

Linear Complementarity Algorithms for Infinite Games^{*}

John Fearnley¹, Marcin Jurdziński¹, and Rahul Savani²

¹ Department of Computer Science, University of Warwick, UK

² Department of Computer Science, University of Liverpool, UK

Abstract. The performance of two pivoting algorithms, due to Lemke and Cottle and Dantzig, is studied on linear complementarity problems (LCPs) that arise from infinite games, such as parity, average-reward, and discounted games. The algorithms have not been previously studied in the context of infinite games, and they offer alternatives to the classical strategy-improvement algorithms. The two algorithms are described purely in terms of discounted games, thus bypassing the reduction from the games to LCPs, and hence facilitating a better understanding of the algorithms when applied to games. A family of parity games is given, on which both algorithms run in exponential time, indicating that in the worst case they perform no better for parity, average-reward, or discounted games than they do for general P-matrix LCPs.

1 Introduction

In this paper we consider infinite-duration zero-sum games played on finite graphs, such as parity, average-reward, and discounted games. Parity games are important in the theory of algorithmic formal verification because they provide a combinatorial characterization of the meaning of nested inductive and co-inductive definitions, as formalized in the modal μ -calculus and other fixpoint logics [12]. In particular, deciding the winner in parity games is polynomial-time equivalent to checking non-emptiness of non-deterministic parity tree automata, and to the modal μ -calculus model checking, two fundamental algorithmic problems in automata theory, logic, and verification [7,18,12]. Discounted and average-reward games have been introduced by Shapley [17] and Gillette [11] in the 1950s, and they have been extensively studied in the game theory, mathematical programming, algorithms, and AI communities [21,8]. Parity, average-reward, and discounted games have an intriguing complexity-theoretic status. The problems of deciding the winner in these games are some of the few known combinatorial problems in $\text{NP} \cap \text{co-NP}$ (and even $\text{UP} \cap \text{co-UP}$ [13]) that are not known to be solvable in polynomial time.

The linear complementarity problem (LCP) is a fundamental problem in mathematical programming. It naturally captures equilibrium problems, as well as the complementary slackness and Karush-Kuhn-Tucker conditions of linear

* An extended version of this paper with full proofs is available as [arXiv:0909.5653](https://arxiv.org/abs/0909.5653).

and quadratic programming, respectively. The monograph of Cottle *et. al.* [6] is the authoritative source on the LCP. In general, deciding if an LCP has a solution is NP-complete [3]. If, however, the matrix (which is a part of the LCP input) is a P-matrix (i.e., if all its principal minors are positive) then the problem is arguably easier computationally. Every P-matrix LCP (P-LCP) has a unique solution and computing it is in $PLS \cap PPAD$. A significant amount of effort has been invested by the mathematical programming community towards finding an efficient algorithm for the P-LCP, which has led to a wide body of literature in this area. Polynomial-time reductions from simple stochastic games [10,19] and discounted games [14] to the P-LCP have been recently proposed, however the techniques commonly used to solve P-LCPs remain largely unknown in the infinite games community. It is possible that these techniques could shed new light on the computational complexity of solving infinite games.

In this paper we consider two classical pivoting algorithms for the P-LCP, Lemke's algorithm and the Cottle-Dantzig principal pivoting algorithm, and we study their performance on P-LCPs obtained from discounted games by the reduction of Jurdziński and Savani [14]. Our first main contribution is to describe both algorithms purely as a process that works on the original discounted game, bypassing the reduction from games to the P-LCP, and hence we facilitate their analysis without the need to consider or understand concepts of the LCP theory. We present the algorithms for discounted games because they have technical advantages that make the exposition particularly transparent [14]. We argue, however, that this is done without loss of generality: the algorithms can be readily applied to parity games and average-reward games because there are transparent polynomial-time reductions from parity games to average-reward games [16,13], and from average-reward games to discounted games [21].

It has long been known that the two algorithms can take exponential time when applied to P-LCPs. However, it is not known whether these lower bounds hold for the LCPs that arise from infinite games. Our second main contribution is to prove that there is a family of discounted games on which the algorithms of Lemke, and Cottle and Dantzig run in exponential time, and hence we indicate limitations of the classical LCP theory in the context of infinite games. Our family of examples are derived from those given by Björklund and Vorobyov [2] for their strategy improvement algorithm for average-reward games. For technical convenience and without loss of generality, we present these families of hard examples as discounted games; it is easy to construct parity and average-reward games from which those discounted games are obtained via the standard reductions [16,13,21].

We stress that these lower bounds are not fatal. The lower bound for Lemke's algorithm requires a specific covering vector and the lower bound for the Cottle-Dantzig algorithm relies on a specific choice of ordering on the vertices. The covering vector and the ordering are free choices left up to the user of the algorithm. This situation can be compared to the classical strategy improvement algorithms for infinite games [5,20,2]. It has long been known that these algorithms can be made to run in exponential time by choosing sufficiently bad

vertices to switch [15]. However, it has only recently been shown that reasonable switching policies can be made to run in exponential time [9]. The complexity of our algorithms when equipped with reasonable covering vectors and reasonable orderings remains open. The literature on LCPs contains exciting complexity results for special cases. For example Adler and Megiddo [1] studied the performance of Lemke’s algorithm for the LCPs arising from linear programming problems. They showed that for randomly chosen linear programs and a carefully selected covering vector, the expected number of pivots performed by the algorithm is quadratic. Our results open the door to extending such analyses to infinite games.

2 Preliminaries

A binary discounted game is given by a tuple $G = (V, V_{\text{Max}}, V_{\text{Min}}, \lambda, \rho, r^\lambda, r^\rho, \beta)$, where V is a set of vertices and V_{Max} and V_{Min} partition V into the set of vertices of player Max and the set of vertices of player Min, respectively. Each vertex has exactly two outgoing edges which are given by the left and right successor functions $\lambda, \rho : V \rightarrow V$. Each edge has a reward associated with it given by the functions $r^\lambda, r^\rho : V \rightarrow \mathbb{R}$. Finally, the discount factor β is such that $0 \leq \beta < 1$.

The game begins with a token on a starting vertex v_0 . In each round, the player who owns the vertex on which the token is placed chooses one of the two successors of that vertex and moves the token to that successor. In this fashion the two players form an infinite path $\pi = \langle v_0, v_1, v_2, \dots \rangle$ where v_{i+1} is equal to either $\lambda(v_i)$ or $\rho(v_i)$. The path yields the infinite sequence of rewards $\langle r_0, r_1, r_2, \dots \rangle$, where $r_i = r^\lambda(v_i)$ if $\lambda(v_i) = v_{i+1}$, and $r_i = r^\rho(v_i)$ otherwise. The payoff of an infinite path is denoted by $\mathcal{D}(\pi) = \sum_{i=0}^{\infty} \beta^i r^i$. Since the game is zero-sum, player Max wins $\mathcal{D}(\pi)$ and player Min loses an equal amount.

A positional strategy for player Max is a function that, for each vertex belonging to player Max, chooses one of the two successors of the vertex. The strategy is denoted by $\chi : V_{\text{Max}} \rightarrow V$ with the condition that, for every vertex v in V_{Max} , the function $\chi(v)$ is equal to either $\lambda(v)$ or $\rho(v)$. Positional strategies for player Min are defined analogously. The sets of pure positional strategies for Max and Min are denoted by Π_{Max} and Π_{Min} , respectively. Given a pair of positional strategies, χ and μ for Max and Min respectively, and an initial vertex v_0 , there is a unique infinite path $\langle v_0, v_1, v_2, \dots \rangle$, where $\chi(v_i) = v_{i+1}$ if v_i is in V_{Max} and $\mu(v_i) = v_{i+1}$ if v_i is in V_{Min} . This path, referred to as the play induced by the two strategies, will be denoted by $\text{Play}(\chi, \mu, v_0)$.

For all v in V , we define $\text{Val}^*(v) = \min_{\mu \in \Pi_{\text{Min}}} \max_{\chi \in \Pi_{\text{Max}}} \mathcal{D}(\text{Play}(\chi, \mu, v))$, and $\text{Val}_*(v) = \max_{\chi \in \Pi_{\text{Max}}} \min_{\mu \in \Pi_{\text{Min}}} \mathcal{D}(\text{Play}(\chi, \mu, v))$. These will be known as the lower and upper values of v , respectively. It is always true that $\text{Val}_*(v) \leq \text{Val}^*(v)$. It is well known that for discounted games the two values are equal, a property known as determinacy.

Theorem 1 ([17]). *For every discounted game G and every vertex $v \in V$, we have $\text{Val}_*(v) = \text{Val}^*(v)$.*

The value of the game starting at a vertex v , equal to both $\text{Val}_*(v)$ and $\text{Val}^*(v)$, is denoted by $\text{Val}(v)$. The computational task associated with discounted games is to compute $\text{Val}(v)$. Moreover, we want to find optimal strategies, i.e., a strategy χ that achieves the upper value and a strategy μ that achieves the lower value.

For convenience, we introduce the concept of a joint strategy $\sigma : V \rightarrow V$ that specifies moves for both players. The notation $\sigma \upharpoonright \text{Max}$ and $\sigma \upharpoonright \text{Min}$ will be used to refer to the individual strategies of Max and Min that constitute the joint strategy. For a vertex v , the function $\bar{\sigma}(v)$ gives the successor of v not chosen by σ . The functions r^σ and $r^{\bar{\sigma}}$ give the reward on the edge chosen by σ and the reward on the edge not chosen by σ , respectively. The path denoted by $\text{Play}(\sigma, v)$ is equal to the path $\text{Play}(\sigma \upharpoonright \text{Max}, \sigma \upharpoonright \text{Min}, v)$. The joint strategy is optimal if both $\sigma \upharpoonright \text{Max}$ and $\sigma \upharpoonright \text{Min}$ are optimal. For a given joint strategy σ , the value of a vertex v when σ is played will be denoted by $\text{Val}^\sigma(v) = \mathcal{D}(\text{Play}(\sigma, v))$.

Given a joint strategy σ and a vertex v , the *balance* of v is the difference between the value of v and the value of the play that starts at v , moves to $\bar{\sigma}(v)$ in the first step, and then follows σ ,

$$\text{Bal}^\sigma(v) = \begin{cases} \text{Val}^\sigma(v) - (r^{\bar{\sigma}}(v) + \beta \cdot \text{Val}^\sigma(\bar{\sigma}(v))) & \text{if } v \in V_{\text{Max}}, \\ (r^{\bar{\sigma}}(v) + \beta \cdot \text{Val}^\sigma(\bar{\sigma}(v))) - \text{Val}^\sigma(v) & \text{if } v \in V_{\text{Min}}. \end{cases} \tag{1}$$

A vertex v is said to be switchable under σ if $\text{Bal}^\sigma(v) < 0$. If $\text{Bal}^\sigma(v) = 0$ for some vertex then that vertex is said to be indifferent. There is a simple characterisation of optimality in terms of switchable vertices.

Theorem 2 ([17]). *If no vertex is switchable in a joint strategy σ then it is an optimal strategy for every choice of starting vertex.*

The two algorithms that we will present use only positional joint strategies. From now on, all joint strategies that we refer to can be assumed to be positional joint strategies. If a play begins at a vertex v and follows a positional joint strategy σ then the resulting infinite path can be represented by a simple path followed by an infinitely repeated cycle. Let $\text{Play}(\sigma, v) = \langle v_0, v_1, \dots, v_{k-1}, \langle c_0, c_1, \dots, c_{l-1} \rangle^\omega \rangle$. It is then easy to see that

$$\text{Val}^\sigma(v) = \sum_{i=0}^{k-1} \beta^i \cdot r^\sigma(v_i) + \sum_{i=0}^{l-1} \frac{\beta^{k+i}}{1 - \beta^l} \cdot r^\sigma(c_i).$$

Therefore, the amount that the reward on the outgoing edge of a vertex u contributes towards the value of v can be defined as follows.

Definition 1 (Contribution Coefficient). *For vertices v and u , and for a positional joint strategy σ , we define:*

$$D_\sigma^v(u) = \begin{cases} \beta^i & \text{if } u = v_i \text{ for some } 0 \leq i < k, \\ \frac{\beta^{k+i}}{1 - \beta^l} & \text{if } u = c_i \text{ for some } 0 \leq i < l, \\ 0 & \text{otherwise.} \end{cases}$$

3 Lemke’s Algorithm for Discounted Games

Lemke’s algorithm is a classical algorithm for solving the linear complementarity problem [6]. We can apply Lemke’s algorithm to a discounted game by utilising the reduction of Jurdziński and Savani [14], however this yields little insight into how the algorithm works on a discounted game. In this section we bypass the reduction, and give a description of Lemke’s algorithm entirely in terms of discounted games.

Lemke’s algorithm begins with the joint strategy $\sigma_0 = \rho$ that selects the right successor for every vertex in the game. This is actually a free choice since the left and right successors can be swapped to obtain an arbitrary starting strategy. The algorithm will then move through a sequence of strategies until it arrives at the optimal strategy. The algorithm will also construct a modified game for each strategy that it considers. The modified games will take the following form.

Definition 2 (Modified Game for Lemke’s Algorithm). *For a real number z , we define the game G_z to be the same as G but with a modified left-edge reward function, denoted by r_z^λ , and defined, for every vertex v , by:*

$$r_z^\lambda(v) = \begin{cases} r^\lambda(v) - z & v \in V_{Max}, \\ r^\lambda(v) + z & v \in V_{Min}. \end{cases} \tag{2}$$

For a modified game G_z , the function r_z^σ will give the rewards on the edges chosen by σ . The notations Val_z^σ and Bal_z^σ will give the values the balances of the vertices in the game G_z , respectively. For every strategy σ_i that is considered, the algorithm must choose an appropriate value z_i so that σ_i is optimal in G_{z_i} . Moreover, we want to choose the minimum value z_i for which this property holds. The next proposition shows how to compute this for the initial strategy σ_0 .

Proposition 1. *Let $z_0 = \max\{-\text{Bal}^{\sigma_0}(v) : v \in V\}$. The strategy σ_0 is optimal in G_{z_0} and the vertex v in V that maximizes $-\text{Bal}^{\sigma_0}(v)$ is indifferent. Moreover, there is no value $y < z_0$ for which σ_0 is optimal in G_y .*

Proposition 1 gives an initial value for the parameter z . The principal idea behind the algorithm is to drive z down from its initial value to 0, while maintaining optimality of the current strategy in G_z . Unfortunately, Proposition 1 implies that we cannot drive z down further without losing the optimality of σ_0 in G_z . We do however know that there is some vertex v that is indifferent under σ_0 in G_{z_0} . We define $\sigma_1 = \sigma_0[\bar{\sigma}_0(v)/v]$, i.e., $\sigma_1(u) = \bar{\sigma}_0(u)$ if $u = v$, and $\sigma_1(u) = \sigma_0(u)$ otherwise. The operation of modifying a strategy by changing the successor of a vertex v will be referred to as switching v .

The value of no vertex changes when switching an indifferent vertex in a strategy. Since σ_0 was optimal in G_{z_0} and v was indifferent we therefore have that σ_1 is optimal in G_{z_0} . There is one important difference however, whereas z could not be decreased without σ_0 losing its optimality, the parameter z can be decreased further whilst maintaining the optimality of σ_1 . The task now is to find z_1 , the minimum value of z for which σ_1 is still optimal.

At a high level, when the algorithm arrives at a strategy σ_i its task is to find z_i , the minimum value of z for which σ_i is optimal in G_z . As we shall show, for this minimum value of z there will always be at least one vertex that is indifferent under σ_i played in G_z . The algorithm then switches this indifferent vertex to create σ_{i+1} and the process is repeated. The remainder of this section is dedicated to showing how z_i can be computed.

Each step begins with a strategy σ_i and the value z_{i-1} , which was the minimum value of z for which σ_{i-1} was optimal in G_z . We now wish to know how much further z can be decreased before σ_i ceases to be optimal. From Theorem 2 we know that a strategy is optimal as long as no vertex is switchable and that a vertex is switchable only when it has a negative balance. It is for this reason that we want to know how the balance of each vertex changes as z is decreased. In order to understand this, we must first know how the value of each vertex changes as z is decreased. We will use the notation $\partial_{-z} \text{Val}_z^\sigma(v)$ to denote the rate of change of the value of v as z decreases, i.e., $-\partial_z \text{Val}_z^\sigma(v)$. This notation will be used frequently throughout the rest of the paper to denote the rate of change of various expressions. For a proposition p , we define $[p]$ to be equal to 1 if p is true, and 0 otherwise. We can now give an explicit formula for $\partial_{-z} \text{Val}_z^\sigma(v)$, which is based on the left edges that are passed through after visiting the vertex v while playing the strategy σ , and the contribution coefficient of those edges to the value of v .

Proposition 2. *For a vertex v and a joint strategy σ , let L be the set of vertices for which σ picks the left successor, $L = \{v \in V : \sigma(v) = \lambda(v)\}$. The rate of change of the value of v is*

$$\partial_{-z} \text{Val}_z^\sigma(v) = \sum_{u \in L} ([u \in V_{Max}] - [u \in V_{Min}]) \cdot D_\sigma^v(u).$$

From equation (1) we know that the balance of a vertex is computed as a difference of the values of two vertices. We now show how the rate of change of the balance can be derived by substituting the rate of change of the values into equation (1).

Proposition 3. *For a vertex v and a joint strategy σ we have*

$$\partial_{-z} \text{Bal}_z^\sigma(v) = \begin{cases} \partial_{-z} \text{Val}_z^\sigma(v) - ([\bar{\sigma}(v) = \lambda(v)] + \beta \cdot \partial_{-z} \text{Val}_z^\sigma(\bar{\sigma}(v))) & \text{if } v \in V_{Max}, \\ -[\bar{\sigma}(v) = \lambda(v)] + \beta \cdot \partial_{-z} \text{Val}_z^\sigma(\bar{\sigma}(v)) - \partial_{-z} \text{Val}_z^\sigma(v) & \text{if } v \in V_{Min}. \end{cases}$$

Now that we have an expression for the rate of change of the balance of a vertex, we can compute how far z can be decreased from z_{i-1} before some vertex gets a negative balance. For each vertex v , the expression $\text{Bal}_{z_{i-1}}^{\sigma_i}(v) / \partial_{-z} \text{Bal}_z^{\sigma_i}(v)$ gives the amount that z can be decreased before v gets a negative balance, and so the minimum over all these ratios gives the amount that z can be decreased before some vertex gets a negative balance. It should also be clear that a vertex that achieves this minimum will be indifferent in the modified game when z is decreased by this amount. We can also show that this is the minimum value of z for which σ_i is optimal in G_z .

Proposition 4. *Let a joint strategy σ_i be optimal in the modified game $G_{z_{i-1}}$, and*

$$z_i = z_{i-1} - \min \left\{ \frac{\text{Bal}_{z_{i-1}}^\sigma(v)}{\partial_{-z} \text{Bal}_z^\sigma(v)} : v \in V \text{ and } \partial_{-z} \text{Bal}_z^\sigma(v) < 0 \right\}. \quad (3)$$

Then strategy σ is optimal in G_{z_i} , and it is not optimal in G_x for all $x < z_i$.

Until now, we have ignored the possibility of reaching a strategy σ in which there is more than one indifferent vertex. In LCP algorithms this is known as a degenerate step. In this case, the task is to find a strategy in which every indifferent vertex v satisfies $\partial_{-z} \text{Bal}_z^\sigma(v) > 0$, so that z can be decreased further. It is not difficult to prove that such a strategy can be reached by switching only the indifferent vertices. One method for degeneracy resolution is Bland’s rule, which uses the least index method to break ties, and another is to use lexicographic perturbations; both methods are well-known, and are also used with the simplex method for linear programming [4].

Algorithm 1. Lemke(G)

```

i := 0;  $\sigma_0 := \rho$ ;  $z_0 := \max\{-\text{Bal}^{\sigma_0}(v) : v \in V\}$ 
while  $z_i > 0$  do
     $\sigma_{i+1} := \sigma_i[\bar{\sigma}_i(v)/v]$  for some vertex  $v$  with  $\text{Bal}_{z_i}^{\sigma_i}(v) = 0$ 
     $z_{i+1} := z_i - \min\left\{\frac{\text{Bal}_{z_i}^{\sigma_{i+1}}(v)}{\partial_{-z} \text{Bal}_z^{\sigma_{i+1}}(v)} : v \in V \text{ and } \partial_{-z} \text{Bal}_z^{\sigma_{i+1}}(v) < 0\right\}$ 
     $i := i + 1$ 
end while

```

Lemke’s algorithm is shown as Algorithm 1. Since in each step we know that there is no value of $z < z_i$ for which σ_i is optimal in G_z and we decrease z in every step it follows that we can never visit the same strategy twice without violating the condition that the current strategy should be optimal in the modified game. Therefore the algorithm must terminate after at most $2^{|V|}$ steps, which corresponds to the total number of joint strategies. The algorithm can only terminate when z has reached 0, and G_0 is the same game as G . It follows that whatever strategy the algorithm terminates with must be optimal in the original game.

Theorem 3. *Algorithm 1 terminates, with a joint strategy σ that is optimal for G after at most $2^{|V|}$ iterations.*

Lemke’s algorithm for LCPs allows a free choice of *covering vector*, and in our description we used a unit covering vector. This can be generalised by giving a positive covering value to every vertex. If each vertex v has a covering value d_v then the modification of the left edges in Definition 2 becomes:

$$r_z^\lambda(v) = \begin{cases} r^\lambda(v) - d_v \cdot z & v \in V_{\text{Max}}, \\ r^\lambda(v) + d_v \cdot z & v \in V_{\text{Min}}. \end{cases}$$

The algorithm can then easily be modified to account for this altered definition.

4 The Cottle-Dantzig Algorithm for Discounted Games

The principle idea behind the Cottle-Dantzig algorithm is to maintain a set of vertices whose balance is non-negative. The algorithm begins with an arbitrary strategy, and it goes through a series of major iterations, where in each iteration one vertex is brought into the set of vertices with non-negative balances, while maintaining the non-negative balances of the vertices that are already in that set. It is clear that if such a task can be accomplished, then the algorithm will terminate after $|V|$ major iterations.

We require a method for bringing some distinguished vertex v into the set of vertices with a non-negative balance without the vertices currently in the set getting a negative balance in the process. To accomplish this we will modify the game by adding a bonus to the edge that the strategy currently chooses at v . We will then drive the bonus up from 0 while maintaining an optimal strategy for the modified game. Eventually the balance of v will become 0 in the modified game, at which point the strategy at v can be switched away from the edge with the bonus attached to it, and the bonus can be removed. We will prove that after this procedure v will have a positive balance.

In this section we will override many of the notations that were used to describe Lemke’s algorithm.

Definition 3 (Modified Game for the Cottle-Dantzig Algorithm). For a real number w , a joint strategy σ , and a distinguished vertex v , we define the game G_w to be the same as G but with a different reward on the edge chosen by σ at v . If σ chooses the left successor at v then the left reward function is defined, for every u in V , by:

$$r_w^\lambda(u) = \begin{cases} r^\lambda(u) + w & \text{if } u = v \text{ and } u \in V_{Max}, \\ r^\lambda(u) - w & \text{if } u = v \text{ and } u \in V_{Min}, \\ r^\lambda(u) & \text{otherwise.} \end{cases}$$

If σ chooses the right successor at v then r^p modified in a similar manner.

We begin the major iteration with a strategy σ_0 , a value $w_0 = 0$, and a set of vertices with non-negative balances P . The task is to raise w from 0 until $\text{Bal}_w^\sigma(v) = 0$, while maintaining the invariant that every vertex in P has a non-negative balance. This can be accomplished using methods that are similar to those used in Lemke’s algorithm. For every vertex in P we must compute how the balance of that vertex changes as w is increased. The following propositions are analogues of Propositions 2, 3, and 4.

Proposition 5. Consider a vertex u and a joint strategy σ . Suppose that v is the distinguished vertex. The rate of change $\partial_w \text{Val}_w^\sigma(u)$ is $D_\sigma^u(v)$.

Proposition 6. Consider a vertex u and a joint strategy σ in the game G_w . The rate of change $\partial_w \text{Bal}_w^\sigma(u)$ is:

$$\partial_w \text{Bal}_w^\sigma(u) = \begin{cases} \partial_w \text{Val}_w^\sigma(u) - \beta \cdot \partial_w \text{Val}_w^\sigma(\bar{\sigma}(u)) & \text{if } u \in V_{Max}, \\ \beta \cdot \partial_w \text{Val}_w^\sigma(\bar{\sigma}(u)) - \partial_w \text{Val}_w^\sigma(u) & \text{if } u \in V_{Min}. \end{cases}$$

Algorithm 2. Cottle-Dantzig(G, σ)

```

 $P := \emptyset$ 
while  $P \neq V$  do
   $i := 0$ ;  $w_0 := 0$ ;  $v :=$  Some vertex in  $V \setminus P$ 
  while  $\text{Bal}_{w_i}^\sigma(v) < 0$  do
     $w_{i+1} := w_i + \min\{-\frac{\text{Bal}_{w_i}^\sigma(u)}{\partial_w \text{Bal}_w^\sigma(u)} : u \in P \cup \{v\} \text{ and } \partial_w \text{Bal}_w^\sigma(u) < 0\}$ 
     $\sigma := \sigma[\bar{\sigma}(u)/u]$  for some vertex  $u$  with  $\text{Bal}_{w_i}^\sigma(v) = 0$ 
     $i := i + 1$ 
  end while
   $\sigma := \sigma[\bar{\sigma}(v)/v]$ ;  $P := P \cup \{v\}$ 
end while

```

Proposition 7. *Consider a modified game G_w , a joint strategy σ , and a set of vertices P which must not have negative balances. Let*

$$y = w + \min\left\{-\frac{\text{Bal}_w^\sigma(u)}{\partial_w \text{Bal}_w^\sigma(u)} : u \in P \cup \{v\} \text{ and } \partial_w \text{Bal}_w^\sigma(u) < 0\right\}.$$

No vertex in P has a negative balance in G_y . Moreover, one vertex in $P \cup \{v\}$ is indifferent, and for all values $x > y$ that vertex has a negative balance in G_x .

The process of raising w up from 0 until the balance of v is 0 in the modified game is the same as the process of decreasing z in Lemke’s algorithm, only using the different definitions from Propositions 5, 6, and 7. Once the balance of v has reached 0 we can stop increasing w . Since v is now indifferent we can switch it away from the edge that has the bonus attached to it. Once this has been done, the values of all vertices are no longer affected by w , since the edge to which it is attached is no longer chosen by the current strategy. Therefore we can remove the bonus and recover the original game. The major iteration then terminates with a strategy in which every vertex in $P \cup \{v\}$ has a non-negative balance, and the next major iteration can begin.

Theorem 4. *Algorithm 2 terminates, with the optimal joint strategy, after at most $2^{|V|}$ iterations.*

5 Exponential Lower Bounds

We show that both Lemke’s and the Cottle-Dantzig algorithms take exponentially many steps on the family of games shown in Figure 1. Max vertices are depicted as squares and Min vertices are depicted as circles. For every vertex, we define the right successor to be the vertex with the same owner as the vertex itself, and the left successor to be the vertex that belongs to the other player. Recall that the initial strategy for Lemke’s algorithm is the one that chooses the right successor for every vertex. When speaking about vertices in the game we often refer to either the leftmost or the rightmost vertex with a certain property.

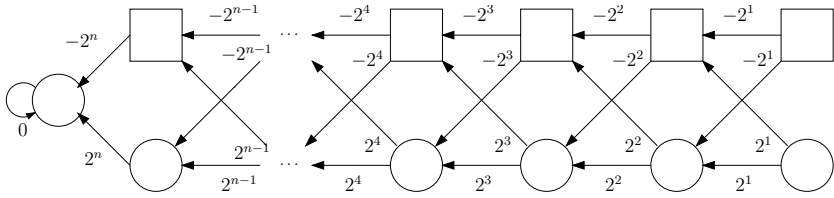


Fig. 1. The game \mathcal{G}_n

In this context, the vertex being referred to is the one that is furthest to the right or to the left in Figure 1.

For ease of exposition, we will describe the steps of the algorithm as if the discount factor was 1. Although this is forbidden by the definition of a discounted game, since the game contains one cycle, whose value is zero, the value of every vertex under every strategy will be finite. As long as the discount factor is chosen sufficiently close to 1, the algorithm will behave as we describe.

Note that the game graph is symmetric with respect to the line that separates the vertices of the two players. We frequently refer to a vertex and the vertex that it is opposite to, and hence we introduce the concept of vertex reflections. For a vertex v that is not the sink, we write \bar{v} to denote the reflection of v , that is the vertex belonging to the other player that is shown directly opposite v in Figure 1. We say that a joint strategy σ is symmetric if for all vertices v , the strategy σ chooses the right successor of v if and only if it chooses the right successor of \bar{v} . The initial strategy for Lemke’s algorithm is a symmetric strategy. Lemke’s algorithm always switches v directly before or after \bar{v} and so it can be seen as traversing through symmetric strategies.

Before discussing the modified games that Lemke’s algorithm constructs, we give a simple characterisation of when a vertex is switchable in the original game.

Proposition 8. *If σ is a symmetric joint strategy, then a vertex v is switchable if and only if the path from v has an even number of left edges.*

We now use this characterisation to give a simple formula for $\partial_{-z} \text{Bal}_z^\sigma(v)$ for every vertex v under every symmetric joint strategy σ .

Proposition 9. *If σ is a symmetric joint strategy, then $\partial_{-z} \text{Bal}_z^\sigma(v)$, the rate of change of the balance of a vertex v , is 1 if v is switchable, and -1 otherwise.*

Together, Propositions 8 and 9 imply that the parameter z can be set to the largest balance of a switchable vertex. We show that the largest balance will always belong to the rightmost switchable vertex.

Proposition 10. *Let σ be a symmetric joint strategy, v be the rightmost switchable Max vertex, and $z = -\text{Bal}^\sigma(v)$. Then no vertex in G_z is switchable, both v and the reflection of v are indifferent, and for every real number $y < z$, there is a switchable vertex in G_y .*

Proposition 10 implies that whenever Lemke’s algorithm is considering a symmetric joint strategy, it must choose a z so that the rightmost switchable vertex is indifferent. We show that this leads to an exponential number of switches.

Theorem 5. *Lemke’s algorithm performs $2^{n+1} - 2$ iterations on the game \mathcal{G}_n .*

The Cottle-Dantzig algorithm is sensitive to the order in which to bring the vertices into the non-negative set. We prove that there is an order that causes exponential-time behaviour for this algorithm. The sequence of strategies is similar to the sequence that Lemke’s algorithm follows.

Theorem 6. *Consider an order in which all Min vertices precede Max vertices, and Max vertices are ordered from right to left. The Cottle-Dantzig algorithm performs $2^{n+1} - 1$ iterations.*

We have shown that both algorithms can take an exponential number of steps on the discounted game \mathcal{G}_n . We argue that this also implies an exponential lower bound for parity and average-reward games. From the game \mathcal{G}_n we can obtain a parity game by replacing the reward $\pm 2^c$ with the priority c . The standard reductions [16,21,13] convert this parity game into average-reward and discounted games where priority c is replaced with reward $(-n)^c$, and the discount factor is chosen to be very close to 1. All arguments used to prove Theorems 5 and 6 continue to hold if rewards of magnitude 2^c are replaced with rewards of magnitude n^c , which implies the exponential lower bounds also hold for parity games and average-reward games.

6 Future Work

Our adaptation of Lemke’s algorithm for solving discounted games corresponds to its implementation in which the unit covering vector is used [6], and our lower bounds are specific to this choice. Similarly our lower bounds for the Cottle-Dantzig algorithm require a specific choice of ordering over the vertices. Randomizing these choices may exhibit better performance and should be considered.

Adler and Megiddo [1] studied the performance of Lemke’s algorithm for the LCPs arising from linear programming problems. They showed that, for randomly chosen linear programs and a carefully selected covering vector, the expected number of pivots performed by the algorithm is quadratic. A similar analysis for randomly chosen discounted games should be considered.

References

1. Adler, I., Megiddo, N.: A Simplex Algorithm whose Average Number of Steps is Bounded between Two Quadratic Functions of the Smaller Dimension. *Journal of the ACM* 32(4), 871–895 (1985)
2. Björklund, H., Vorobyov, S.: A Combinatorial Strongly Subexponential Strategy Improvement Algorithm for Mean Payoff Games. *Discrete Applied Mathematics* 155(2), 210–229 (2007)

3. Chung, S.J.: NP-Completeness of the Linear Complementarity Problem. *Journal of Optimization Theory and Applications* 60(3), 393–399 (1989)
4. Chvátal, V.: *Linear Programming*. Freeman, New York (1983)
5. Condon, A.: On Algorithms for Simple Stochastic Games. In: *Advances in Computational Complexity Theory. DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 13, pp. 51–73. American Mathematical Society (1993)
6. Cottle, R.W., Pang, J.-S., Stone, R.E.: *The Linear Complementarity Problem*. Academic Press, London (1992)
7. Emerson, E.A., Jutla, C.S., Sistla, A.P.: On Model-Checking for Fragments of μ -Calculus. In: Courcoubetis, C. (ed.) *CAV 1993. LNCS*, vol. 697, pp. 385–396. Springer, Heidelberg (1993)
8. Filar, J., Vrieze, K.: *Competitive Markov Decision Processes*. Springer, Heidelberg (1997)
9. Friedman, O.: A Super-Polynomial Lower Bound for the Parity Game Strategy Improvement Algorithm as We Know It. In: *Logic in Computer Science (LICS)*. IEEE, Los Alamitos (to appear, 2009)
10. Gärtner, B., Rüst, L.: Simple Stochastic Games and P-Matrix Generalized Linear Complementarity Problems. In: Liškiewicz, M., Reischuk, R. (eds.) *FCT 2005. LNCS*, vol. 3623, pp. 209–220. Springer, Heidelberg (2005)
11. Gillette, D.: Stochastic Games with Zero Stop Probabilities. In: *Contributions to the Theory of Games*, pp. 179–187. Princeton University Press, Princeton (1957)
12. Grädel, E., Thomas, W., Wilke, T. (eds.): *Automata, Logics, and Infinite Games. A Guide to Current Research. LNCS*, vol. 2500. Springer, Heidelberg (2002)
13. Jurdziński, M.: Deciding the Winner in Parity Games Is in $UP \cap co-UP$. *Information Processing Letters* 68(3), 119–124 (1998)
14. Jurdziński, M., Savani, R.: A Simple P-Matrix Linear Complementarity Problem for Discounted Games. In: Beckmann, A., Dimitracopoulos, C., Löwe, B. (eds.) *CIE 2008. LNCS*, vol. 5028, pp. 283–293. Springer, Heidelberg (2008)
15. Melekopoglou, M., Condon, A.: On the Complexity of the Policy Improvement Algorithm for Markov Decision Processes. *ORSA Journal on Computing* 6, 188–192 (1994)
16. Puri, A.: *Theory of Hybrid Systems and Discrete Event Systems*. PhD Thesis, University of California, Berkeley (1995)
17. Shapley, L.S.: Stochastic Games. *Proceedings of the National Academy of Sciences of the United States of America* 39(10), 1095–1100 (1953)
18. Stirling, C.: Local Model Checking Games (Extended abstract). In: Lee, I., Smolka, S.A. (eds.) *CONCUR 1995. LNCS*, vol. 962, pp. 1–11. Springer, Heidelberg (1995)
19. Svensson, O., Vorobyov, S.: Linear Complementarity and P-Matrices for Stochastic Games. In: Virbitskaite, I., Voronkov, A. (eds.) *PSI 2006. LNCS*, vol. 4378, pp. 409–423. Springer, Heidelberg (2007)
20. Vöge, J., Jurdziński, M.: A Discrete Strategy Improvement Algorithm for Solving Parity Games (Extended abstract). In: Emerson, E.A., Sistla, A.P. (eds.) *CAV 2000. LNCS*, vol. 1855, pp. 202–215. Springer, Heidelberg (2000)
21. Zwick, U., Paterson, M.: The Complexity of Mean Payoff Games on Graphs. *Theoretical Computer Science* 158, 343–359 (1996)