CLS: New Problems and Completeness

John Fearnley[1], Spencer Gordon[2], Ruta Mehta[2], and Rahul Savani[1]

1 **University of Liverpool, Liverpool, UK**
{john.fearnley, rahul.savani}@liverpool.ac.uk
2 **University of Illinois at Urbana-Champaign, Urbana IL, USA**
rutamehta@cs.illinois.edu, slgordo2@illinois.edu

Abstract

The complexity class CLS was introduced by Daskalakis and Papadimitriou in [9] with the goal of capturing the complexity of some well-known problems in PPAD ∩ PLS that have resisted, in some cases for decades, attempts to put them in polynomial time. No complete problem was known for CLS, and in [9], the problems Contraction, i.e., the problem of finding an approximate fixpoint of a contraction map, and P-LCP, i.e., the problem of solving a P-matrix Linear Complementarity Problem, were identified as prime candidates.

First, we present a new CLS-complete problem MetametricContraction, which is closely related to the Contraction. Second, we introduce EndOfPotentialLine, which captures aspects of PPAD and PLS directly via a monotonic directed path, and show that EndOfPotentialLine is in CLS via a two-way reduction to EndOfMeteredLine. The latter was defined in [18] to keep track of how far a vertex is on the PPAD path via a restricted potential function. Third, we reduce P-LCP to EndOfPotentialLine, thus making EndOfPotentialLine and EndOfMeteredLine at least as likely to be hard for CLS as P-LCP. This last result leverages the monotonic structure of Lemke paths for P-LCP problems, making EndOfPotentialLine a likely candidate to capture the exact complexity of P-LCP; we note that the structure of Lemke-Howson paths for finding a Nash equilibrium in a two-player game very directly motivated the definition of the complexity class PPAD, which eventually ended up capturing this problem's complexity exactly.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases CLS, PPAD, PLS, Contraction Map, P-matrix LCP

1 Introduction

The complexity class TFNP, which stands for total function problems in NP, contains search problems that are guaranteed to have a solution, and whose solutions can be verified in polynomial time [26]. While TFNP is a semantically defined complexity class and is thus unlikely to contain complete problems, a number of syntactically defined subclasses of TFNP have proven very successful at capturing the complexity of total search problems. For example, the complexity class PPAD, introduced in [29] to capture the difficulty of search problems that are guaranteed total by a parity argument, attracted intense attention in the past decade culminating in a series of papers showing that the problem of computing a Nash-equilibrium in two-player games is PPAD-complete [4, 8]. There are no known polynomial-time algorithms for PPAD-complete problems, and recent work suggests that no such algorithms are likely to exist [1, 15]. The class of problems that can be solved by local search (in perhaps exponentially-many steps), PLS, has also attracted much interest since it was introduced in [19], and looks similarly unlikely to have polynomial-time algorithms. Examples of problems that are complete for PLS include the problem of computing a pure Nash equilibrium in a congestion game [11] and computing a locally optimal max cut [31].

If a problem lies in both PPAD and PLS then it is unlikely to be complete for either class, since this would imply a extremely surprising containment of one class in the other. Motivated by the existence of several total function problems in PPAD $\cap$ PLS that have resisted researchers attempts to design polynomial-time algorithms, in their 2011 paper [9], Daskalakis and Papadimitriou introduced the class CLS, a syntactically defined subclass of PPAD $\cap$ PLS. CLS is intended to capture the class of optimization problems over a continuous domain in which a continuous potential function is being minimized and the optimization algorithm has access to an continuous improvement function. Daskalakis and Papadimitriou showed that many classical problems of unknown complexity were shown to be in CLS including the problem of solving a simple stochastic game, the more general problem of solving a Linear Complementarity Problem with a P-matrix, and the problem of finding an approximate fixpoint to a contraction map. Moreover, CLS is the smallest known subclass of TFNP and hardness results for it imply hardness results for PPAD and PLS simultaneously.

Recent work by Hubáček and Yogev [18] proved lower bounds for CLS. They introduced a problem known as EndOfMeteredLine which they showed was in CLS, and for which they proved a query complexity lower bound of $\Omega(2^{n/2}/\sqrt{n})$ and hardness under the assumption that there were one-way permutations and indinstinguishability obfuscators for problems in $\mathsf{P}_{/\mathsf{poly}}$. Another recent result showed that the search version of the Colorful Carathéodory Theorem is in PPAD $\cap$ PLS, and left open whether the problem is also in CLS [27].

Unfortunately CLS is not particularly well-understood, and a glaring deficiency is the current lack of any complete problem for the class. In their original paper, Daskalakis and Papadimitriou suggested two natural candidates for complete problems for CLS, ContractionMap and P-LCP, and this remains an open problem. Another motivation for studying these two problems is that the problems of solving Condon's simple stochastic games can be reduced to each of them (separately) in polynomial time and, in turn, there is sequence of polynomial-time reductions from parity games to mean-payoff games to discounted games to simple stochastic games [16, 17, 21, 30, 33]. The complexity of solving these problems is unresolved and has received much attention over many years (see, for example, [2, 6, 12, 13, 20, 33]). In a recent breakthrough, a quasi-polynomial time algorithm for parity games was presented [3]. For mean-payoff, discounted, and simple stochastic games, the best-known algorithms run in subexponential time [24]. The existence of a polynomial time algorithm for solving any of these games would be a major breakthrough. For ContractionMap and P-LCP no subexponential time algorithms are known, and providing such algorithms would be a major breakthrough. As the most general of these problems, and thus most likely to be CLS-hard, we study ContractionMap and P-LCP.

Our contribution. We make progress towards settling the complexity of both of these problems. In the problem ContractionMap, as defined in [9], we are asked to find an approximate fixed point of a function f that is purported to contracting with respect to a metric induced by a norm (where the choice of norm does not matter but is not part of the input), or to give a violation of the contraction property for f. We introduce a problem, MetametricContraction, that allows the specification of a purpoted meta-metric d as part of the input of the problem, along with the function f. We are asked to either find an approximate fixed point of f, a violation of the contraction property for f with respect to d, a violation of the Lipschitz continuity of f or d, or a witness that d violates the meta-metric properties. We show that MetametricContraction is CLS-complete, thus identifying a first natural CLS-complete problem. We note that, contemporaneously and independently of our work, Daskalakis, Tzamos, and Zampetakis [10] have defined the problem MetricBanach and shown it is in CLS-complete. Their CLS-hardness reduction produces a

metric and is thus stronger than our CLS-hardness result for MetametricContraction. We discuss both results in more detail in Section 3.

Our second result is to show that P-LCP can be reduced to EndOfMeteredLine. The EndOfMeteredLine problem was introduced to capture problems that have a PPAD directed path structure while that also allow us to keep count of exactly how far the vertex is from the start of the path. In a sense, this may seem rather unnatural, as many common problems do not seem to have this property. In particular, while the P-LCP problem has a natural measure of progress towards a solution given by Lemke's algorithm, this is given in the form of a potential function, rather than an exact measure of the number of steps from the beginning of the algorithm.

To address this, we introduce a new problem EndOfPotentialLine which captures problems with a PPAD path structure that also allow have a potential function that decreases along this path. It is straightforward to show that EndOfPotentialLine is more general than EndOfMeteredLine. However, despite its generality, we are also able to show that EndOfPotentialLine can be reduced to EndOfMeteredLine in polynomial time, and so the two problems are equivalent under polynomial time reductions. We show that P-LCP can be reduced to EndOfPotentialLine, which provides an alternative proof that P-LCP is in CLS.

We believe that the EndOfPotentialLine problem is of independent interest, as it naturally unifies the circuit-based view of PPAD and of PLS, and is defined in the spirit of the canonical definitions of PPAD and PLS. There are two obvious lines for further research. Given the reduction we provide, EndOfPotentialLine and EndOfMeteredLine, are more likely candidates for CLS-hardness than P-LCP. Alternatively, one could attempt to reduce EndOfPotentialLine to P-LCP, thereby showing that that P-LCP is complete for the complexity class defined by these two problems, and in doing so finally resolve the long-standing open problem of the complexity of P-LCP. We note that, in the case of finding a Nash equilibrium of a two-player game, which we now know is PPAD-complete [4, 8], the definition of PPAD was inspired by the path structure of the Lemke-Howson algorithm, as our definition of EndOfPotentialLine is directly inspired by the path structure of Lemke paths for P-matrix LCPs.

2 Preliminaries

In this section, we define polynomial-time reductions between total search problems and the complexity class CLS.

▶ **Definition 1.** For total functions problems, a (polynomial-time) reduction from problem A to problem B is a pair of polynomial-time functions (f, g), such that f maps an instance x of A to an instance $f(x)$ of B, and g maps any solution y of $f(x)$ to a solution $g(y)$ of x.

Following [9], we define the complexity class CLS as the class of problems that are reducible to the following problem ContinuousLocalOpt.

▶ **Definition 2** (ContinuousLocalOpt [9]). Given two arithmetic circuits computing functions $f : [0, 1]^3 \to [0, 1]^3$ and $p : [0, 1]^3 \to [0, 1]$ and parameters $\epsilon, \lambda > 0$, find either:

(C1) a point $x \in [0, 1]^3$ such that $p(x) \leq p(f(x)) - \epsilon$ or
(C2) a pair of points $x, y \in [0, 1]^3$ satisfying either
  (C2a) $\|f(x) - f(y)\| > \lambda \|x - y\|$ or
  (C2b) $\|p(x) - p(y)\| > \lambda \|x - y\|$.

In Definition 2, p should be thought of as a *potential* function, and f as a *neighbourhood* function that gives a candidate solution with better potential if one exists. Both of these functions are purported to be Lipschitz continuous. A solution to the problem is either an approximate potential minimizer or a witness for a violation of Lipschitz continuity.

▶ **Definition 3** (Contraction [9]). We are given as input an arithmetic circuit computing $f : [0,1]^3 \to [0,1]^3$, a choice of norm $\|\cdot\|$, constants $\epsilon, c \in (0,1)$, and $\delta > 0$, and we are promised that f is c-contracting w.r.t. $\|\cdot\|$. The goal is to find

(CM1) a point $x \in [0,1]^3$ such that $d(f(x), x) \leq \delta$,
(CM2) or two points $x, y \in [0,1]^3$ such that $\|f(x) - f(y)\| \,/\, \|x - y\| > c$.

In other words, the problem asks either for an approximate fixed point of f or a violation of contraction. As shown in [9], Contraction is easily seen to be in CLS by creating instances of ContinuousLocalOpt with $p(x) = \|f(x) - x\|$, f remains as f, Lipschitz constant $\lambda = c + 1$, and $\epsilon = (1 - c)\delta$.

3 MetametricContraction is CLS-Complete

In this section, we define MetametricContraction and show that it is CLS-complete. In a *meta-metric*, all the requirements of a metric are satisfied except that the distance between identical points is not necessarily zero. The requirements for d to be a meta-metric are given in the following definition.

▶ **Definition 4** (Meta-metric). Let $\mathcal{D}$ be a set and $d : \mathcal{D}^2 \mapsto \mathbb{R}$ a function such that:

1. $d(x, y) \geq 0$;
2. $d(x, y) = 0$ implies $x = y$ (but, unlike for a metric, the converse is not required);
3. $d(x, y) = d(y, x)$;
4. $d(x, z) \leq d(x, y) + d(y, z)$.

Then d is a meta-metric on $\mathcal{D}$.

The problem Contraction, as defined in [9], was inspired by Banach's fixed point theorem, where the contraction can be with respect to any metric. In [9], for Contraction the assumed metric was any metric induced by a norm. The choice of this norm (and thus metric) was considered part of the definition of the problem, rather than part of the problem input. In the following definition of MetametricContraction, the contraction is with respect to a meta-metric, rather than a metric, and this meta-metric is given as part of the input of the problem.

▶ **Definition 5** (MetametricContraction). We are given as input an arithmetic circuit computing $f : [0,1]^3 \to [0,1]^3$, an arithmetic circuit computing a meta-metric $d : [0,1]^3 \times [0,1]^3 \to [0,1]$, some p-norm $\|\cdot\|_r$ and constants $\epsilon, c \in (0,1)$ and $\delta > 0$, and we are promised that f is c-contracting with respect to d, and λ-continuous with respect to $\|\cdot\|$, and that d is γ-continuous with respect to $\|\cdot\|$. The goal is to find

(M1) a point $x \in [0,1]^3$ such that $d(f(x), x) \leq \epsilon$,
(M2) or two points $x, y \in [0,1]^3$ such that

- (M2a) $d(f(x), f(y))/d(x, y) > c$,
- (M2b) $\|d(x, y) - d(x', y')\| \,/\, \|(x, y) - (x', y')\| > \delta$, or
- (M2c) $\|f(x) - f(y)\| \,/\, \|x - y\| > \lambda$.

(M3) points x, y, or x, y, z in $[0,1]^3$ that witness a violation of one of the four defining properties of a meta-metric (Definition 4).

▶ **Definition 6** (GeneralContraction)**.** The definition is identical to that of Definition 5 identical except for the fact that solutions of type (M3) are not allowed.

So, while MetametricContraction allows violations of d being a meta-metric as solutions, GeneralContraction does not.

▶ **Theorem 7.** GeneralContraction *is in* CLS.

Proof. Given an instance $X = (f, d, \epsilon, c, \lambda, \delta)$ of GeneralContraction, we set $p(x) \triangleq d(f(x), x)$. Then our ContinuousLocalOpt instance is the following:

$$Y = (f, p, \lambda' \triangleq (\lambda + 1)\delta, \epsilon' \triangleq (1 - c)\epsilon).$$

Now consider any solution to Y. If our solution is of type (C1), a point x such that $p(f(x)) > p(x) - \epsilon'$, then we have $d(f(f(x)), f(x)) > d(f(x), x) - (1 - c)\epsilon$, and either $d(f(x), x) \leq \epsilon$, in which case x is a solution for X, or $d(f(x), x) > \epsilon$. In the latter case, we can divide on both sides to get

$$\frac{d(f(f(x)), f(x))}{d(f(x), x)} > 1 - \frac{(1 - c)\epsilon}{d(f(x), x)} \geq 1 - (1 - c) = c,$$

giving us a violation of the claimed contraction factor of c, and a solution of type (M2a).

If our solution is a pair of points x, y of type (C2a) satisfying $\|f(x) - f(y)\| / \|x - y\| > \lambda' \geq \lambda$, then this gives a violation of the λ-continuity of f. If instead x, y are of type (C2b) so that $\|p(x) - p(y)\| / \|x - y\| > \lambda'$, then we have

$$|d(f(x), x) - d(f(y), y)| = |p(x) - p(y)| > (\lambda + 1)\delta \|x - y\|.$$

We now observe that if

$$|d(f(x), x) - d(f(y), y)| \leq \delta(\|f(x) - f(y)\| + \|x - y\|) \quad \text{and} \, \|f(x) - f(y)\| / \|x - y\| \leq \lambda,$$

then we would have

$$|d(f(x), x) - d(f(y), y)| \leq \delta(\|f(x) - f(y)\| + \|x - y\|) \leq (\lambda + 1)\delta \|x - y\|,$$

which contradicts the above inequality, so either the δ continuity of d must be violated giving a solution to X of type (M2b) or the λ continuity of f must be violated giving a solution of type (M2c). Thus we have shown that GeneralContraction is in CLS. ◀

Now that we have shown that GeneralContraction is total, we note that since the solutions of GeneralContraction are a subset of those for MetametricContraction, we have the following.

▶ **Observation 8.** MetametricContraction can be reduced in polynomial-time to GeneralContraction.

Thus, by Theorem 7, we have that MetametricContraction is in CLS. Next, we show that MetametricContraction is CLS-hard by a reduction from the canonical CLS-complete problem ContinuousLocalOpt to an instance of MetametricContraction. By Observation 8, we then also have that GeneralContraction is CLS-hard.

▶ **Theorem 9.** MetametricContraction *is* CLS*-hard.*

Proof. Given an instance $X = (f, p, \epsilon, \lambda)$ of ContinuousLocalOpt, we construct a meta-metric $d(x, y) = p(x)+p(y)+1$. Since p is non-negative, d is non-negative, and by construction, d is symmetric and satisfies the triangle inequality. Finally, $d(x, y) > 0$ for all choices of x and y so d is a valid meta-metric (Definition 4) Furthermore, if p is λ-continuous with respect to the given p-norm $\|\cdot\|_r$, then d is $(2^{1/r-1}\lambda)$-continuous with respect to $\|\cdot\|_r$. For clarity, in the below proof we'll omit the subscript r when writing the norm of an expression. To see this we observe that $x, x', y, y' \in [0, 1]^n$, we have $\|p(x) - p(x')\| / \|x - x'\| \leq \lambda$ and $\|p(y) - p(y')\| / \|y - y'\| \leq \lambda$, so

$$\frac{\|d(x,y) - d(x',y')\|}{\|(x,y) - (x',y')\|} = \frac{\|p(x) - p(x') + p(y) - p(y') + 1 - 1\|}{\|(x,y) - (x',y')\|} \leq \frac{\lambda \|x - x'\| + \lambda \|y - y'\|}{\|(x,y) - (x',y')\|}$$
$$\leq \frac{\lambda \|x - x'\| + \lambda \|y - y'\|}{2^{1-1/r}(\|x - x'\| + \|y - y'\|)} \leq 2^{1/r-1}\lambda.$$

We'll output an instance $Y = (f, d, \epsilon' = \epsilon, c = 1 - \epsilon/4, \delta = \lambda, \lambda' = 2^{1/r-1}\lambda)$.

Now we consider solutions for the instance Y and show that they correspond to solutions for our input instance X. First, we consider a solution of type (M1), a point $x \in [0, 1]^3$ such that $d(f(x), x) \leq \epsilon' = \epsilon$. We have $p(f(x)) + p(x) + 1 \leq \epsilon$, but this can't happen since $\epsilon < 1$ and p is non-negative, so solutions of this type cannot exist.

Now consider a solution that is a pair of points $x, y \in [0, 1]^3$ satisfying one of the conditions in (M2). If the solution is of type (M2a), we have $d(f(x), f(y)) > cd(x, y)$, and by our choice of c this is exactly

$$\frac{d(f(x), f(y))}{d(x, y)} > (1 - \epsilon/4)$$

and

$$p(f(x)) + p(f(y)) + 1 > (1 - \epsilon/4)(p(x) + p(y) + 1)$$
$$\geq p(x) + p(y) - 3\epsilon/4$$

so either $p(f(x)) > p(x) - \epsilon$ or $p(f(y)) > p(y) - \epsilon$, and one of x or y must be a fixpoint solution to our input instance. Solutions of type (M2b) or (M2c) immediately give us violations of the λ-continuity of f, and thus solutions to X.

This completes the proof that MetametricContraction is CLS-hard. ◀

So combining these results we have the following.

▶ **Theorem 10.** MetametricContraction *and* GeneralContraction *are* CLS*-complete.*

Finally, as mentioned in the introduction, we note the following. Contemporaneously and independently of our work, Daskalakis, Tzamos, and Zampetakis [10] defined the problem MetricBanach, which is like MetametricContraction except that it requires a metric, as opposed to a meta-metric. They show that MetricBanach is CLS-complete. Since every metric is a meta-metric, MetricBanach can be trivially reduced in polynomial-time to MetametricContraction. Thus, their CLS-hardness result is stronger than our Theorem 9. The containment of MetricBanach in CLS is implied by the containment of MetametricContraction in CLS. To prove that MetametricContraction is in CLS, we first reduce to GeneralContraction, which we then show is in CLS. Likewise, the proof in [10] that MetricBanach is in CLS works even when violations of the metric properties are not required as solutions, so they, like us, actually show that GeneralContraction is in CLS.

4 EndOfMeteredLine to EndOfPotentialLine and Back

In this section, we define a new problem EndOfPotentialLine. Then, we design polynomial-time reductions from EndOfMeteredLine to EndOfPotentialLine, and from EndOfPotentialLine to EndOfMeteredLine, thereby showing that the two problems are polynomial-time equivalent. In Section 5, we reduce P-LCP to EndOfPotentialLine.

First we recall the definition of EndOfMeteredLine, which was first defined in [18]. It is close in spirit to the problem EndOfLine that is used to define PPAD [29].

▶ **Definition 11** (EndOfMeteredLine [18]). Given circuits $S, P : \{0,1\}^n \to \{0,1\}^n$, and $V : \{0,1\}^n \to \{0,\ldots,2^n\}$ such that $P(0^n) = 0^n \neq S(0^n)$ and $V(0^n) = 1$, find a string $\boldsymbol{x} \in \{0,1\}^n$ satisfying one of the following

(T1) either $S(P(\boldsymbol{x})) \neq \boldsymbol{x} \neq 0^n$ or $P(S(\boldsymbol{x})) \neq \boldsymbol{x}$,
(T2) $\boldsymbol{x} \neq 0^n, V(\boldsymbol{x}) = 1$,
(T3) either $V(\boldsymbol{x}) > 0$ and $V(S(\boldsymbol{x})) - V(\boldsymbol{x}) \neq 1$, or $V(\boldsymbol{x}) > 1$ and $V(\boldsymbol{x}) - V(P(\boldsymbol{x})) \neq 1$.

Intuitively, an EndOfMeteredLine is an EndOfLine instance that is also equipped with an "odometer" function. The circuits P and S implicitly define an exponentially large graph in which each vertex has degree at most 2, just as in EndOfLine, and condition T1 says that the end of every line (other than 0^n) is a solution. In particular, the string 0^n is guaranteed to be the end of a line, and so a solution can be found by following the line that starts at 0^n. The function V is intended to help with this, by giving the number of steps that a given string is from the start of the line. We have that $V(0^n) = 1$, and that V increases by exactly 1 for each step we make along the line. Conditions T2 and T3 enforce this by saying that any violation of the property is also a solution to the problem.

In EndOfMeteredLine, the requirement of incrementing V by exactly one as walk along the line is quite restrictive. We define a new problem, EndOfPotentialLine, which is similar in spirit to EndOfLine, but drops the requirement of always incrementing the potential by one as we move along the line.

▶ **Definition 12** (EndOfPotentialLine). Given Boolean circuits $S, P : \{0,1\}^n \to \{0,1\}^n$ such that $P(0^n) = 0^n \neq S(0^n)$ and a Boolean circuit $V : \{0,1\}^n \to \{0,1,\ldots,2^m - 1\}$ such that $V(0^n) = 0$ find one of the following:

(R1) A point $x \in \{0,1\}^n$ such that $S(P(x)) \neq x \neq 0^n$ or $P(S(x)) \neq x$.
(R2) A point $x \in \{0,1\}^n$ such that $x \neq S(x)$, $P(S(x)) = x$, and $V(S(x)) - V(x) \leq 0$.

The key difference here is that the function V is required to be strictly monotonically increasing as we walk along the line, but the amount that it increases in each step is not specified. At first glance, the definition of EndOfPotentialLine may seem more general and more likely to capture the whole class CLS. In fact, we will show that EndOfMeteredLine and EndOfPotentialLine are inter-reducible in polynomial-time.

▶ **Theorem 13.** EndOfMeteredLine *and* EndOfPotentialLine *are equivalent under polynomial-time reductions.*

As expected, the reduction from EndOfMeteredLine to EndOfPotentialLine is relatively easy. It requires handling the difference in potential at 0^n and vertices with potential zero that are not discarded directly as possible solutions in EndOfPotentialLine. We make the latter self loops, but that creates extra starts and ends of lines which need to be handled. Full details of the reduction with proofs are in Appendix A.

The reduction from EndOfPotentialLine to EndOfMeteredLine is involved, and appears in detail in Appendix B. Here the basic idea is to insert missing single increments in between by introducing new vertices along the original edges. To allow this we need to encode potential itself in the vertex description. If there is an edge from $\boldsymbol{u}$ to $\boldsymbol{u}'$ in the EndOfPotentialLine instance whose respective potentials are p and p' such that say $p < p'$ then we create edges $(u, p) \to (u, p+1) \to \ldots \to (u, p'-1) \to (u, p')$. However, this creates a lot of dummy vertices, namely those that never appear on any edge due to irrelevant potential values, i.e., in this example (u, π) with $\pi < p$ or $\pi \geq p'$. We make them self loops (not an end-of-line) with zero potential, and since non-end-of-line solutions of EndOfMeteredLine, namely $T2$ and $T3$, must have strictly positive potential, these will never create a solution of the EndOfMeteredLine instance.

In addition, a number of issues need to be handled with consistency: (a) a $T2$ type solution of EndOfMeteredLine may be neither at the end of any line nor be a potential violation in EndOfPotentialLine; we do extra (linear time) work to handle such solutions, (b) a $T3$ type potential violation may not be on a "valid" edge as required by EndOfPotentialLine. (c) "invalid" edges, (d) potential difference at the initial vertex 0^n, etc.

5 Reduction from P-LCP to EndOfPotentialLine

In this section we present a polynomial-time reduction from the P-matrix Linear Complementarity Problem (P-LCP) to EndOfPotentialLine. A Linear Complementarity Problem (LCP) is defined as follows. Now on by $[n]$ we mean set $\{1, \ldots, n\}$.

▶ **Definition 14** (LCP)**.** Given a matrix $M \in R^{d \times d}$ and a vector $\boldsymbol{q} \in \mathbb{R}^{d \times 1}$, find a vector $\boldsymbol{y} \in \mathbb{R}^{d \times 1}$ such that:

$$M\boldsymbol{y} \leq \boldsymbol{q}; \quad \boldsymbol{y} \geq 0; \quad y_i(\boldsymbol{q} - M\boldsymbol{y})_i = 0, \ \forall i \in [n]. \tag{1}$$

In general, an LCP may have no solution, and deciding whether one does is NP-complete [5]. If the matrix M is a P-matrix, as defined next, then the LCP $(M, \boldsymbol{q})$ has a unique solution for all $\boldsymbol{q} \in \mathbb{R}^{d \times 1}$.

▶ **Definition 15** (P-matrix)**.** A matrix $M \in \mathbb{R}^{d \times d}$ is called a P-matrix if every principle minor of M is positive, *i.e.,* for every subset $S \subseteq [d]$, the sub-matrix $N = [M_{i,j}]_{i \in S, j \in S}$ has strictly positive determinant.

In order to define a problem that takes all matrices M as input without a promise, Megiddo [25] defined P-LCP as the following problem (see also [26]).

▶ **Definition 16** (P-LCP)**.** Given a matrix $M \in \mathbb{R}^{d \times d}$ and a vector $\boldsymbol{q} \in \mathbb{R}^{d \times 1}$, either:

(Q1) Find vector $\boldsymbol{y} \in \mathbb{R}^{n \times 1}$ that satisfies (1)
(Q2) Produce a witness that M is a not a P-matrix, *i.e.,* find $S \subset [d]$ such that for submatrix $N = [M_{i,j}]_{i \in S, j \in S}$, $det(N) \leq 0$.

Later, Papadimitriou showed that P-LCP is in PPAD [29], and then Daskalakis and Papadimitrou showed that it is in CLS [9] (based on the potential reduction method in [22]). Designing a polynomial-time solution for the P-LCP problem has been open for decades, at least since the 1978 paper of Murty [28] that provided exponential-time examples for *complementary pivoting algorithms*, such as *Lemke's algorithm* [23], for P-matrix Linear Complementarity Problems. Murty's family of P-matrices were based on the Klee-Minty's cubes that had been used to give exponential-time examples for the simplex method, and

which inspired the research that led to polynomial-time algorithms for Linear Programming. No similar polynomial-time algorithms are known for P-LCP though.

Lemke's algorithm introduces an extra variable, say z, to the LCP polytope, and follows a path on the 1-skeleton of the new polytope (like the simplex method for linear programming) based on complementary pivot rule (details below). A general LCP need not have a solution, and thus Lemke's algorithm is not guaranteed to terminate with a solution. However, for P-matrix LCPs, Lemke's algorithm terminates. Indeed, if Lemke's algorithm does not terminate with a solution, it provides a witness that the matrix M is not a P-matrix. The structure of the path traced by Lemke's algorithm is crucial for our reduction, so let us first briefly describe the algorithm.

5.1 Lemke's Algorithm

The explanation of Lemke's algorithm in this section is taken from [14]. The problem is interesting only when $\boldsymbol{q} \not\geq 0$, since otherwise $\boldsymbol{y} = 0$ is a trivial solution. Let us introduce slack variables $\boldsymbol{s}$ to obtain the following equivalent formulation:

$$M\boldsymbol{y} + \boldsymbol{s} = \boldsymbol{q}, \quad \boldsymbol{y} \geq 0, \quad \boldsymbol{s} \geq 0 \quad \text{and} \quad y_i s_i = 0, \ \forall i \in [d]. \tag{2}$$

Let $\mathcal{Q}$ be the polyhedron in $2d$ dimensional space defined by the first three conditions; we will assume that $\mathcal{Q}$ is non-degenerate (just for simplicity of exposition; this will not matter for our reduction). Under this condition, any solution to (2) will be a vertex of $\mathcal{Q}$, since it must satisfy $2d$ equalities. Note that the set of solutions may be disconnected. An ingenious idea of Lemke was to introduce a new variable and consider the system:

$$M\boldsymbol{y} + \boldsymbol{s} - z\mathbf{1} = \boldsymbol{q}, \quad \boldsymbol{y} \geq 0, \quad \boldsymbol{s} \geq 0, \quad z \geq 0 \quad \text{and} \quad y_i s_i = 0, \ \forall i \in [d]. \tag{3}$$

The next lemma follows by construction of (3).

▶ **Lemma 17.** *Given* $(M, \boldsymbol{q})$, $(\boldsymbol{y}, \boldsymbol{s}, z)$ *satisfies* (3) *with* $z = 0$ *iff* $\boldsymbol{y}$ *satisfies* (1).

Let $\mathcal{P}$ be the polyhedron in $2d + 1$ dimensional space defined by the first four conditions of (3), i.e.,

$$\mathcal{P} = \{(\boldsymbol{y}, \boldsymbol{s}, z) \mid M\boldsymbol{y} + \boldsymbol{s} - z\mathbf{1} = \boldsymbol{q}, \quad \boldsymbol{y} \geq 0, \quad \boldsymbol{s} \geq 0, \quad z \geq 0\}; \tag{4}$$

we will assume that $\mathcal{P}$ is *non-degenerate*.

Since any solution to (3) must still satisfy $2d$ equalities in $\mathcal{P}$, the set of solutions, say S, will be a subset of the one-skeleton of $\mathcal{P}$, i.e., it will consist of edges and vertices of $\mathcal{P}$. Any solution to the original system (2) must satisfy the additional condition $z = 0$ and hence will be a vertex of $\mathcal{P}$.

Now S turns out to have some nice properties. Any point of S is *fully labeled* in the sense that for each i, $y_i = 0$ or $s_i = 0$. We will say that a point of S *has duplicate label* i if $y_i = 0$ and $s_i = 0$ are both satisfied at this point. Clearly, such a point will be a vertex of $\mathcal{P}$ and it will have only one duplicate label. Since there are exactly two ways of relaxing this duplicate label, this vertex must have exactly two edges of S incident at it. Clearly, a solution to the original system (i.e., satisfying $z = 0$) will be a vertex of $\mathcal{P}$ that does not have a duplicate label. On relaxing $z = 0$, we get the unique edge of S incident at this vertex.

As a result of these observations, we can conclude that S consists of paths and cycles. Of these paths, Lemke's algorithm explores a special one. An unbounded edge of S such that the vertex of $\mathcal{P}$ it is incident on has $z > 0$ is called a *ray*. Among the rays, one is special –

the one on which $\boldsymbol{y} = 0$. This is called the *primary ray* and the rest are called *secondary rays*. Now Lemke's algorithm explores, via pivoting, the path starting with the primary ray. This path must end either in a vertex satisfying $z = 0$, i.e., a solution to the original system, or a secondary ray. In the latter case, the algorithm is unsuccessful in finding a solution to the original system; in particular, the original system may not have a solution. We give the full pseudo-code for Lemke's algorithm in Appendix C.

5.2 Polynomial time reduction from P-LCP to EndOfPotentialLine

It is well known that if matrix M is a P-matrix (P-LCP), then z strictly decreases on the path traced by Lemke's algorithm [7]. Furthermore, by a result of Todd [32, Section 5], paths traced by complementary pivot rule can be locally oriented. Based on these two facts, we now derive a polynomial-time reduction from P-LCP to EndOfPotentialLine.

Let $\mathcal{I} = (M, \boldsymbol{q})$ be a given P-LCP instance, and let $\mathcal{L}$ be the length of the bit representation of M and $\boldsymbol{q}$. We will reduce $\mathcal{I}$ to an EndOfPotentialLine instance $\mathcal{E}$ in time $poly(\mathcal{L})$. According to Definition 12, the instance $\mathcal{E}$ is defined by its vertex set vert, and procedures S (successor), P (predecessor) and V (potential). Next we define each of these.

As discussed in Section 5.1 the linear constraints of (3) on which Lemke's algorithm operates forms a polyhedron $\mathcal{P}$ given in (4). We assume that $\mathcal{P}$ is non-degenerate. This is without loss of generality since, a typical way to ensure this is by perturbing $\boldsymbol{q}$ so that configurations of solution vertices remain unchanged [7], and since M is unchanged the LCP is still P-LCP.

Lemke's algorithm traces a path on feasible points of (3) which is on 1-skeleton of $\mathcal{P}$ starting at $(\boldsymbol{y}^0, \boldsymbol{s}^0, z^0)$, where:

$$\boldsymbol{y}^0 = 0, \qquad z^0 = |\min_{i \in [d]} q_i|, \qquad \boldsymbol{s}^0 = \boldsymbol{q} + z\mathbf{1} \tag{5}$$

We want to capture vertex solutions of (3) as vertices in EndOfPotentialLine instance $\mathcal{E}$. To differentiate we will sometimes call the latter *configurations*. Vertex solutions of (3) are exactly the vertices of polyhedron $\mathcal{P}$ with either $y_i = 0$ or $s_i = 0$ for each $i \in [d]$. Vertices of (3) with $z = 0$ are our final solutions (Lemma 17). While each of its *non-solution* vertex has a duplicate label. Thus, a vertex of this path can be uniquely identified by which of $y_i = 0$ and $s_i = 0$ hold for each i and its duplicate label. This gives us a representation for vertices in the EndOfPotentialLine instance $\mathcal{E}$.

EndOfPotentialLine Instance $\mathcal{E}$.

- Vertex set $\mathsf{vert} = \{0,1\}^n$ where $n = 2d$.
- Procedures S and P as defined in Tables 1 and 3 respectively
- Potential function $V : \mathsf{vert} \to \{0, 1, \ldots, 2^m - 1\}$ defined in Table 2 for $m = \lceil ln(2\Delta^3) \rceil$, where
$$\Delta = (n! \cdot I_{max}^{2d+1}) + 1$$
and $I_{max} = \max\{\max_{i,j \in [d]} M(i,j),\ \max_{i \in [d]} |q_i|\}$.

For any vertex $\boldsymbol{u} \in \mathsf{vert}$, the first d bits of $\boldsymbol{u}$ represent which of the two inequalities, namely $y_i \geq 0$ and $s_i \geq 0$, are tight for each $i \in [d]$. A valid setting of the second set of d bits will have at most one non-zero bit – if none is one then $z = 0$, otherwise the location of one bit indicates the duplicate label. Thus, there are many invalid configurations, namely those with more than one non-zero bit in the second set of d bits. These are dummies that we will handle separately, and we define a procedure IsValid to identify non-dummy vertices in Table

4 (in Appendix D.1). To go between "valid" vertices of $\mathcal{E}$ and corresponding vertices of the Lemke polytope $\mathcal{P}$ of LCP $\mathcal{I}$, we define procedures EtoI and ItoE in Table 5 (in Appendix D.1). By construction of IsValid, EtoI and ItoE, the next lemma follows. All the missing proofs of this section may be found in Appendix D.

▶ **Lemma 18.** *If IsValid*$(\boldsymbol{u}) = 1$ *then* $\boldsymbol{u} = ItoE(EtoI(\boldsymbol{u}))$*, and the corresponding vertex* $(\boldsymbol{y}, \boldsymbol{s}, z) \in EtoI(\boldsymbol{u})$ *of* $\mathcal{P}$ *is feasible in (3). If* $(\boldsymbol{y}, \boldsymbol{s}, z)$ *is a feasible vertex of (3) then* $\boldsymbol{u} =$ *ItoE*$(\boldsymbol{y}, \boldsymbol{s}, z)$ *is a valid configuration, i.e.,* *IsValid*$(\boldsymbol{u}) = 1$.

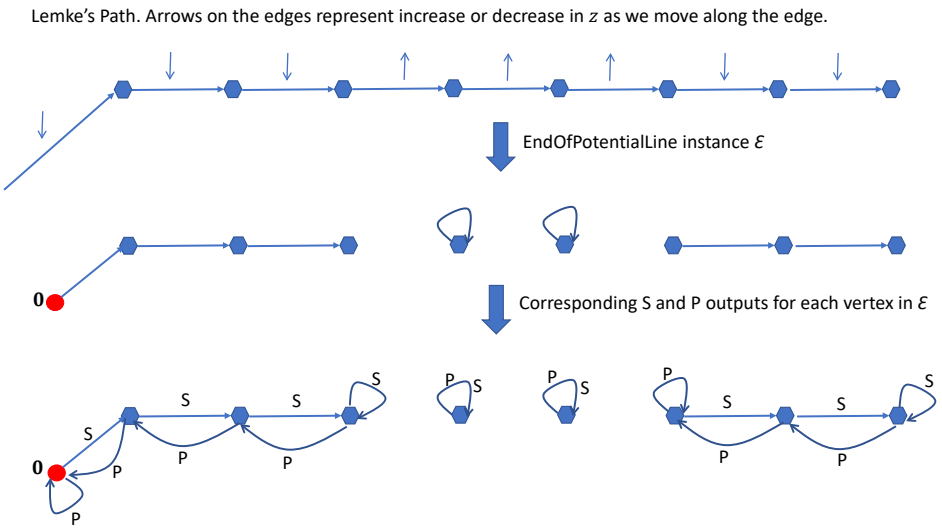

Figure 1 Construction of S and P for EndOfPotentialLine instance $\mathcal{E}$ from the Lemke path. The first path is the Lemke path and the arrows on its edges indicate whether the value of z increases or decreases along the edge. Note that the end or start of a path in $\mathcal{E}$, which is an intermediate vertex in Lemke path that has either decreased and then increased, or increased and then decreased in the value of z, is a violation of M being a P matrix [7], i.e., Q2 type solution of P-LCP.

The main idea behind procedures S and P, given in Tables 1 and 3 respectively, is the following (also see Figure 1): Make dummy configurations in vert to point to themselves with cycles of length one, so that they can never be solutions. The starting vertex $0^n \in$ vert points to the configuration that corresponds to the first vertex of the Lemke path, namely $\boldsymbol{u}^0 = \text{ItoE}(\boldsymbol{y}^0, \boldsymbol{s}^0, z^0)$. Precisely, $S(0^n) = \boldsymbol{u}^0$, $P(\boldsymbol{u}^0) = 0^n$ and $P(0^n) = 0^n$ (start of a path).

For the remaining cases, let $\boldsymbol{u} \in$ vert have corresponding representation $\boldsymbol{x} = (\boldsymbol{y}, \boldsymbol{s}, z) \in \mathcal{P}$, and suppose $\boldsymbol{x}$ has a duplicate label. As one traverses a Lemke path for a P-LCPs, the value of z monotonically decreases. So, for $S(\boldsymbol{u})$ we compute the adjacent vertex $\boldsymbol{x}' = (\boldsymbol{y}', \boldsymbol{s}', z')$ of $\boldsymbol{x}$ on Lemke path such that the edge goes from $\boldsymbol{x}$ to $\boldsymbol{x}'$, and if the $z' < z$, as expected, then we point $S(\boldsymbol{u})$ to configuration of $\boldsymbol{x}'$ namely $\text{ItoE}(\boldsymbol{x}')$. Otherwise, we let $S(\boldsymbol{u}) = \boldsymbol{u}$. Similarly, for $P(\boldsymbol{u})$, we find $\boldsymbol{x}'$ such that edge is from $\boldsymbol{x}'$ to $\boldsymbol{x}$, and then we let $P(\boldsymbol{u})$ be $\text{ItoE}(\boldsymbol{x}')$ if $z' > z$ as expected, otherwise $P(\boldsymbol{u}) = \boldsymbol{u}$.

For the case when $\boldsymbol{x}$ does not have a duplicate label, then we have $z = 0$. This is handled separately since such a vertex has exactly one incident edge on the Lemke path, namely the one obtained by relaxing $z = 0$. According to the direction of this edge, we do similar process as before. For example, if the edge goes from $\boldsymbol{x}$ to $\boldsymbol{x}'$, then, if $z' < z$, we set $S(\boldsymbol{u}) = \text{ItoE}(\boldsymbol{x}')$ else $S(\boldsymbol{u}) = \boldsymbol{u}$, and we always set $P(\boldsymbol{u}) = \boldsymbol{u}$. In case the edge goes from $\boldsymbol{x}'$ to $\boldsymbol{x}$, we always set $S(\boldsymbol{u}) = \boldsymbol{u}$, and we set $P(\boldsymbol{u})$ depending on whether or not $z' > z$.

Table 1 Successor Procedure $S(\boldsymbol{u})$

If IsValid$(\boldsymbol{u}) = 0$ **then Return** $\boldsymbol{u}$
If $\boldsymbol{u} = 0^n$ **then Return** ItoE$(\boldsymbol{y}^0, \boldsymbol{s}^0, z^0)$
$\boldsymbol{x} = (\boldsymbol{y}, \boldsymbol{s}, z) \leftarrow$ EtoI$(\boldsymbol{u})$
If $z = 0$ **then**
  $\boldsymbol{x}^1 \leftarrow$ vertex obtained by relaxing $z = 0$ at $\boldsymbol{x}$ in $\mathcal{P}$.
  If Todd [32] prescribes edge from $\boldsymbol{x}$ to $\boldsymbol{x}^1$
   then set $\boldsymbol{x}' \leftarrow \boldsymbol{x}^1$. **Else Return** $\boldsymbol{u}$
Else set $l \leftarrow$ duplicate label at $\boldsymbol{x}$
  $\boldsymbol{x}^1 \leftarrow$ vertex obtained by relaxing $y_l = 0$ at $\boldsymbol{x}$ in $\mathcal{P}$
  $\boldsymbol{x}^2 \leftarrow$ vertex obtained by relaxing $s_l = 0$ at $\boldsymbol{x}$ in $\mathcal{P}$
  If Todd [32] prescribes edge from $\boldsymbol{x}$ to $\boldsymbol{x}^1$
   then $\boldsymbol{x}' = \boldsymbol{x}^1$
  Else $\boldsymbol{x}' = \boldsymbol{x}^2$
Let $\boldsymbol{x}'$ be $(\boldsymbol{y}', \boldsymbol{s}', z')$.
If $z > z'$ **then Return** ItoE$(\boldsymbol{x}')$. **Else Return** $\boldsymbol{u}$.

Table 2 Potential Value $V(\boldsymbol{u})$

If IsValid$(\boldsymbol{u}) = 0$
 then Return 0
If $\boldsymbol{u} = 0^n$
 then Return 0
$(\boldsymbol{y}, \boldsymbol{s}, z) \leftarrow$ EtoI$(\boldsymbol{u})$
Return $\lfloor \Delta^2 * (\Delta - z) \rfloor$

The potential function V, formally defined in Table 2, gives a value of zero to dummy vertices and the starting vertex 0^n. To all other vertices, essentially it is $((z^0 - z) * \Delta^2) + 1$. Since value of z starts at z^0 and keeps decreasing on the Lemke path this value will keep increasing starting from zero at the starting vertex 0^n. Multiplication by Δ^2 will ensure that if $z_1 > z_2$ then the corresponding potential values will differ by at least one. This is because, since z_1 and z_2 are coordinates of two vertices of polytope $\mathcal{P}$, their maximum value is Δ and their denominator is also bounded above by Δ. Hence $z_1 - z_2 \leq 1/\Delta^2$ (Lemma 20).

To show correctness of the reduction we need to show two things: (i) All the procedures are well-defined and polynomial time. (ii) We can construct a solution of $\mathcal{I}$ from a solution of $\mathcal{E}$ in polynomial time.

Table 3 Predecessor Procedure $P(\boldsymbol{u})$

If IsValid$(\boldsymbol{u}) = 0$ **then Return** $\boldsymbol{u}$
If $\boldsymbol{u} = 0^n$ **then Return** $\boldsymbol{u}$
$(\boldsymbol{y}, \boldsymbol{s}, z) \leftarrow$ EtoI$(\boldsymbol{u})$
If $(\boldsymbol{y}, \boldsymbol{s}, z) = (\boldsymbol{y}^0, \boldsymbol{s}^0, z^0)$ **then Return** 0^n
If $z = 0$ **then**
 $\boldsymbol{x}^1 \leftarrow$ vertex obtained by relaxing $z = 0$ at $\boldsymbol{x}$ in $\mathcal{P}$.
 If Todd [32] prescribes edge from $\boldsymbol{x}^1$ to $\boldsymbol{x}$ **then** set $\boldsymbol{x}' \leftarrow \boldsymbol{x}^1$. **Else Return** $\boldsymbol{u}$
Else
 $l \leftarrow$ duplicate label at $\boldsymbol{x}$
 $\boldsymbol{x}^1 \leftarrow$ vertex obtained by relaxing $y_l = 0$ at $\boldsymbol{x}$ in $\mathcal{P}$
 $\boldsymbol{x}^2 \leftarrow$ vertex obtained by relaxing $s_l = 0$ at $\boldsymbol{x}$ in $\mathcal{P}$
 If Todd [32] prescribes edge from $\boldsymbol{x}^1$ to $\boldsymbol{x}$ **then** $\boldsymbol{x}' = \boldsymbol{x}^1$ **Else** $\boldsymbol{x}' = \boldsymbol{x}^2$
Let $\boldsymbol{x}'$ be $(\boldsymbol{y}', \boldsymbol{s}', z')$. **If** $z < z'$ **then Return** ItoE$(\boldsymbol{x}')$. **Else Return** $\boldsymbol{u}$.

▶ **Lemma 19.** *Functions P, S and V of instance $\mathcal{E}$ are well defined, making $\mathcal{E}$ a valid* EndOfPotentialLine *instance.*

There are two possible types of solutions of an EndOfPotentialLine instance. One

indicates the beginning or end of a line, and the other is a vertex with locally optimal potential (that does not point to itself). First we show that the latter case never arise. For this, we need the next lemma, which shows that potential differences in two adjacent configurations adheres to differences in the value of z at corresponding vertices.

▶ **Lemma 20.** *Let $\boldsymbol{u} \neq \boldsymbol{u}'$ be two valid configurations, i.e., IsValid$(\boldsymbol{u})$ = IsValid$(\boldsymbol{u}') = 1$, and let $(\boldsymbol{y}, \boldsymbol{s}, z)$ and $(\boldsymbol{y}', \boldsymbol{s}', z')$ be the corresponding vertices in $\mathcal{P}$. Then the following holds: (i) $V(\boldsymbol{u}) = V(\boldsymbol{u}')$ iff $z = z'$. (ii) $V(\boldsymbol{u}) > V(\boldsymbol{u}')$ iff $z < z'$.*

Using the above lemma, we will next show that instance $\mathcal{E}$ has no local maximizer.

▶ **Lemma 21.** *Let $\boldsymbol{u}, \boldsymbol{v} \in$ vert s.t. $\boldsymbol{u} \neq \boldsymbol{v}$, $\boldsymbol{v} = S(\boldsymbol{u})$, and $\boldsymbol{u} = P(\boldsymbol{v})$. Then $V(\boldsymbol{u}) < V(\boldsymbol{v})$.*

Proof. Let $\boldsymbol{x} = (\boldsymbol{y}, \boldsymbol{s}, z)$ and $\boldsymbol{x}' = (\boldsymbol{y}', \boldsymbol{s}', z')$ be the vertices in polyhedron $\mathcal{P}$ corresponding to $\boldsymbol{u}$ and $\boldsymbol{v}$ respectively. From the construction of $\boldsymbol{v} = S(\boldsymbol{u})$ implies that $z' < z$. Therefore, using Lemma 20 it follows that $V(\boldsymbol{v}) < V(\boldsymbol{u})$. ◀

Due to Lemma 21 the only type of solutions available in $\mathcal{E}$ is where $S(P(\boldsymbol{u})) \neq \boldsymbol{u}$ and $P(S(\boldsymbol{u})) \neq \boldsymbol{u}$. Next two lemmas shows how to construct solutions of $\mathcal{I}$ from these.

▶ **Lemma 22.** *Let $\boldsymbol{u} \in$ vert, $\boldsymbol{u} \neq 0^n$. If $P(S(\boldsymbol{u})) \neq \boldsymbol{u}$ or $S(P(\boldsymbol{u})) \neq \boldsymbol{u}$, then IsValid$(\boldsymbol{u}) = 1$, and for $(\boldsymbol{y}, \boldsymbol{s}, z)$ = EtoI$(\boldsymbol{u})$ if $z = 0$ then $\boldsymbol{y}$ is a Q1 type solution of* P-LCP *instance $\mathcal{I} = (M, \boldsymbol{q})$.*

Proof. By construction, if IsValid$(\boldsymbol{u}) = 0$, then $S(P(\boldsymbol{u})) = \boldsymbol{u}$ and $P(S(\boldsymbol{u})) = \boldsymbol{u}$, therefore IsValid$(\boldsymbol{u}) = 0$ when $\boldsymbol{u}$ has a predecessor or successor different from $\boldsymbol{u}$. Given this, from Lemma 18 we know that $(\boldsymbol{y}, \boldsymbol{s}, z)$ is a feasible vertex in (3). Therefore, if $z = 0$ then using Lemma 17 we have a solution of the LCP (1), *i.e.,* a type Q1 solution of our P-LCP instance $\mathcal{I} = (M, \boldsymbol{q})$. ◀

▶ **Lemma 23.** *Let $\boldsymbol{u} \in$ vert, $\boldsymbol{u} \neq 0^n$ such that $P(S(\boldsymbol{u})) \neq \boldsymbol{u}$ or $S(P(\boldsymbol{u})) \neq \boldsymbol{u}$, and let $\boldsymbol{x} = (\boldsymbol{y}, \boldsymbol{s}, z)$ = EtoI$(\boldsymbol{u})$. If $z \neq 0$ then $\boldsymbol{x}$ has a duplicate label, say l. And for directions σ_1 and σ_2 obtained by relaxing $y_l = 0$ and $s_l = 0$ respectively at $\boldsymbol{x}$, we have $\sigma_1(z) * \sigma_2(z) \geq 0$, where $\sigma_i(z)$ is the coordinate corresponding to z.*

Proof. From Lemma 22 we know that IsValid$(\boldsymbol{u}) = 1$, and therefore from Lemma 18, $\boldsymbol{x}$ is a feasible vertex in (3). From the last line of Tables 1 and 3 observe that $S(\boldsymbol{u})$ points to the configuration of vertex next to $\boldsymbol{x}$ on Lemke's path only if it has lower z value otherwise it gives back $\boldsymbol{u}$, and similarly $P(\boldsymbol{u})$ points to the previous only if value of z increases.

First consider the case when $P(S(\boldsymbol{u})) \neq \boldsymbol{u}$. Let $\boldsymbol{v} = S(\boldsymbol{u})$ and corresponding vertex in $\mathcal{P}$ be $(\boldsymbol{y}', \boldsymbol{s}', z')$ = EtoI$(\boldsymbol{v})$. If $\boldsymbol{v} \neq \boldsymbol{u}$, then from the above observation we know that $z' > z$, and in that case again by construction of P we will have $P(\boldsymbol{v}) = \boldsymbol{u}$, contradicting $P(S(\boldsymbol{u})) \neq \boldsymbol{u}$. Therefore, it must be the case that $\boldsymbol{v} = \boldsymbol{u}$. Since $z \neq 0$ this happens only when the next vertex on Lemke path after $\boldsymbol{x}$ has higher value of z (by above observation). As a consequence of $\boldsymbol{v} = \boldsymbol{u}$, we also have $P(\boldsymbol{u}) \neq \boldsymbol{u}$. By construction of P this implies for $(\boldsymbol{y}'', \boldsymbol{s}'', z'')$ = EtoI$(P(\boldsymbol{u}))$, $z'' > z$. Putting both together we get increase in z when we relax $y_l = 0$ as well as when we relax $s_l = 0$ at $\boldsymbol{x}$.

For the second case $S(P(\boldsymbol{u})) \neq \boldsymbol{u}$ similar argument gives that value of z decreases when we relax $y_l = 0$ as well as when we relax $s_l = 0$ at $\boldsymbol{x}$. The proof follows. ◀

Finally, we are ready to prove our main result of this section using Lemmas 21, 22 and 23. Together with Lemma 23, we will use the fact that on Lemke path z monotonically decreases if M is a P-matrix or else we get a witness that M is not a P-matrix [7].

▶ **Theorem 24.** P-LCP *reduces to* EndOfPotentialLine *in polynomial-time.*

Proof. Given an instance of $\mathcal{I} = (M, \boldsymbol{q})$ of P-LCP, where $M \in \mathbb{R}^{d\times d}$ and $\boldsymbol{q} \in \mathbb{R}^{d\times 1}$ reduce it to an instance $\mathcal{E}$ of EndOfPotentialLine as described above with vertex set $\mathsf{vert} = \{0,1\}^{2d}$ and procedures S, P and V as given in Table 1, 3, and 2 respectively.

Among solutions of EndOfPotentialLine instance $\mathcal{E}$, there is no local potential maximizer, i.e., $\boldsymbol{u} \neq \boldsymbol{v}$ such that $\boldsymbol{v} = S(\boldsymbol{u})$, $\boldsymbol{u} = P(\boldsymbol{v})$ and $V(\boldsymbol{u}) > V(\boldsymbol{v})$ due to Lemma 21. We get a solution $\boldsymbol{u} \neq 0$ such that either $S(P(\boldsymbol{u})) \neq \boldsymbol{u}$ or $P(S(\boldsymbol{u})) \neq \boldsymbol{u}$, then by Lemma 22 it is valid configuration and has a corresponding vertex $\boldsymbol{x} = (\boldsymbol{y}, \boldsymbol{s}, z)$ in $\mathcal{P}$. Again by Lemma 22 if $z = 0$ then $\boldsymbol{y}$ is a Q1 type solution of our P-LCP instance $\mathcal{I}$. On the other hand, if $z > 0$ then from Lemma 23 we get that on both the two adjacent edges to $\boldsymbol{x}$ on Lemke path the value of z either increases or deceases. This gives us a minor of M which is non-positive [7], i.e., a Q2 type solution of the P-LCP instance $\mathcal{I}$. ◀

References

1. Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a Nash equilibrium. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 1480–1498. IEEE, 2015.
2. Henrik Björklund, Sven Sandberg, and Sergei Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. In *International Symposium on Mathematical Foundations of Computer Science*, pages 673–685. Springer, 2004.
3. Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. STOC '17 (to appear); `https://www.comp.nus.edu.sg/~fstephan/paritygame.ps`.
4. Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibria. *Journal of the ACM (JACM)*, 56(3):14, 2009.
5. Sung-Jin Chung. NP-completeness of the linear complementarity problem. *Journal of Optimization Theory and Applications*, 60(3):393–399, 1989.
6. Anne Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, 1992.
7. Richard W Cottle, Jong-Shi Pang, and Richard E Stone. *The linear complementarity problem*. SIAM, 2009.
8. Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009.
9. Constantinos Daskalakis and Christos Papadimitriou. Continuous local search. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms (SODA)*, pages 790–804. Society for Industrial and Applied Mathematics, 2011.
10. Constantinos Daskalakis, Christos Tzamos, and Manolis Zampetakis. A Converse to Banach's Fixed Point Theorem and its CLS Completeness. *ArXiv e-prints*. `arXiv:1702.07339`.
11. Alex Fabrikant, Christos Papadimitriou, and Kunal Talwar. The complexity of pure Nash equilibria. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing (STOC)*, pages 604–612. ACM, 2004.
12. John Fearnley, Marcin Jurdziński, and Rahul Savani. Linear complementarity algorithms for infinite games. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 382–393. Springer, 2010.
13. John Fearnley and Rahul Savani. The complexity of all-switches strategy improvement. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 130–139. Society for Industrial and Applied Mathematics, 2016.
14. Jugal Garg, Ruta Mehta, Milind Sohoni, and Vijay V. Vazirani. A complementary pivot algorithm for market equilibrium under separable piecewise-linear concave utilities. *SIAM J. Comput.*, 44(6):1820–1847, 2015.
15. Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a Nash equilibrium. In *Annual Cryptology Conference*, pages 579–604. Springer, 2016.
16. Bernd Gärtner and Leo Rüst. Simple stochastic games and P-matrix generalized linear complementarity problems. In *International Symposium on Fundamentals of Computation Theory*, pages 209–220. Springer, 2005.
17. Thomas Dueholm Hansen and Rasmus Ibsen-Jensen. The complexity of interior point methods for solving discounted turn-based stochastic games. In *Conference on Computability in Europe*, pages 252–262. Springer, 2013.
18. Pavel Hubáček and Eylon Yogev. Hardness of continuous local search: Query complexity and cryptographic lower bounds. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1352–1371. SIAM, 2017.

19 David S Johnson, Christos H Papadimitriou, and Mihalis Yannakakis. How easy is local search? *Journal of computer and system sciences*, 37(1):79–100, 1988.

20 Marcin Jurdziński. Deciding the winner in parity games is in UP ∩ co-UP. *Information Processing Letters*, 68(3):119–124, 1998.

21 Marcin Jurdziński and Rahul Savani. A simple P-matrix linear complementarity problem for discounted games. In *Conference on Computability in Europe*, pages 283–293. Springer, 2008.

22 Masakazu Kojima, Nimrod Megiddo, and Yinyu Ye. An interior point potential reduction algorithm for the linear complementarity problem. *Mathematical Programming*, 54(1-3):267–279, 1992.

23 Carlton E Lemke. Bimatrix equilibrium points and mathematical programming. *Management science*, 11(7):681–689, 1965.

24 Walter Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and computation*, 117(1):151–155, 1995.

25 Nimrod Megiddo. *A note on the complexity of P-matrix LCP and computing an equilibrium.* IBM Thomas J. Watson Research Division, 1988.

26 Nimrod Megiddo and Christos H Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, 1991.

27 Frédéric Meunier, Wolfgang Mulzer, Pauline Sarrabezolles, and Yannik Stein. The rainbow at the end of the line: A PPAD formulation of the colorful Carathéodory theorem with applications. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1342–1351, 2017.

28 Katta G Murty. Computational complexity of complementary pivot methods. In *Complementarity and fixed point problems*, pages 61–73. Springer, 1978.

29 Christos H Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and system Sciences*, 48(3):498–532, 1994.

30 Anuj Puri. Theory of hybrid systems and discrete event systems. 1996.

31 Alejandro A Schäffer and Mihalis Yannakakis. Simple local search problems that are hard to solve. *SIAM journal on Computing*, 20(1):56–87, 1991.

32 Michael J Todd. Orientation in complementary pivot algorithms. *Mathematics of Operations Research*, 1(1):54–66, 1976.

33 Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1-2):343–359, 1996.

A EndOfMeteredLine to EndOfPotentialLine

Given an instance $\mathcal{I}$ of EndOfMeteredLine defined by circuits S, P and V on vertex set $\{0,1\}^n$ we are going to create an instance $\mathcal{I}'$ of EndOfPotentialLine with circuits S', P', and V' on vertex set $\{0,1\}^{(n+1)}$, i.e., we introduce one extra bit. This extra bit is essentially to take care of the difference in the value of potential at the starting point in EndOfMeteredLine and EndOfPotentialLine, namely 1 and 0 respectively.

Let $k = n+1$, then we create a potential function $V' : \{0,1\}^k \to \{0, \ldots, 2^k - 1\}$. The idea is to make 0^k the starting point with potential zero as required, and to make all other vertices with first bit 0 be dummy vertices with self loops. The real graph will be embedded in vertices with first bit 1, i.e., of type $(1, \boldsymbol{u})$. Here by $(b, \boldsymbol{u}) \in \{0,1\}^k$, where $b \in \{0,1\}$ and $\boldsymbol{u} \in \{0,1\}^n$, we mean a k length bit string with first bit set to b and for each $i \in [2:k]$ bit i set to bit u_i.

Procedure $V'(b, \boldsymbol{u})$**:** If $b = 0$ then Return 0, otherwise Return $V(\boldsymbol{u})$.

Procedure $S'(b, \boldsymbol{u})$**:**

1. If $(b, \boldsymbol{u}) = 0^k$ then Return $(1, 0^n)$
2. If $b = 0$ and $\boldsymbol{u} \neq 0^n$ then Return $(b, \boldsymbol{u})$ (creating self loop for dummy vertices)
3. If $b = 1$ and $V(\boldsymbol{u}) = 0$ then Return $(b, \boldsymbol{u})$ (vertices with zero potentials have self loops)
4. If $b = 1$ and $V(\boldsymbol{u}) > 0$ then Return $(b, S(\boldsymbol{u}))$ (the rest follows S)

Procedure $P'(b, \boldsymbol{u})$**:**

1. If $(b, \boldsymbol{u}) = 0^k$ then Return $(b, \boldsymbol{u})$ (initial vertex points to itself in P').
2. If $b = 0$ and $\boldsymbol{u} \neq 0^n$ then Return $(b, \boldsymbol{u})$ (creating self loop for dummy vertices)
3. If $b = 1$ and $\boldsymbol{u} = 0^n$ then Return 0^k (to make $(0, 0^n) \to (1, 0^n)$ edge consistent)
4. If $b = 1$ and $V(\boldsymbol{u}) = 0$ then Return $(b, \boldsymbol{u})$ (vertices with zero potentials have self loops)
5. If $b = 1$ and $V(\boldsymbol{u}) > 0$ and $\boldsymbol{u} \neq 0^n$ then Return $(b, P(\boldsymbol{u}))$ (the rest follows P)

Valid solutions of EndOfMeteredLine of type T2 and T3 requires the potential to be strictly greater than zero, while solutions of EndOfPotentialLine may have zero potential. However, a solution of EndOfPotentialLine can not be a self loop, so we've added self-loops around vertices with zero potential in the EndOfPotentialLine instance. By construction, the next lemma follows:

▶ **Lemma 25.** *S', P', V' are well defined and polynomial in the sizes of S, P, V respectively.*

Our main theorem in this section is a consequence of the following three lemmas.

▶ **Lemma 26.** *For any $\boldsymbol{x} = (b, \boldsymbol{u}) \in \{0,1\}^k$, $P'(\boldsymbol{x}) = S'(\boldsymbol{x}) = \boldsymbol{x}$ (self loop) iff $\boldsymbol{x} \neq 0^k$, and $b = 0$ or $V(\boldsymbol{u}) = 0$.*

Proof. This follows by the construction of V', the second condition in S' and P', and third and fourth conditions in S' and P' respectively. ◀

▶ **Lemma 27.** *Let $\boldsymbol{x} = (b, \boldsymbol{u}) \in \{0,1\}^k$ be such that $S'(P'(\boldsymbol{x})) \neq \boldsymbol{x} \neq 0^k$ or $P'(S'(\boldsymbol{x})) \neq \boldsymbol{x}$ (an R1 type solution of* EndOfPotentialLine *instance $\mathcal{I}'$), then $\boldsymbol{u}$ is a solution of* EndOfMeteredLine *instance $\mathcal{I}$.*

Proof. The proof requires a careful case analysis. By the first conditions in the descriptions of S', P' and V', we have $\boldsymbol{x} \neq 0^k$. Further, since $\boldsymbol{x}$ is not a self loop, Lemma 26 implies $b = 1$ and $V'(1, \boldsymbol{u}) = V(\boldsymbol{u}) > 0$.

Case I. If $S'(P'(\boldsymbol{x})) \neq \boldsymbol{x} \neq 0^k$ then we will show that either $\boldsymbol{u}$ is a genuine start of a line other than 0^n giving a T1 type solution of EndOfMeteredLine instance $\mathcal{I}$, or there is some issue with the potential at $\boldsymbol{u}$ giving either a T2 or T3 type solution of $\mathcal{I}$. Since $S'(P'(1, 0^n)) = (1, 0^n)$, $\boldsymbol{u} \neq 0^n$. Thus if $S(P(\boldsymbol{u})) \neq \boldsymbol{u}$ then we get a T1 type solution of $\mathcal{I}$ and proof follows. If $V(\boldsymbol{u}) = 1$ then we get a T2 solution of $\mathcal{I}$ and proof follows.

Otherwise, we have $S(P(\boldsymbol{u})) = \boldsymbol{u}$ and $V(\boldsymbol{u}) > 1$. Now since also $b = 1$ $(1, \boldsymbol{u})$ is not a self loop (Lemma 26). Then it must be the case that $P'(1, \boldsymbol{u}) = (1, P(\boldsymbol{u}))$. However, $S'(1, P(\boldsymbol{u})) \neq (1, \boldsymbol{u})$ even though $S(P(\boldsymbol{u})) = \boldsymbol{u}$. This happens only when $P(\boldsymbol{u})$ is a self loop because of $V(P(\boldsymbol{u})) = 0$ (third condition of P'). Therefore, we have $V(\boldsymbol{u}) - V(P(\boldsymbol{u})) > 1$ implying that $\boldsymbol{u}$ is a T3 type solution of $\mathcal{I}$.

Case II. Similarly, if $P'(S'(\boldsymbol{x})) \neq \boldsymbol{x}$, then either $\boldsymbol{u}$ is a genuine end of a line of $\mathcal{I}$, or there is some issue with the potential at $\boldsymbol{u}$. If $P(S(\boldsymbol{u})) \neq \boldsymbol{u}$ then we get T1 solution of $\mathcal{I}$. Otherwise, $P(S(\boldsymbol{u})) = \boldsymbol{u}$ and $V(\boldsymbol{u}) > 0$. Now as $(b, \boldsymbol{u})$ is not a self loop and $V(\boldsymbol{u}) > 0$, it must be the case that $S'(b, \boldsymbol{u}) = (1, S(\boldsymbol{u}))$. However, $P'(1, S(\boldsymbol{u})) \neq (b, \boldsymbol{u})$ even though $P(S(\boldsymbol{u})) = \boldsymbol{u}$. This happens only when $S(\boldsymbol{u})$ is a self loop because of $V(S(\boldsymbol{u})) = 0$. Therefore, we get $V(S(\boldsymbol{u})) - V(\boldsymbol{u}) < 0$, i.e., $\boldsymbol{u}$ is a type T3 solution of $\mathcal{I}$. ◀

▶ **Lemma 28.** *Let $\boldsymbol{x} = (b, \boldsymbol{u}) \in \{0,1\}^k$ be an R2 type solution of the constructed* EndOfPotentialLine *instance $\mathcal{I}'$, then $\boldsymbol{u}$ is a type T3 solution of* EndOfMeteredLine *instance $\mathcal{I}$.*

Proof. Clearly, $\boldsymbol{x} \neq 0^k$. Let $\boldsymbol{y} = (b', \boldsymbol{u}') = S'(\boldsymbol{x}) \neq \boldsymbol{x}$, and observe that $P(\boldsymbol{y}) = \boldsymbol{x}$. This also implies that $\boldsymbol{y}$ is not a self loop, and hence $b = b' = 1$ and $V(\boldsymbol{u}) > 0$ (Lemma 26). Further, $\boldsymbol{y} = S'(1, \boldsymbol{u}) = (1, S(\boldsymbol{u}))$, hence $\boldsymbol{u}' = S(\boldsymbol{u})$. Also, $V'(\boldsymbol{x}) = V'(1, \boldsymbol{u}) = V(\boldsymbol{u})$ and $V'(\boldsymbol{y}) = V'(1, \boldsymbol{u}') = V(\boldsymbol{u}')$.

Since $V'(\boldsymbol{y}) - V'(\boldsymbol{x}) \leq 0$ we get $V(\boldsymbol{u}') - V(\boldsymbol{u}) \leq 0 \Rightarrow V(S(\boldsymbol{u})) - V(\boldsymbol{u}) \leq 0 \Rightarrow V(S(\boldsymbol{u})) - V(\boldsymbol{u}) \neq 1$. Given that $V(\boldsymbol{u}) > 0$, $\boldsymbol{u}$ gives a type T3 solution of EndOfMeteredLine. ◀

▶ **Theorem 29.** *An instance of* EndOfMeteredLine *can be reduced to an instance of* EndOfPotentialLine *in linear time such that a solution of the former can be constructed in a linear time from the solution of the latter.*

B EndOfPotentialLine to EndOfMeteredLine

In this section we give a linear time reduction from an instance $\mathcal{I}$ of EndOfPotentialLine to an instance $\mathcal{I}'$ of EndOfMeteredLine. Let the given EndOfPotentialLine instance $\mathcal{I}$ be defined on vertex set $\{0,1\}^n$ and with procedures S, P and V, where $V : \{0,1\}^n \to \{0, \ldots, 2^m - 1\}$.

Valid Edge. We call an edge $\boldsymbol{u} \to \boldsymbol{v}$ valid if $\boldsymbol{v} = S(\boldsymbol{u})$ and $\boldsymbol{u} = P(\boldsymbol{v})$.

We construct an EndOfMeteredLine instance $\mathcal{I}'$ on $\{0,1\}^k$ vertices where $k = n + m$. Let S', P' and V' denotes the procedures for $\mathcal{I}'$ instance. The idea is to capture value $V(\boldsymbol{x})$ of the potential in the m least significant bits of vertex description itself, so that it can be gradually increased or decreased on valid edges. For vertices with irrelevant values of these least m significant bits we will create self loops. Invalid edges will also become self loops, e.g.,

if $\boldsymbol{y} = S(\boldsymbol{x})$ but $P(\boldsymbol{y}) \neq \boldsymbol{x}$ then set $S'(\boldsymbol{x}, .) = (\boldsymbol{x}, .)$. We will see how these can not introduce new solutions.

In order to ensure $V'(0^k) = 1$, the $V(S(0^n)) = 1$ case needs to be discarded. For this, we first do some initial checks to see if the given instance $\mathcal{I}$ is not trivial. If the input EndOfPotentialLine instance is trivial, in the sense that either 0^n or $S(0^n)$ is a solution, then we can just return it.

▶ **Lemma 30.** *If 0^n or $S(0^n)$ are not solutions of* EndOfPotentialLine *instance $\mathcal{I}$ then $0^n \to S(0^n) \to S(S(0^n))$ are valid edges, and $V(S(S(0^n)) \geq 2$.*

Proof. Since both 0^n and $S(0^n)$ are not solutions, we have $V(0^n) < V(S(0^n)) < V(S(S(0^n)))$, $P(S(0^n)) = 0^n$, and for $\boldsymbol{u} = S(0^n)$, $S(P(\boldsymbol{u})) = \boldsymbol{u}$ and $P(S(\boldsymbol{u})) = \boldsymbol{u}$. In other words, $0^n \to S(0^n) \to S(S(0^n))$ are valid edges, and since $V(0^n) = 0$, we have $V(S(S(0^n)) \geq 2$. ◀

Let us assume now on that 0^n and $S(0^n)$ are not solutions of $\mathcal{I}$, and then by Lemma 30, we have $0^n \to S(0^n) \to S(S(0^n))$ are valid edges, and $V(S(S(0^n)) \geq 2$. We can avoid the need to check whether $V(S(0))$ is one all together, by making 0^n point directly to $S(S(0^n))$ and make $S(0^n)$ a dummy vertex.

We first construct S' and P', and then construct V' which will give value zero to all self loops, and use the least significant m bits to give a value to all other vertices. Before describing S' and P' formally, we first describe the underlying principles. Recall that in $\mathcal{I}$ vertex set is $\{0,1\}^n$ and possible potential values are $\{0, \ldots, 2^m - 1\}$, while in $\mathcal{I}'$ vertex set is $\{0,1\}^k$ where $k = m + n$. We will denote a vertex of $\mathcal{I}'$ by a tuple $(\boldsymbol{u}, \pi)$, where $\boldsymbol{u} \in \{0,1\}^n$ and $\pi \in \{0, \ldots, 2^m - 1\}$. Here when we say that we introduce an *edge* $\boldsymbol{x} \to \boldsymbol{y}$ we mean that we introduce a valid edge from $\boldsymbol{x}$ to $\boldsymbol{y}$, i.e., $\boldsymbol{y} = S'(\boldsymbol{x})$ and $\boldsymbol{x} = P(\boldsymbol{y})$.

- Vertices of the form $(S(0^n), \pi)$ for any $\pi \in \{0,1\}^m$ and the vertex $(0^n, 1)$ are dummies and hence have self loops.
- If $V(S(S(0^n)) = 2$ then we introduce an edge $(0^n, 0) \to (S(S(0^n)), 2)$, otherwise
  - for $p = V(S(S(0^n))$, we introduce the edges $(0^n, 0) \to (0^n, 2) \to (0^n, 3) \ldots (0^n, p-1) \to (S(S(0^n)), p)$.
- If $\boldsymbol{u} \to \boldsymbol{u}'$ valid edge in $\mathcal{I}$ then let $p = V(\boldsymbol{u})$ and $p' = V(\boldsymbol{u}')$
  - If $p = p'$ then we introduce the edge $(\boldsymbol{u}, p) \to (\boldsymbol{u}', p')$.
  - If $p < p'$ then we introduce the edges $(\boldsymbol{u}, p) \to (\boldsymbol{u}, p+1) \to \ldots \to (\boldsymbol{u}, p'-1) \to (\boldsymbol{u}', p')$.
  - If $p > p'$ then we introduce the edges $(\boldsymbol{u}, p) \to (\boldsymbol{u}, p-1) \to \ldots \to (\boldsymbol{u}, p'+1) \to (\boldsymbol{u}', p')$.
- If $\boldsymbol{u} \neq 0^n$ is the start of a path, i.e., $S(P(\boldsymbol{u})) \neq \boldsymbol{u}$, then make $(\boldsymbol{u}, V(\boldsymbol{u}))$ start of a path by ensuring $P'(\boldsymbol{u}, V(\boldsymbol{u})) = (\boldsymbol{u}, V(\boldsymbol{u}))$.
- If $\boldsymbol{u}$ is the end of a path, i.e., $P(S(\boldsymbol{u})) \neq \boldsymbol{u}$, then make $(\boldsymbol{u}, V(\boldsymbol{u}))$ end of a path by ensuring $S'(\boldsymbol{u}, V(\boldsymbol{u})) = (\boldsymbol{u}, V(\boldsymbol{u}))$.

Last two bullets above remove singleton solutions from the system by making them self loops. However, this can not kill all the solutions since there is a path starting at 0^n, which has to end somewhere. Further, note that this entire process ensures that no new start or end of a paths are introduced.

Procedure $S'(\boldsymbol{u}, \pi)$.

1. If ($\boldsymbol{u} = 0^n$ and $\pi = 1$) or $\boldsymbol{u} = S(0^n)$ then Return $(\boldsymbol{u}, \pi)$.
2. If $(\boldsymbol{u}, \pi) = 0^k$, then let $\boldsymbol{u}' = S(S(0^n))$ and $p' = V(\boldsymbol{u}')$.
   a. If $p' = 2$ then Return $(\boldsymbol{u}', 2)$ else Return $(0^n, 2)$.

3. If $\boldsymbol{u} = 0^n$ then
   a. If $2 \leq \pi < p' - 1$ then Return $(0^n, \pi + 1)$.
   b. If $\pi = p' - 1$ then Return $(S(S(0^n)), p')$.
   c. If $\pi \geq p'$ then Return $(\boldsymbol{u}, \pi)$.
4. Let $\boldsymbol{u}' = S(\boldsymbol{u})$, $p' = V(\boldsymbol{u}')$, and $p = V(\boldsymbol{u})$.
5. If $P(\boldsymbol{u}') \neq \boldsymbol{u}$ or $\boldsymbol{u}' = \boldsymbol{u}$ then Return $(\boldsymbol{u}, \pi)$
6. If $\pi = p = p'$ or ($\pi = p$ and $p' = p + 1$) or ($\pi = p$ and $p' = p - 1$) then Return $(\boldsymbol{u}', p')$.
7. If $\pi < p \leq p'$ or $p \leq p' \leq \pi$ or $\pi > p \geq p'$ or $p \geq p' \geq \pi$ then Return $(\boldsymbol{u}, \pi)$
8. If $p < p'$, then If $p \leq \pi < p' - 1$ then Return $(\boldsymbol{u}, \pi + 1)$. If $\pi = p' - 1$ then Return $(\boldsymbol{u}', p')$.
9. If $p > p'$, then if $p \geq \pi > p' + 1$ then Return $(\boldsymbol{u}, \pi - 1)$. If $\pi = p' + 1$ then Return $(\boldsymbol{u}', p')$.

Procedure $P'(\boldsymbol{u}, \pi)$.

1. If ($\boldsymbol{u} = 0^n$ and $\pi = 1$) or $\boldsymbol{u} = S(0^n)$ then Return $(\boldsymbol{u}, \pi)$.
2. If $\boldsymbol{u} = 0^n$, then
   a. If $\pi = 0$ then Return 0^k.
   b. If $\pi < V(S(S(0^n)))$ and $\pi \notin \{1, 2\}$ then Return $(0^n, \pi - 1)$.
   c. If $\pi < V(S(S(0^n)))$ and $\pi = 2$ then Return 0^k.
3. If $\boldsymbol{u} = S(S(0^n))$ and $\pi = V(S(S(0^n))$ then
   a. If $\pi = 2$ then Return $(0^n, 0)$, else Return $(0^n, \pi - 1)$.
4. If $\pi = V(\boldsymbol{u})$ then
   a. Let $\boldsymbol{u}' = P(\boldsymbol{u})$, $p' = V(\boldsymbol{u}')$, and $p = V(\boldsymbol{u})$.
   b. If $S(\boldsymbol{u}') \neq \boldsymbol{u}$ or $\boldsymbol{u}' = \boldsymbol{u}$ then Return $(\boldsymbol{u}, \pi)$
   c. If $p = p'$ then Return $(\boldsymbol{u}', p')$
   d. If $p' < p$ then Return $(\boldsymbol{u}', p - 1)$ else Return $(\boldsymbol{u}', p + 1)$
5. Else % when $\pi \neq V(\boldsymbol{u})$
   a. Let $\boldsymbol{u}' = S(\boldsymbol{u})$, $p' = V(\boldsymbol{u}')$, and $p = V(\boldsymbol{u})$
   b. If $P(\boldsymbol{u}') \neq \boldsymbol{u}$ or $\boldsymbol{u}' = \boldsymbol{u}$ then Return $(\boldsymbol{u}, \pi)$
   c. If $p' = p$ or $\pi < p < p'$ or $p < p' \leq \pi$ or $\pi > p > p'$ or $p > p' \geq \pi$ then Return $(\boldsymbol{u}, \pi)$
   d. If $p < p'$, then If $p < \pi \leq p' - 1$ then Return $(\boldsymbol{u}, \pi - 1)$.
   e. If $p > p'$, then if $p > \pi \geq p' + 1$ then Return $(\boldsymbol{u}, \pi + 1)$.

As mentioned before, the intuition for the potential function procedure V' is to return zero for self loops, return 1 for 0^k, and return the number specified by the lowest m bits for the rest.

Procedure $V'(\boldsymbol{u}, \pi)$. Let $\boldsymbol{x} = (\boldsymbol{u}, \pi)$ for notational convenience.

1. If $\boldsymbol{x} = 0^k$, then Return 1.
2. If $S'(\boldsymbol{x}) = \boldsymbol{x}$ and $P'(\boldsymbol{x}) = \boldsymbol{x}$ then Return 0.
3. If $S'(\boldsymbol{x}) \neq \boldsymbol{x}$ or $P'(\boldsymbol{x}) \neq \boldsymbol{x}$ then Return π.

The fact that procedures S', P' and V' give a valid EndOfMeteredLine instance follows from construction.

▶ **Lemma 31.** *Procedures S', P' and V' gives a valid* EndOfMeteredLine *instance on vertex set $\{0,1\}^k$, where $k = m + n$ and $V' : \{0,1\}^k \to \{0, \ldots, 2^k - 1\}$.*

The next three lemmas shows how to construct a solution of EndOfPotentialLine instance $\mathcal{I}$ from a type T1, T2, or T3 solution of constructed EndOfMeteredLine instance $\mathcal{I}'$. The basic idea for next lemma, which handles type T1 solutions, is that we never create spurious end or start of a path.

▶ **Lemma 32.** *Let $\boldsymbol{x} = (\boldsymbol{u}, \pi)$ be a type T1 solution of constructed* EndOfMeteredLine *instance $\mathcal{I}'$. Then $\boldsymbol{u}$ is a type R1 solution of the given* EndOfPotentialLine *instance $\mathcal{I}$.*

Proof. Let $\Delta = 2^m - 1$. In $\mathcal{I}'$, clearly $(0^n, \pi)$ for any $\pi \in 1, \ldots, \Delta$ is not a start or end of a path, and $(0^n, 0)$ is not an end of a path. Therefore, $\boldsymbol{u} \neq 0^n$. Since $(S(0^n), \pi), \forall \pi \in \{0, \ldots, \Delta\}$ are self loops, $\boldsymbol{u} \neq S(0^n)$.

If to the contrary, $S(P(\boldsymbol{u})) = \boldsymbol{u}$ and $P(S(\boldsymbol{u})) = \boldsymbol{u}$. If $S(\boldsymbol{u}) = \boldsymbol{u} = P(\boldsymbol{u})$ then $(\boldsymbol{u}, \pi)$, $\forall \pi \in \{0, \ldots, \Delta\}$ are self loops, a contradiction.

For the remaining cases, let $P'(S'(\boldsymbol{x})) \neq \boldsymbol{x}$, and let $\boldsymbol{u}' = S(\boldsymbol{u})$. . There is a valid edge from $\boldsymbol{u}$ to $\boldsymbol{u}'$ in $\mathcal{I}$. Then we will create valid edges from $(\boldsymbol{u}, V(\boldsymbol{u}))$ to $(S(\boldsymbol{u}), V(S(\boldsymbol{u}))$ with appropriately changing second coordinates. The rest of $(\boldsymbol{u}, .)$ are self loops, a contradiction.

Similar argument follows for the case when $S'(P'(\boldsymbol{x})) \neq \boldsymbol{x}$. ◀

The basic idea behind the next lemma is that a T2 type solution in $\mathcal{I}'$ has potential 1. Therefore, it is surely not a self loop. Then it is either an end of a path or near an end of a path, or else near a potential violation.

▶ **Lemma 33.** *Let $\boldsymbol{x} = (\boldsymbol{u}, \pi)$ be a type T2 solution of $\mathcal{I}'$. Either $\boldsymbol{u} \neq 0^n$ is start of a path in $\mathcal{I}$ (type R1 solution), or $P(\boldsymbol{u})$ is an R1 or R2 type solution in $\mathcal{I}$, or $P(P(\boldsymbol{u}))$ is an R2 type solution in $\mathcal{I}$.*

Proof. Clearly $\boldsymbol{u} \neq 0^n$, and $\boldsymbol{x}$ is not a self loop, i.e., it is not a dummy vertex with irrelevant value of π. Further, $\pi = 1$. If $\boldsymbol{u}$ is a start or end of a path in $\mathcal{I}$ then done.

Otherwise, if $V(P(\boldsymbol{u})) > \pi$ then we have $V(\boldsymbol{u}) \leq \pi$ and hence $V(\boldsymbol{u}) - V(P(\boldsymbol{u})) \leq 0$ giving $P(\boldsymbol{u})$ as an R2 type solution of $\mathcal{I}$. If $V(P(\boldsymbol{u})) < \pi = 1$ then $V(P(\boldsymbol{u})) = 0$. Since potential can not go below zero, either $P(\boldsymbol{u})$ is an end of a path, or for $\boldsymbol{u}'' = P(P(\boldsymbol{u}))$ and $\boldsymbol{u}' = P(\boldsymbol{u})$ we have $\boldsymbol{u}' = S(\boldsymbol{u}'')$ and $V(\boldsymbol{u}') - V(\boldsymbol{u}'') \leq 0$, giving $\boldsymbol{u}''$ as a type R2 solution of $\mathcal{I}$. ◀

At a type T3 solution of $\mathcal{I}'$ potential is strictly positive, hence these solutions are not self loops. If they correspond to potential violation in $\mathcal{I}$ then we get a type R2 solution. But this may not be the case, if we made S' or P' self pointing due to end or start of a path respectively. In that case, we get a type R1 solution. The next lemma formalizes this intuition.

▶ **Lemma 34.** *Let $\boldsymbol{x} = (\boldsymbol{u}, \pi)$ be a type T3 solution of $\mathcal{I}'$. If $\boldsymbol{x}$ is a start or end of a path in $\mathcal{I}'$ then $\boldsymbol{u}$ gives a type R1 solution in $\mathcal{I}$. Otherwise $\boldsymbol{u}$ gives a type R2 solution of $\mathcal{I}$.*

Proof. Since $V'(\boldsymbol{x}) > 0$, it is not a self loop and hence is not dummy, and $\boldsymbol{u} \neq 0^n$. If $\boldsymbol{u}$ is start or end of a path then $\boldsymbol{u}$ is a type R1 solution of $\mathcal{I}$. Otherwise, there are valid incoming and outgoing edges at $\boldsymbol{u}$, therefore so at $\boldsymbol{x}$.

If $V((S(\boldsymbol{x})) - V(\boldsymbol{x}) \neq 1$, then since potential either remains the same or increases or decreases exactly by one on edges of $\mathcal{I}'$, it must be the case that $V(S(\boldsymbol{x})) - V(\boldsymbol{x}) \leq 0$. This is possible only when $V(S(\boldsymbol{u})) \leq V(\boldsymbol{u})$. Since $\boldsymbol{u}$ is not an end of a path we do have $S(\boldsymbol{u}) \neq \boldsymbol{u}$ and $P(S(\boldsymbol{u})) = \boldsymbol{u}$. Thus, $\boldsymbol{u}$ is a type T2 solution of $\mathcal{I}$.

If $V((\boldsymbol{x}) - V(P(\boldsymbol{x})) \neq 1$, then by the same argument we get that for $(\boldsymbol{u}'', \pi'') = P(\boldsymbol{u})$, $\boldsymbol{u}''$ is a type R2 solution of $\mathcal{I}$. ◀

Our main theorem follows using Lemmas 31, 32, 33, and 34.

▶ **Theorem 35.** *An instance of* EndOfPotentialLine *can be reduced to an instance of* EndOfMeteredLine *in polynomial time such that a solution of the former can be constructed in a linear time from the solution of the latter.*

C Pseudo-code for Lemke's algorithm

If $\boldsymbol{q} \geq 0$ **then Return** $\boldsymbol{y} \leftarrow \boldsymbol{0}$
$\boldsymbol{y} \leftarrow 0, z \leftarrow |\min_{i\in[d]} q_i|, \boldsymbol{s} = \boldsymbol{q} + z\boldsymbol{1}$
$i \leftarrow$ duplicate label at vertex $(\boldsymbol{y}, \boldsymbol{s}, z)$ in $\mathcal{P}$. $flag \leftarrow 1$
While $z > 0$ **do**
  If $flag = 1$ **then** set $(\boldsymbol{y}', \boldsymbol{s}', z') \leftarrow$ vertex obtained by relaxing $y_i = 0$ at $(\boldsymbol{y}, \boldsymbol{s}, z)$ in $\mathcal{P}$
  Else set $(\boldsymbol{y}', \boldsymbol{s}', z') \leftarrow$ vertex obtained by relaxing $s_i = 0$ at $(\boldsymbol{y}, \boldsymbol{s}, z)$ in $\mathcal{P}$
  If $z > 0$ **then**
    $i \leftarrow$ duplicate label at $(\boldsymbol{y}', \boldsymbol{s}', z')$
    If $v_i > 0$ and $v_i' = 0$ **then** $flag \leftarrow 1$. **Else** $flag \leftarrow 0$
    $(\boldsymbol{y}, \boldsymbol{s}, z) \leftarrow (\boldsymbol{y}', \boldsymbol{s}', z')$
End **While**
Return $\boldsymbol{y}$

D Missing Procedures and Proofs from Section 5

D.1 Procedures IsValid, ItoE, and EtoI

Table 4 Procedure IsValid($\boldsymbol{u}$)

If $\boldsymbol{u} = 0^n$ **then Return** 1
Else let $\tau = (u_{(d+1)} + \cdots + u_{2d})$
  If $\tau > 1$ **then Return** 0
  Let $S \leftarrow \emptyset$. % set of tight inequalities.
  If $\tau = 0$ **then** $S = S \cup \{z = 0\}$.
  Else
    Set $l \leftarrow$ index of the non-zero coordinate in vector $(u_{(d+1)}, \ldots, u_{2d})$.
    Set $S = \{y_l = 0, s_l = 0\}$.
  For each i from 1 to d **do**
    If $u_i = 0$ **then** $S = S \cup \{y_i = 0\}$, **Else** $S = S \cup \{s_i = 0\}$
  Let A be a matrix formed by lhs of equalities $M\boldsymbol{y} + \boldsymbol{s} - \mathbf{1}z = \boldsymbol{q}$ and that of set S
  Let $\boldsymbol{b}$ be the corresponding rhs, namely $\boldsymbol{b} = [\boldsymbol{q}; \mathbf{0}_{d\times 1}]$.
  Let $(\boldsymbol{y}', \boldsymbol{s}', z') \leftarrow \boldsymbol{b} * A^{-1}$
  If $(\boldsymbol{y}', \boldsymbol{s}', z') \in \mathcal{P}$ **then Return** 1, **Else Return** 0

Table 5 Procedures ItoE($\boldsymbol{u}$) and EtoI($\boldsymbol{y}$,$\boldsymbol{s}$,z)

ItoE($\boldsymbol{y}, \boldsymbol{s}, z$)
  If $\exists i \in [d]$ s.t. $y_i * s_i \neq 0$ **then Return** $(\mathbf{0}_{(2d-2)\times 1}; 1; 1)$ % Invalid
  Set $\boldsymbol{u} \leftarrow \mathbf{0}_{2d\times 1}$. Let $DL = \{i \in [d] \mid y_i = 0 \text{ and } s_i = 0\}$.
  If $|DL| > 1$ **then Return** $(\mathbf{0}_{(2d-2)\times 1}; 1; 1)$ %In valid
  If $|DL| = 1$ **then** for $i \in DL$, set $u_i \leftarrow 1$
  For each $i \in [d]$ **If** $s_i = 0$ **then** set $u_{d+i} \leftarrow 1$
  Return $\boldsymbol{u}$

EtoI($\boldsymbol{u}$)
  If $\boldsymbol{u} = 0^n$ **then Return** $(\mathbf{0}_{d\times 1}, \boldsymbol{q} + z^0 + 1, z^0 + 1)$ % This case will never happen
  If IsValid($\boldsymbol{u}$)=0 **then Return** $\mathbf{0}_{(2d+1)\times 1}$
  Let $\tau = (u_{(d+1)} + \cdots + u_{2d})$
  Let $S \leftarrow \emptyset$. % set of tight inequalities.
  If $\tau = 0$ **then** $S = S \cup \{z = 0\}$.
  Else
    Set $l \leftarrow$ index of non-zero coordinate in vector $(u_{(d+1)}, \ldots, u_{2d})$.
    Set $S = \{y_l = 0, s_l = 0\}$.
  For each i from 1 to d **do**
    If $u_i = 0$ **then** $S = S \cup \{y_i = 0\}$, **Else** $S = S \cup \{s_i = 0\}$
  Let A be a matrix formed by lhs of equalities $M\boldsymbol{y} + \boldsymbol{s} - \mathbf{1}z = \boldsymbol{q}$ and that of set S
  Let $\boldsymbol{b}$ be the corresponding rhs, namely $\boldsymbol{b} = [\boldsymbol{q}; \mathbf{0}_{d\times 1}]$.
  Return $\boldsymbol{b} * A^{-1}$

D.2 Proof of Lemma 18

Lemma 18 (restated) : *If IsValid*$(\boldsymbol{u}) = 1$ *then* $\boldsymbol{u} = ItoE(EtoI(\boldsymbol{u}))$*, and the corresponding vertex* $(\boldsymbol{y}, \boldsymbol{s}, z) \in EtoI(\boldsymbol{u})$ *of* $\mathcal{P}$ *is feasible in (3). If* $(\boldsymbol{y}, \boldsymbol{s}, z)$ *is a feasible vertex of (3) then* $\boldsymbol{u} = ItoE(\boldsymbol{y}, \boldsymbol{s}, z)$ *is a valid configuration,* i.e., *IsValid*$(\boldsymbol{u}) = 1$.

Proof. The only thing that can go wrong is that the matrix A generated in IsValid and EtoI procedures are singular, or the set of double labels DL generated in ItoE has more than one elements. Each of these are possible only when more than $2d + 1$ equalities of $\mathcal{P}$ hold at the corresponding point $(\boldsymbol{y}, \boldsymbol{s}, z)$, violating non-degeneracy assumption. ◀

D.3 Proof of Lemma 19

Lemma 19 (restated) : *Functions* P*,* S *and* V *of instance* $\mathcal{E}$ *are well defined, making* $\mathcal{E}$ *a valid* EndOfPotentialLine *instance.*

Proof. Since all three procedures are polynomial-time in $\mathcal{L}$, they can be defined by $poly(\mathcal{L})$-sized Boolean circuits. Furthermore, for any $\boldsymbol{u} \in \mathsf{vert}$, we have that $S(\boldsymbol{u}), P(\boldsymbol{u}) \in \mathsf{vert}$. For V, since the value of $z \in [0,\ \Delta - 1]$, we have $0 \leq \Delta^2(\Delta - z) \leq \Delta^3$. Therefore, $V(\boldsymbol{u})$ is an integer that is at most $2 \cdot \Delta^3$ and hence is in set $\{0, \ldots, 2^m - 1\}$. ◀

D.4 Proof of Lemma 20

Lemma 20 (restated) : *Let* $\boldsymbol{u} \neq \boldsymbol{u}'$ *be two valid configurations, i.e.,* *IsValid*$(\boldsymbol{u}) =$ *IsValid*$(\boldsymbol{u}') = 1$*, and let* $(\boldsymbol{y}, \boldsymbol{s}, z)$ *and* $(\boldsymbol{y}', \boldsymbol{s}', z')$ *be the corresponding vertices in* $\mathcal{P}$*. Then the following holds:* (i) $V(\boldsymbol{u}) = V(\boldsymbol{u}')$ *iff* $z = z'$*.* (ii) $V(\boldsymbol{u}) > V(\boldsymbol{u}')$ *iff* $z < z'$*.*

Proof. Among the valid configurations all except $\mathbf{0}$ has positive V value. Therefore, wlog let $\boldsymbol{u}, \boldsymbol{u}' \neq \mathbf{0}$. For these we have $V(\boldsymbol{u}) = \lfloor \Delta^2 * (\Delta - z) \rfloor$, and $V(\boldsymbol{u}') = \lfloor \Delta^2 * (\Delta - z') \rfloor$.

Note that since both z and z' are coordinates of vertices of $\mathcal{P}$, whose description has highest coefficient of $\max\{\max_{i,j \in [d]} M(i,j), \max_{i \in [d]} |q_i|\}$, and therefore their numerator and denominator both are bounded above by Δ. Therefore, if $z < z'$ then we have

$$z' - z \geq \frac{1}{\Delta^2} \Rightarrow ((\Delta - z) - (\Delta - z')) * \Delta^2 \geq 1 \Rightarrow V(\boldsymbol{u}) - V(\boldsymbol{u}') \geq 1.$$

For (i), if $z = z'$ then clearly $V(\boldsymbol{u}) = V(\boldsymbol{u}')$, and from the above argument it also follows that if $V(\boldsymbol{u}) = V(\boldsymbol{u}')$ then it can not be the case that $z \neq z'$. Similarly for (ii), if $V(\boldsymbol{u}) > V(\boldsymbol{u}')$ then clearly, $z' > z$, and from the above argument it follows that if $z' > z$ then it can not be the case that $V(\boldsymbol{u}') \geq V(\boldsymbol{u})$. ◀