introduced by Fearnley et.al. [12]. This problem considers an instance of EndOfPotentialLine that contains a *"single"* line which starts at the standard source.

CRediT authorship contribution statement

Takashi Ishizuka: Conceptualization, Methodology, Validation, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We sincerely thank the anonymous reviewers for many helpful comments that simplify some proofs.

References

[1] James Aisenberg, Maria Luisa Bonet, Sam Buss, 2-D Tucker is PPA complete, J. Comput. Syst. Sci. 108 (2020) 92–103.

[2] Yakov Babichenko, Aviad Rubinstein, Settling the complexity of Nash equilibrium in congestion games, in: Proceedings of the 53rd Annual ACM Symposium on Theory of Computing, 2021.
[3] Paul Beame, Stephen Cook, Jeff Edmonds, Russell Impagliazzo, Toniann Pitassi, The relative complexity of NP search problems, Ann. Pure Appl. Log. 57 (1) (1998) 3–19.
[4] Yang Cai, Ozan Candogan, Constantinos Daskalakis, Christos Papadimitriou, Zero-sum polymatrix games: a generalization of minmax, Math. Oper. Res. 41 (2) (2016) 648–655.
[5] Xi Chen, Xiaotie Deng, Shang-Hua Teng, Settling the complexity of computing two-player Nash equilibria, J. ACM 56 (3) (2009) 14:1–14:57.
[6] Constantinos Daskalakis, Paul W. Goldberg, Christos H. Papadimitriou, The complexity of computing a Nash equilibrium, SIAM J. Comput. 39 (1) (2009) 195–259.
[7] Constantinos Daskalakis, Christos H. Papadimitriou, Continuous local search, in: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, 2011, pp. 790–804.
[8] Constantinos Daskalakis, Christos Tzamos, Manolis Zampetakis, A converse to Banach's fixed point theorem and its CLS-completeness, in: Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, 2018, pp. 44–50.
[9] Xiaotie Deng, Zhe Feng, Rucha Kulkarni, Octahedral Tucker is PPA-complete, in: Electronic Colloquium on Computational Complexity, TR17-118, 2017.
[10] Alex Fabrikant, Christos H. Papadimitriou, Kunal Talwar, The complexity of pure Nash equilibria, in: Proceedings of the 36th Annual ACM Symposium on Theory of Computing, 2004, pp. 604–612.
[11] John Fearnly, Paul W. Goldberg, Alexandros Hollender, Rahul Savani, The complexity of gradient descent: $\mathtt{CLS} = \mathtt{PPAD} \cap \mathtt{PLS}$, in: Proceedings of the 53rd Annual ACM Symposium on Theory of Computing, 2021.
[12] John Fearnley, Spencer Gordon, Ruta Mehta, Rahul Savani, Unique end of potential line, J. Comput. Syst. Sci. 114 (2020) 1–35.
[13] Aris Filos-Ratsikas, Paul W. Goldberg, Consensus halving is PPA-complete, in: Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, 2018, pp. 51–64.
[14] Aris Filos-Ratsikas, Paul W. Goldberg, The complexity of splitting necklaces and bisecting ham sandwiches, in: Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, 2019, pp. 638–649.
[15] Paul W. Goldberg, Alexandros Hollender, The Hairy ball problem is PPAD-complete, in: Proceedings of the 46th International Colloquium on Automata, Languages, and Programming, vol. 132, 2019, pp. 65:1–65:14.
[16] Paul W. Goldberg, Christos H. Papadimitriou, TFNP: an update, in: Algorithms and Complexity, 2017, pp. 3–9.
[17] Paul W. Goldberg, Christos H. Papadimitriou, Towards a unified complexity theory of total functions, J. Comput. Syst. Sci. 94 (2018) 167–192.
[18] Alexandros Hollender, Paul W. Goldberg, The complexity of multi-source variants of the end-of-line problem, and the concise mutilated chessboard, in: Electronic Colloquium on Computational Complexity, TR18–120, 2018.
[19] Pavel Hubáček, Eylon Yogev, Hardness of continuous local search: query complexity and cryptographic lower bounds, in: Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, 2017, pp. 1352–1371.
[20] David S. Johnson, Christos H. Papadimitriou, Mihalis Yannakakis, How easy is local search?, J. Comput. Syst. Sci. 37 (1) (1988) 79–100.
[21] Nimrod Megiddo, Christos H. Papadimitriou, On total functions, existence theorems and computational complexity, Theor. Comput. Sci. 81 (2) (1991) 317–324.
[22] Christos H. Papadimitriou, On the complexity of the parity argument and other inefficient proofs of existence, J. Comput. Syst. Sci. 48 (3) (1994) 498–532.