Noname manuscript No.
(will be inserted by the editor)

An exponential lower bound for Cunningham's rule

David Avis and Oliver Friedmann

the date of receipt and acceptance should be inserted later

Abstract In this paper we give an exponential lower bound for Cunningham's least recently considered (round-robin) rule as applied to parity games, Markov decision processes and linear programs. This improves a recent subexponential bound of Friedmann for this rule on these problems. The round-robin rule fixes a cyclical order of the variables and chooses the next pivot variable starting from the previously chosen variable and proceeding in the given circular order. It is perhaps the simplest example from the class of history-based pivot rules.

Our results are based on a new lower bound construction for parity games. Due to the nature of the construction we are also able to obtain an exponential lower bound for the round-robin rule applied to acyclic unique sink orientations of hypercubes (AUSOs). Furthermore these AUSOs are realizable as polytopes. We believe these are the first such results for history based rules for AUSOs, realizable or not.

The paper is self-contained and requires no previous knowledge of parity games.

Keywords simplex method, Cunningham's rule, parity games, acyclic unique sink orientations, Markov decision processes

Mathematics Subject Classification (2000) 90C05

Oliver Friedmann
University of Munich
E-mail: Oliver.Friedmann@ifi.lmu.de

David Avis
School of Informatics, Kyoto University, Kyoto, Japan and
School of Computer Science, McGill University, Montréal, Québec, Canada
E-mail: avis@cs.mcgill.ca

1 Introduction

The search for a polynomial time pivoting rule for the simplex method is as old as the method itself. Klee and Minty showed in 1972 [16] that Dantzig's original rule was exponential and similar results were soon found for most of the other known rules. All such lower bound constructions were based on variations of the deformed hypercube that appeared in Klee and Minty's original paper. The constructions have the property that some variables pivoted only a very few times - sometimes only once - in the exponential pivot path. This motivated Cunningham [5], Zadeh [24] and others to consider so-called history based rules. See [1] for a formal description of these and several other history based rules.

Cunningham's *least recently considered rule (round-robin rule)* assigns a cyclic order to the variables and remembers the last variable to enter the basis. The next entering variable is chosen to be the first allowable candidate starting from the last chosen variable and following the given circular order. Zadeh's *least entered rule* chooses the entering variable to be the candidate that has entered the basis least often. History based rules defeat the deformed hypercube constructions because they tend to average out how many times a variable pivots. This pseudo-random behaviour held out the possibility that they might be at worst subexponential, if not polynomial, since the random facet rule [15,17] is subexponential.

Friedmann gave the first evidence that history-based rules can sometimes be non-polynomial by showing the least entered rule [9] and the round-robin rule [10] are superpolynomial in the worst case. These results were obtained by first constructing a certain two person game, known as a parity game, for which the players (zero and one) follow a superpolynomial number of moves. These games are then related to Markov decision processes (MDPs) and finally to linear programs (LPs). It is shown that each strategy in the parity game corresponds to a vertex in the derived LP and improving from one strategy to the next corresponds to a pivot step in the LP.

In this paper we first obtain a new lower bound construction for parity games. We define a strategy improvement rule for player zero that corresponds to the least recently considered rule and show an exponential lower bound on the number of strategy improvements made to complete the game. Using the earlier transformations this shows an exponential lower bound for MDPs and LPs using this rule.

However the nature of the new construction allows us to do more. An acyclic unique sink orientation of a hypercube (AUSO) [22] is an orientation of the hypercube's edges so that the resulting directed graph has no cycles and each face of the hypercube has a unique sink (vertex of outdegree zero)[1]. LP pivot rules have natural analogues on AUSOs and their analysis has been the subject of several papers. For example in [18] a sub-exponential lower

[1] We do not consider here the more general unique sink orientations (USOs), which may contain cycles, and were introduced earlier by Stickney and Watson [21] and studied by Gärtner and Schurr [13], among others.

bound is given for the random edge pivot selection rule. We are able to show an exponential lower bound for the least recently considered rule on AUSOs, which are realizable as LPs.

The paper is organized as follows. In the next section we begin by defining parity games and policy iteration, and present the new lower bound construction. The longest part of the paper is a proof that this game requires an exponential number of moves before terminating. The section is concluded with some applications of this result to other types of games. In Section 3 we review the known connection between parity games and Markov decision processes. This gives rise to an exponential lower bound on MDPs that use an analogue of Cunningham's rule. In Section 4 we again exploit a known connection to obtain explicit LPs from the MDP examples. The least recently considered pivot rule on these LPs is shown to require an exponential number of steps. We turn to AUSOs in Section 5. Here we show that the parity game exhibited in Section 2 gives a natural acyclic orientation of a hypercube built on player zero's strategies. Cunningham's rule on this AUSO follows an exponentially long path. Furthermore we show that each AUSO we construct can be realized as a polytope. In fact the polytope is the one that arises from the same parity game after transforming it to an MDP and then to an LP. The paper concludes with some open problems for future research.

2 Parity Game Policy Iteration Lower Bound

This section is organized as follows. We first define parity games and how the general strategy improvement algorithms operate on them. Since some readers may not be familiar with this material we will illustrate them on an example that will later be generalized for the lower bound results. We then describe the lower bound construction and prove it correct. Finally, we show how to extend the results to related game classes.

2.1 Parity Games

A *parity game* is a tuple $G = (V, V_0, V_1, E, \Omega)$ where (V, E) forms a directed graph whose node set is partitioned into $V = V_0 \cup V_1$ with $V_0 \cap V_1 = \emptyset$, and $\Omega : V \to \mathbb{N}$ is the *priority function* that assigns to each node a natural number called the *priority* of the node. We assume the graph to be total, i.e. for every $v \in V$ there is a $w \in V$ s.t. $(v, w) \in E$.

We depict parity games as directed graphs where nodes owned by player 0 are drawn as circles and nodes owned by player 1 are drawn as rectangles; all nodes are labelled with their respective name and priority. An example of such a graph is shown in Figure 1. For each node the name, $a_2, F_1, t, \cdots$, is on top and the priority is underneath. For the moment we ignore the colours on the edges and that some edges are shown dashed. (For monochromatic figures, we ignore that some edges are bold and some are dashed.) A very important

property of this game is that the out-degree of each node belonging to player 0 is two. We call a game with this property a *binary game*.

We use infix notation vEw instead of $(v, w) \in E$ and define the set of all *successors* of v as $vE := \{w \mid vEw\}$. The size $|G|$ of a parity game $G = (V, V_0, V_1, E, \Omega)$ is defined to be the cardinality of E, i.e. $|G| := |E|$; since we assume parity games to be total w.r.t. E, this is a reasonable way to measure the size. The example has size 36 and $F_2E := \{h_2, e_2, d_2\}$.

The game is played between two players called 0 and 1: starting at a node $v_0 \in V$, they construct an infinite path through the graph as follows. If the construction so far has yielded a finite sequence $v_0 \dots v_n$ and $v_n \in V_i$ then player i selects a $w \in v_nE$ and the play continues with $v_0 \dots v_nw$. In the example a game may have started as the sequence $a_2g_2F_2d_2$ ending at a node owned by player 0. She can choose between F_2 and g_1 and could continue by appending either node to the sequence.

Every play has a unique winner given by the *parity* of the greatest priority that occurs infinitely often. The winner of the play $v_0v_1v_2 \dots$ is player i iff $\max\{p \mid \forall j \in \mathbb{N}\, \exists k \geq j : \Omega(v_k) = p\} \equiv_2 i$[2]. That is, player 0 tries to make an even priority occur infinitely often without any greater odd priorities occurring infinitely often, player 1 attempts the converse. In the example we may consider the infinite path $a_2g_2F_2d_2F_2d_2....$ In this case 6 is the largest priority that occurs infinitely often and player 0 wins since this number is even.

A *strategy* for player i is a – possibly partial – function $\sigma : V^*V_i \to V$, s.t. for all sequences $v_0 \dots v_n$ with $v_{j+1} \in v_jE$ for all $j = 0, \dots, n-1$, and all $v_n \in V_i$ we have: $\sigma(v_0 \dots v_n) \in v_nE$. A play $v_0v_1 \dots$ *conforms* to a strategy σ for player i if for all $j \in \mathbb{N}$ we have: if $v_j \in V_i$ then $v_{j+1} = \sigma(v_0 \dots v_j)$. Intuitively, conforming to a strategy means to always make those choices that are prescribed by the strategy. A strategy σ for player i is a *winning strategy* in node v if player i wins every play that begins in v and conforms to σ.

A strategy σ for player i is called *positional* if for all $v_0 \dots v_n \in V^*V_i$ and all $w_0 \dots w_m \in V^*V_i$ we have: if $v_n = w_m$ then $\sigma(v_0 \dots v_n) = \sigma(w_0 \dots w_m)$. That is, the choice of the strategy on a finite path only depends on the last node on that path. So in this case we need only specify $\sigma(v)$ for each $v \in V$. The set of positional strategies for player i is denoted by $\mathcal{S}_i(G)$.

Given a positional strategy σ and an edge (v, w), we define the *strategy update* $\sigma[(v, w)]$ by $\sigma[(v, w)](u) = \sigma(u)$ for all $u \neq v$ and $\sigma[(v, w)](v) = w$.

In the example a set of partial positional strategies σ, τ could consist of $\sigma(a_2) = g_2$, $\sigma(g_2) = F_2$, $\tau(F_2) = d_2$, $\sigma(d_2) = F_2$ and player 0 wins. Note that we do not need to give a strategy for out-degree one nodes, such as g_2, and will omit these in the sequel.

Recall that a *binary game* is one where each node belonging to player 0 has out-degree two. In this case player 0's positional strategy has a very simple representation. Suppose she owns n nodes and labels the out-edges for each of them 0 or 1 in any arbitrary way. Then her positional strategy can be

[2] $x \equiv_2 y$ if and only if x and y are congruent mod 2

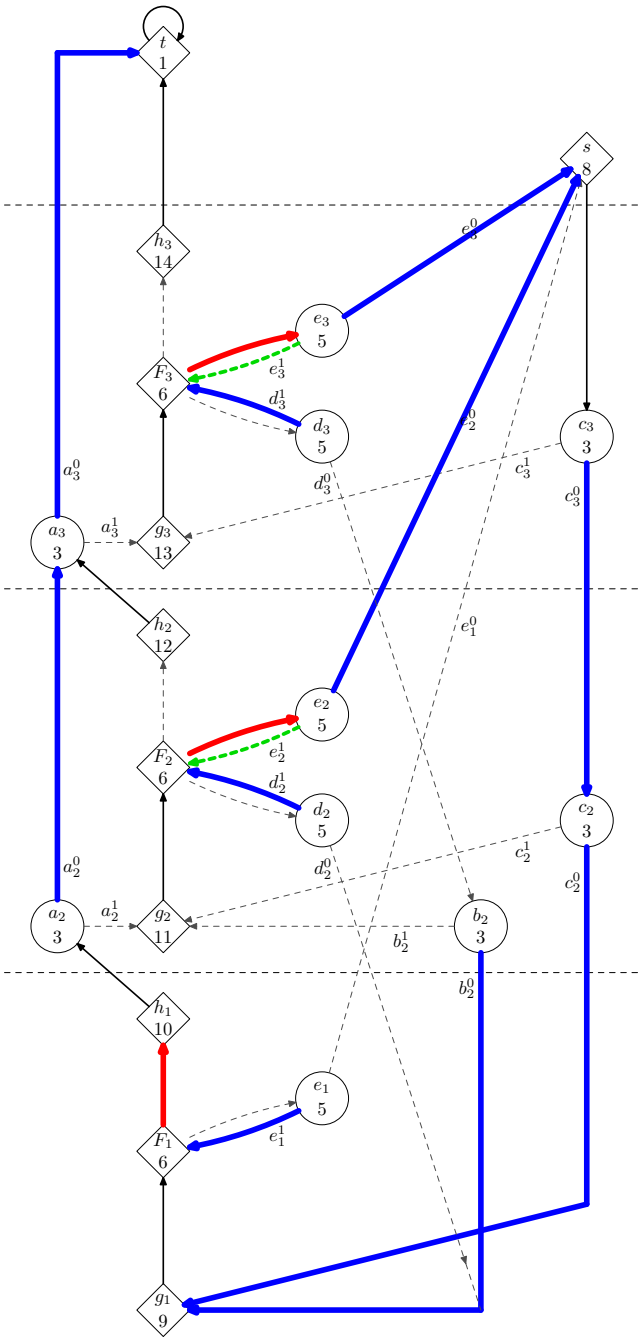

Fig. 1 Parity Game Lower Bound Graph G_3 (with $g_1 = b_1 = c_1$)

represented as a binary n-vector specifying for each node which edge is chosen in the strategy.

With G we associate two sets $W_0, W_1 \subseteq V$, where W_i is the set of all nodes v where player i wins the game G starting at v. Here we may restrict ourselves to positional strategies because it is well-known that a player has a (general) winning strategy if and only if she has a positional winning strategy for a given game. In fact, parity games enjoy positional determinacy meaning that for every node v in the game either $v \in W_0$ or $v \in W_1$ [7]. Furthermore, it is not difficult to show that, whenever player i has winning strategies σ_v for all $v \in U$ for some $U \subseteq V$, then there is also a single strategy σ that is winning for player i from every node in U.

The problem of solving a parity game is to compute W_0 and W_1 as well as corresponding winning strategies σ_0 and σ_1 for the players on their respective winning regions. In the example $F_3 \in W_1$ since player 1 can set $\sigma(F_3) = h_3$ and end up in the infinite loop on t with priority one which is odd.

2.2 Strategy Improvement

We describe here the basic definitions of the strategy improvement algorithm. For a given parity game $G = (V,\ V_0,\ V_1,\ E,\ \Omega)$, the *reward* of node v is defined as follows: $rew(v) := \Omega(v)$ if $\Omega(v) \equiv_2 0$ and $rew(v) := -\Omega(v)$ otherwise. The set of *profitable nodes* for player 0 resp. 1 is defined to be $V_\oplus := \{v \in V \mid \Omega(v) \equiv_2 0\}$ resp. $V_\ominus := \{v \in V \mid \Omega(v) \equiv_2 1\}$.

The *relevance ordering* $<$ on V is induced by Ω: $v < u :\iff \Omega(v) < \Omega(u)$. Additionally one defines the *reward ordering* $\prec$ on V by $v \prec u :\iff rew(v) < rew(u)$. In our construction, although priorities are not unique, they are unique on each cycle. Therefore on each cycle both orderings are total.

Let v be a node, σ be a positional player 0 strategy and τ be a positional player 1 strategy. Starting at v, there is exactly one path $\pi_{\sigma,\tau,v}$ that conforms to σ and τ. Since σ and τ are positional strategies, this path can be uniquely written as follows.

$$\pi_{\sigma,\tau,v} = v_1 \dots v_k (w_1 \dots w_l)^\omega$$

The superscript ω denotes an infinite cycle on the given vertex or vertices. Here $v_1 \dots v_k$ is a (possibly empty) non-repeating set of vertices and $w_1 \dots w_l$ is an infinite cycle with $w_1 > w_j$ for all $1 < j \leq l$. If $k \geq 1$ then $v = v_1$ otherwise v is a member of the cycle. In other words, the first node of the cycle is the one with the highest priority. Note that the uniqueness follows from the fact that all nodes on the cycle have different priorities.

Discrete strategy improvement relies on a more abstract description of such a play $\pi_{\sigma,\tau,v}$. In fact, we only consider the *dominating cycle node* w_1, the set of *more relevant nodes* – i.e. all $v_i > w_1$ – on the path to the cycle node, and the *length* k of the path leading to the cycle node.

The *node valuation of* v *w.r.t.* σ *and* τ is defined as follows.

$$\vartheta_{\sigma,\tau,v} := (w_1, \{v_i > w_1 \mid 1 \leq i \leq k\}, k)$$

Given a node valuation ϑ, we refer to w_1 as the *cycle component*, to $\{v_i > w_1 \mid 1 \leq i \leq k\}$ as the *path component*, and to k as the *length component* of ϑ.

In the example if we have $\sigma(a_3) = g_3$, $\tau(F_3) = d_3$ and $\sigma(d_3) = F_3$ then $\pi_{\sigma,\tau,h_2} = h_2 a_3 g_3 (F_3 d_3)^\omega$ and node valuation $\vartheta_{\sigma,\tau,h_2} := (F_3, \{h_2, g_3\}, 3)$. F_3 is the cycle component with dominating node F_3, $\{h_2, a_3, g_3\}$ is the path to the cycle and has length 3, $\{h_2, g_3\}$ are the more relevant nodes.

In order to compare node valuations with each other, we introduce a partial ordering $\prec$ on the set of node valuations. For this we first define a partial ordering on the path components. The basic idea is that we first compare winning priorities, then paths, then lengths. More precisely, for a set of nodes M, let $\Omega(M)$ denote the *priority occurrence mapping*:

$$\Omega(M) : p \mapsto |\{v \in M \mid \Omega(v) = p\}|$$

Let M and N be two distinct sets of vertices. If $\Omega(M) = \Omega(N)$ then M and N are not comparable and we write this $M \sim N$. Otherwise let p be the highest priority s.t. $\Omega(M)(p) \neq \Omega(N)(p)$. Then $M \prec N$ if $\Omega(M)(p) > \Omega(N)(p)$ and $p \equiv_2 1$, or $\Omega(M)(p) < \Omega(N)(p)$ and $p \equiv_2 0$. Otherwise $N \prec M$. We observe that in a game with all priorities unique $\prec$ defines a total order on the subsets of vertices.

In the example, if we compare $M = \{c_3, c_2, g_2\}$ with $N = \{h_1, a_2, g_2\}$, we have $\Omega(M) = \{3 \mapsto 2, 11 \mapsto 1\}$ and $\Omega(N) = \{10 \mapsto 1, 3 \mapsto 1, 11 \mapsto 1\}$. The priority with maximum value in which both sets differ is 10. Since this priority is even and occurs in N, we have $M \prec N$.

Now we are able to extend the partial ordering on sets of nodes to node valuations. If $u \prec v$ then $(u, M, e) \prec (v, N, f)$. If $rew(u) = rew(v)$, we define

$$(u, M, e) \prec (v, N, f) \iff \begin{cases} M \prec N \\ M \sim N,\ e < f \text{ and } u \in V_\ominus \\ M \sim N,\ e > f \text{ and } u \in V_\oplus \end{cases}$$

We write $(u, M, e) \sim (v, N, f)$ iff neither $(u, M, e) \prec (v, N, f)$ nor $(v, N, f) \prec (u, M, e)$. We write $(u, M, e) \preceq (v, N, f)$ to abbreviate $(u, M, e) \prec (v, N, f)$ or $(u, M, e) \sim (v, N, f)$.

We observe that if all priorities are unique then we have a total order on node valuations. For in this case if $rew(u) = rew(v)$ then $u = v$ and if $M \sim N$ then $M = N$. We cannot have $e = f$ for otherwise $(u, M, e) = (v, N, f)$.

The motivation behind the above ordering is a lexicographic measurement of the profitability of a positional play w.r.t. player 0: the most prominent part of a positional play is the cycle in which the plays eventually stays, and here it is the reward ordering on the dominating cycle node that defines the profitability for player 0. The second important part is the loopless path that leads to the dominating cycle node. Here, we measure the profitability of a loopless path by a *lexicographic* ordering on the *relevancy* of the nodes on path, applying the *reward* ordering on each component in the lexicographic ordering. Finally, we consider the length, and the intuition behind the definition is that, assuming we have an even-priority dominating cycle node, it is better to reach

the cycle fast whereas it is better to stay as long as possible out of the cycle otherwise.

In the example suppose σ and τ give rise to the paths $c_3c_2g_2(F_2d_2)^\omega$ and $h_1a_2g_2(F_2d_2)^\omega$. We have node valuations $\vartheta_{\sigma,\tau,h_1} := (F_2, \{h_1, g_2\}, 3)$, $\vartheta_{\sigma,\tau,c_3} := (F_2, \{g_2\}, 3)$ and $\vartheta_{\sigma,\tau,c_2} := (F_2, \{g_2\}, 2)$. We have

$$(F_2, \{g_2\}, 3) \prec (F_2, \{g_2\}, 2) \prec (F_2, \{h_1, g_2\}, 3)$$

The first $\prec$ is due to the fact that the length of the path to the cycle node is smaller in the first node valuation. The second $\prec$ is because the symmetric difference of the path components is h_1 with priority 10 which is even.

Given a player 0 strategy σ, it is player 1's goal to find a best response counter-strategy τ that minimizes the associated node valuations. A strategy τ is an *optimal counter-strategy* w.r.t. σ iff for every opponent strategy τ' and for every node v we have: $\vartheta_{\sigma,\tau,v} \preceq \vartheta_{\sigma,\tau',v}$.

Is is well-known that an optimal counter-strategy always exists and that it is efficiently computable.

Lemma 1 ([23]) *Let G be a parity game and σ be a player 0 strategy. An optimal counter-strategy for player 1 w.r.t. σ exists and can be computed in polynomial time.*

A fixed but arbitrary optimal counter-strategy will be denoted by τ_σ from now on. The associated *game valuation* Ξ_σ is a map that assigns to each node the node valuation w.r.t. σ and τ_σ:

$$\Xi_\sigma : v \mapsto \vartheta_{\sigma,\tau_\sigma,v}$$

Game valuations are used to measure the performance of a strategy of player 0. For a fixed strategy σ of player 0 and a node v, the associated valuation essentially states which is the worst cycle that can be reached from v conforming to σ as well as the worst loopless path leading to that cycle (also conforming to σ). We also write $v \prec_\sigma u$ to compare the Ξ_σ-valuations of two nodes, i.e. to abbreviate $\Xi_\sigma(v) \prec \Xi_\sigma(u)$.

A run of the strategy improvement algorithm can be expressed by a sequence of *improving* game valuations; a partial ordering on game valuations is quite naturally defined as follows:

$$\Xi \lhd \Xi' \;:\Longleftrightarrow\; (\Xi(v) \preceq \Xi'(v) \text{ for all } v \in V) \text{ and } (\Xi \neq \Xi')$$

Let σ be a strategy, $v \in V_0$ and $w \in vE$. We say that (v, w) is a σ-*improving switch* iff $\sigma(v) \prec_\sigma w$. We say that σ is *improvable* iff σ has an improving switch. We write I_σ to denote the set of improving switches and write $I_\sigma(v) = \{w \mid (v, w) \in I_\sigma\}$.

Again things become very simple for binary games. For any node v we can write $vE = \{\sigma(v), w\}$ and an improving switch selects edge (v, w) if $\sigma(v) \prec_\sigma w$. Clearly the notion of improving switch requires that the node valuations of $\sigma(v)$ and w are ordered by $\prec_\sigma$. The binary games we construct do not have unique priorities and $\prec$ defines only a partial order on the node valuations.

Recall that for binary games player 0's current strategy can be represented by a binary n-vector. An improving switch just flips one of the bits in this vector. The connection with paths on hypercubes now becomes apparent.

The improvement step from one strategy to the next is carried out by an *improvement rule*. It is a map $\mathcal{I}_G : \mathcal{S}_0(G) \to \mathcal{S}_0(G)$ s.t. $\Xi_\sigma \trianglelefteq \Xi_{\mathcal{I}_G(\sigma)}$ for every σ and additionally $\Xi_\sigma \triangleleft \Xi_{\mathcal{I}_G(\sigma)}$ if σ is improvable. We say that a function $\mathcal{I}_G : \mathcal{S}_0(G) \to \mathcal{S}_0(G)$ is a *standard improvement rule* iff it only selects improving switches for finding a successor strategy, i.e.

1. For every node $v \in V_0$ it holds that $\sigma(v) \preceq_\sigma \mathcal{I}_G(\sigma)(v)$.
2. If σ is improvable then there is a node $v \in V_0$ s.t. $\sigma(v) \prec_\sigma \mathcal{I}_G(\sigma)(v)$.

Jurdziński and Vöge [23] showed that an improving switch always exists for any non-optimal strategy σ. They also showed that improving σ by an arbitrary, non-empty selection of improving switches can only result in strategies with valuations strictly better than the valuation of σ.

Theorem 1 ([23]) *Let G be a parity game, σ be an improvable strategy and $\mathcal{I}_G$ be a standard improvement rule. Then $\Xi_\sigma \triangleleft \Xi_{\mathcal{I}_G(\sigma)}$.*

If a strategy is not improvable, the strategy improvement procedure comes to an end. The game has been solved. The winning sets for both players as well as associated winning strategies can be easily derived from the given valuation.

Theorem 2 ([23]) *Let G be a parity game and σ be a non-improvable strategy. Then the following holds:*

1. $W_0 = \{v \mid \Xi_\sigma(v) = (w, \_, \_) \text{ and } w \in V_\oplus\}$
2. $W_1 = \{v \mid \Xi_\sigma(v) = (w, \_, \_) \text{ and } w \in V_\ominus\}$
3. *σ is a winning strategy for player 0 on W_0*
4. *τ_σ is a winning strategy for player 1 on W_1*
5. *σ is $\trianglelefteq$-optimal*

Strategy improvement starts with an initial strategy σ and runs for a given improvement rule $\mathcal{I}$ as follows and returns an optimal player 0 strategy as outlined in the pseudo-code of Algorithm 1.

Algorithm 1 Strategy Improvement

```
1: procedure StandardStratIt(I, G, σ)
2:     while σ is improvable do
3:         σ ← I_G(σ)
4:     end while
5:     return σ.
6: end procedure
```

Given an initial strategy σ, a game G and a rule $\mathcal{I}$, the unique execution trace, called *run*, of strategy improvement is the sequence of strategies $\sigma_1, \ldots, \sigma_k$ s.t. $\sigma_1 = \sigma$, $\sigma_{i+1} = \mathcal{I}_G(\sigma_i)$ for all $i < k$, σ_k optimal and σ_i improvable

for all $i < k$. The *length* of the run is denoted by k and we say that strategy improvement *requires k iterations to find the optimal strategy.*

We call a parity game G (in combination with an initial strategy θ) a *sink game* iff the following two properties hold:

1. *Sink Existence*: there is a node v^* (called the *sink* of G) with v^*Ev^* and $\Omega(v^*) = 1$ reachable from all nodes; also, there is no other node w with $\Omega(w) \leq \Omega(v^*)$.
2. *Sink Seeking*: for each player 0 strategy σ with $\Xi_\theta \trianglelefteq \Xi_\sigma$ and each node w it holds that the cycle component of $\Xi_\sigma(w)$ equals v^*.

At this point the reader may wish to verify that the example is a sink game with $v^* = t$. Obviously, a sink game is won by player 1. Note that comparing node valuations in a sink game can be reduced to comparing the path components of the respective node valuations, for two reasons. First, the cycle component remains constant. Second, the path-length component equals the cardinality of the path component, because all nodes except the sink node are more relevant than the cycle node itself. In the case of a sink game, we will therefore identify node valuations with their path component.

Given a parity game G the sink existence property can be verified by standard graph algorithms. Given an initial strategy θ the sink seeking property can also be easily checked, as shown by the following lemma.

Lemma 2 ([8]) *Let G be a parity game with initial strategy θ fulfilling the sink existence property w.r.t. v^*. G is a sink game iff G is completely won by player 1 (i.e. $W_1 = V$) and for each node w it holds that the cycle component of $\Xi_\theta(w)$ equals v^*.*

We will make use of this lemma to show that our family of games are sink games indeed.

Let G be a sink game and $v, r \in V_G$. We define $\Xi_\sigma^{>r}(v)$ to be the path component of $\Xi_\sigma(v)$ by filtering the nodes which are more relevant than r, i.e.

$$\Xi_\sigma^{>r}(v) = \{u \in \Xi_\sigma(v) \mid \Omega(u) > \Omega(r)\}$$

It is easy to see that $\Xi_\sigma^{>r}(v) \prec \Xi_\sigma^{>r}(u)$ implies $\Xi_\sigma(v) \prec \Xi_\sigma(u)$.

We assume from now on that every game we consider is a sink game.

Cunningham's rule [5] is a *deterministic history based* pivot rule for selecting entering variables in the network simplex method. It fixes an initial ordering on all variables and then selects the entering variables in a round-robin fashion starting from the last entering variable selected. The history is simply to remember this variable. The rule can be adapted in a straightforward manner to other local improvement algorithms.

We describe Cunningham's pivoting rule in the context of parity games. We assume that we are given a total ordering $\prec$ on the player 0 edges of the parity game. The history is simply to record the last edge that has been applied.

Given a non-empty subset of player 0 edges $\emptyset \neq F \subseteq E_0$ and a player 0 edge $e \in E_0$, we define a *successor operator* as follows:

$$\mathrm{succ}_{\prec}(e, F) := \begin{cases} \min_{\prec}\{e' \in F \mid e \preceq e'\} & \text{if } \{e' \in F \mid e \preceq e'\} \neq \emptyset \\ \min_{\prec}\{e' \in F \mid e' \preceq e\} & \text{otherwise} \end{cases}$$

See Algorithm 2 for a pseudo-code specification of Cunningham's rule applied to parity games. The algorithm is based on a strategy-improving while loop. In every iteration, the algorithm selects an improving edge e to pivot on and an updated strategy σ. We call the sequence of (σ, e)-pairs that the algorithm goes through to find an unimprovable strategy the *trace* of the algorithm.

Algorithm 2 Cunningham's Improvement Algorithm

```
1: procedure RoundRobin(G, σ, ≺, e)
2:     while σ is improvable do
3:         e ← succ_≺(e, I_σ)
4:         σ ← σ[e]
5:     end while
6:     return σ.
7: end procedure
```

Let (σ_1, e_1), …, (σ_n, e_n) be a trace of the algorithm w.r.t. some selection ordering $\prec$. We write $(\sigma, e) \rightsquigarrow_{\prec} (\sigma', e')$ iff there are $i < j$ s.t. $(\sigma, e) = (\sigma_i, e_i)$ and $(\sigma', e') = (\sigma_j, e_j)$.

In the original specification of Cunningham's rule [5] it is assumed that the ordering on the edges and the initial edge e is given as part of the input. In fact, we know that the asymptotic behaviour of Cunningham's improvement rule highly depends on the ordering used, at least in the world of parity games and strategy improvement for games in general. We have the following theorem which is easy to verify (the idea is that there is at least one improving switch towards the optimal strategy in each step).

Theorem 3 *Let G be a parity game with n nodes and σ_0 be a strategy. There is a sequence of strategies $\sigma_0, \sigma_1, \ldots, \sigma_N$ and a sequence of* different *switches $e_1, e_2, \ldots, e_N$ with $N \leq n$ s.t. σ_N is optimal, $\sigma_{i+1} = \sigma_i[e_{i+1}]$ and e_{i+1} is an σ_i-improving switch.*

Since all switches are different in the sequence, it follows immediately that there is always a way to select an ordering that results in a linear number of pivoting steps to solve a parity game with Cunningham's improvement rule. However, there is no obvious method to efficiently find such an ordering. In order to derive a lower bound we are entitled to give both the input graph and the ordering to be used.

2.3 Lower Bound Construction

Our lower bound construction is a natural generalization of the parity game G_3, shown in Figure 1, that we used throughout the last subsection. For each $n \geq 3$ we define the parity game $G_n = (V_0 \cup V_1, V_0, V_1, E, \Omega)$ as follows.

$$V_0 := \{a_i, c_i, d_i \mid 1 < i \leq n\} \cup \{b_i \mid 1 < i < n\} \cup \{e_i \mid 1 \leq i \leq n\}$$
$$V_1 := \{F_i \mid 1 \leq i \leq n\} \cup \{g_i, h_i \mid 1 \leq i \leq n\} \cup \{s, t\}$$

Figure 2 defines the edge sets and the priorities of G_n. For convenience of notation, we identify the node names a_{n+1} with t, b_1 with g_1, and c_1 with g_1. Explicit constructions of G_n for small n are available online [11].

Node	Successors	Priority	Node	Successors	Priority
a_i	g_i, a_{i+1}	3	F_i	d_i if $i > 1$, h_i, e_i	6
b_i	g_i, b_{i-1}	3	g_i	F_i	$2 \cdot i + 7$
c_i	g_i, c_{i-1}	3	h_i	a_{i+1}	$2 \cdot i + 8$
d_i	F_i, b_{i-1}	5	s	c_n	8
e_i	F_i, s	5	t	t	1

Fig. 2 Parity Game Lower Bound Graph

Lemma 3 *For every n, the game G_n is a binary sink parity game.*

To avoid special cases we assume $n \geq 3$. It is easy to verify that the total number of nodes in G_n is $8n - 3$, the total number of edges is $15n - 9$, the number of different priorities is $2n + 5$ and the highest priority is $2n + 8$. Therefore we have $|G_n| \in \mathcal{O}(n)$.

We refer to the edges of player 0 by the names given in Figure 3. For convenience of notation, we write $\sigma(a_i) = j$ to indicate that $a_i^j \in \sigma$ etc.

Name	Node	Successor	Name	Node	Successor
a_i^1	a_i	g_i	a_i^0	a_i	a_{i+1}
b_i^1	b_i	g_i	b_i^0	b_i	b_{i-1}
c_i^1	c_i	g_i	c_i^0	c_i	c_{i-1}
d_i^1	d_i	F_i	d_i^0	d_i	b_{i-1}
e_i^1	e_i	F_i	e_i^0	e_i	s

Fig. 3 Parity Game Lower Bound Player 0 Edges

From a high-level point of view, a run of the strategy improvement algorithm mimics the counting process of a binary counter, yielding an exponential number of steps. Obviously, the specifics of the run depend on our choice of the edge-ordering. Every strategy that we obtain during a run corresponds to the state of the binary counter. However, a single increment step of the binary counter corresponds to several consecutive improvements in strategy

improvement. These intermediate steps can be partitioned into well-defined *phases*.

Before describing the phases in detail, consider the layout of the game graph. It is separated into uniform layers that correspond to the different bits of the binary counter. The first and the last bit have less nodes than all the other bits. We could include the additional nodes in the game graph but they would be of no use for the counting process. Every (disregarding the first and the last) such layer contains five player 0 nodes a_i, b_i, c_i, d_i, e_i, and three player 1 nodes F_i, g_i, h_i.

The general construction extends Figure 1 in a natural way. We use terms such as up, down, left, right, etc. based on the layout used in the figure. The a_i-nodes build up a ladder-like structure that connect layers with each other, starting from the least-significant to the most-significant bit. For every layer, player 0 has the choice to either enter the layer or to directly pass on to the next layer. By making g_i highly unprofitable and h_i highly profitable, it follows that it will only be profitable for player 0 to enter a layer, if player 1 moves from F_i to h_i. This corresponds to a set bit.

Player 0 can force player 1 to move from F_i to h_i by moving from both d_i and e_i to F_i. This is due to the fact the game is a sink game and so the optimal counter-response to a strategy cannot be moving into any other cycle than t.

The other two remaining nodes of player 0, b_i and c_i, build up ladder-like structures as well that connect layers with each other, but this time starting from the most-significant to the least-significant bit. The nodes d_i and e_i have direct access to these additional two ladders which will allow them to get reset, corresponding to unsetting a set bit in the binary counter.

Next, we explain, from a high-level point of view, how the intermediate phases contribute to incrementing the counter. At beginning of phase 1 the switches have the following settings depending on the value in the binary counter. All a_i and e_i point upwards if and only if bit i is zero, all d_i point to the left, and both ladder b and c move down to the least significant set bit in the counter. We initiate the binary counter at $00\cdots001$ which corresponds to the *initial strategy* $\{a^0_*, b^0_*, c^0_*, d^1_*, e^1_1, e^0_{*>1}\}$ (which is depicted by the bold edges in the figure).

1. We apply improving switches in this phase s.t. all e_i point left, exactly all those d_i point to the left that correspond to a set bit in the current or in the next counter state, and update the ladder b s.t. it moves down to the least significant set bit in the next counter state.
2. In phase 2, we update the ladder c s.t. it moves down to least significant set bit in the next counter state.
3. In phase 3, we apply improving switches s.t. exactly all those e_i point left that correspond to a set bit in the next counter state.
4. In phase 4, we apply improving switches s.t. all d_i point left.
5. In phase 5, we update the a_i ladder to point to the right at exactly those bits which are set in the next counter state.

After completing phase 5, the counter has been incremented by one and we can start with phase 1 again.

Phases 1-5 finish when the counter reaches all ones. The a_i nodes point to the right, the d_i and e_i nodes point to the left, the c chain leads down to g_2, the b chain leads down to g_1 and all F_i nodes point vertically up. This corresponds to the strategy $\{a_*^1, b_*^0, c_*^0, d_*^1, e_*^1\}$ with the exception that c_2^1 is chosen instead of c_2^0. In fact replacing c_2^1 by c_2^0 is an improving switch and the only such switch. Therefore it will be chosen by Algorithm 2. At this point Player 0 has no improving switches and loses the game. We call $\{a_*^1, b_*^0, c_*^0, d_*^1, e_*^1\}$ the *terminal strategy* of the game.

Next we specify a total ordering of player 0 edges to be used in Algorithm 2.

$$\underbrace{\{b_*^*, d_*^0, e_*^1\}}_{\text{Phase 1}} \prec \underbrace{\{c_*^*\}}_{\text{Phase 2}} \prec \underbrace{\{e_*^0\}}_{\text{Phase 3}} \prec \underbrace{\{d_*^1\}}_{\text{Phase 4}} \prec \underbrace{\{a_*^*\}}_{\text{Phase 5}}$$

The detailed ordering is as follows:

$$\begin{aligned} &e_1^1 \prec d_2^0 \prec e_2^1 \prec b_2^1 \prec b_2^0 \prec \ldots \prec d_{n-1}^0 \prec e_{n-1}^1 \prec b_{n-1}^1 \prec b_{n-1}^0 \prec d_n^0 \prec e_n^1 \prec \\ &c_2^0 \prec c_2^1 \prec \ldots \prec c_n^0 \prec c_n^1 \prec e_1^0 \prec e_2^0 \prec \ldots \prec e_n^0 \prec \\ &d_2^1 \prec d_3^1 \prec \ldots \prec d_n^1 \prec a_n^1 \prec a_n^0 \prec \ldots \prec a_2^1 \prec a_2^0 \end{aligned} \tag{1}$$

We indicate by phases 1-5 for which phase the specific ordering is relevant.

For each $n \geq 3$ we define P_n to be the sequence of improving switches generated by Algorithm 2 following this edge ordering, starting at the initial strategy and ending at the terminal strategy. Our goal will be to show that the sequence P_n has exponential length in n.

Before proceeding with the proof let us apply Algorithm 2 to the example in Figure 1. The edge ordering is

$$e_1^1, d_2^0, e_2^1, b_2^1, b_2^0, d_3^0, e_3^1, c_2^0, c_2^1, c_3^0, c_3^1, e_1^0, e_2^0, e_3^0, d_2^1, d_3^1, a_3^1, a_3^0, a_2^1, a_2^0$$

Figure 1 corresponds to the initial state 001 of the binary counter, The current strategy for player 0 is shown by the blue edges, and that for player 1 by the red edges. Improving edges for player 0 are shown in dotted green and the other non-strategy edges are shown in dotted black. (Coloured edges show in bold on monochromatic printing.)

Note that since bits 2 and 3 of the counter (reading from right to left) are set to zero the strategy edges for a_2, a_3, e_2 and e_3 all point upwards. Each d_i edge points to F_i, and the b and c ladders point to the first bit, which is the least bit set. The improving edges for player zero are e_2^1, e_3^1 and e_2^1 is chosen as it comes first in order. Since player 0 stands to win on the infinite loop $(e_2 F_2)$ player 1 counters by changing the strategy on F_2 to point to h_2. The first five phases involve nine player 0 moves and are given in Figure 4. Note in some cases player 1 does not make a response as the position is still winning for him. At the end of the nine moves the counter has moved to 010 and we are back to the settings required to initiate Phase 1.

Phase	Improving Edges	Selected Edge	Player 1 Response
1	e_2^1, e_3^1	e_2^1	$F_2 h_2$
1	$a_2^1, b_2^1, c_2^1, e_3^1$	b_2^1	
1	$a_2^1, c_2^1, d_3^0, e_3^1$	d_3^0	
1	a_2^1, c_2^1, e_3^1	e_3^1	$F_3 d_3$
2	a_2^1, c_2^1, d_3^1	c_2^1	
3	$a_2^1, d_3^1, e_1^0, e_3^0$	e_3^0	
3	a_2^1, d_3^1, e_1^0	e_1^0	
4	a_2^1, d_3^1	d_3^1	$F_3 e_3$
5	a_2^1, e_3^1	a_2^1	$F_1 e_1$

Fig. 4 Binary counter moving from 001 to 010

Continuing in this fashion for a total of 36 improving switches by player 0 we arrive at the terminating position where she loses from each node and has no further improving edges. A complete simulation is given on the website [11].

2.4 Lower Bound Proof

In this section we prove the fundamental result of the paper.

Theorem 4 *The sequence P_n of improving switches followed by Algorithm 2 in G_n from the initial strategy to the terminal strategy using the ordering (1) has length at least 2^n where G_n has size $O(n)$.*

First, we introduce notation to succinctly describe binary counters. It will be convenient for us to consider counter configurations with an *infinite* tape, where unused bits are zero. The set of n-bit configurations is formally defined as $\mathcal{B}_n = \{\mathfrak{b} \in \{0,1\}^\infty \mid \forall i > n : \mathfrak{b}_i = 0\}$.

Any given $\mathfrak{b} \in \mathcal{B}_n$ corresponds to a tuple $(\mathfrak{b}_n, \ldots, \mathfrak{b}_1)$, with $\mathfrak{b}_1$ being the least and $\mathfrak{b}_n$ being the most significant bit. By $\mathbf{0}$, we denote the configuration in which all bits are zero, and by $\mathbf{1}_n$, we denote the configuration in which the first n bits are one. We write $\mathcal{B} = \bigcup_{n>0} \mathcal{B}_n$ to denote the set of all counter configurations.

The *integer value* of a $\mathfrak{b} \in \mathcal{B}$ is defined as usual, i.e. $|\mathfrak{b}| := \sum_{i>0} \mathfrak{b}_i \cdot 2^{i-1} < \infty$. For two $\mathfrak{b}, \mathfrak{b}' \in \mathcal{B}$, we induce the lexicographic linear ordering $\mathfrak{b} < \mathfrak{b}'$ by $|\mathfrak{b}| < |\mathfrak{b}'|$. It is well-known that $\mathfrak{b} \in \mathcal{B} \mapsto |\mathfrak{b}| \in \mathbb{N}$ is a bijection. For $\mathfrak{b} \in \mathcal{B}$ let $\mathfrak{b}^\oplus$ denote the unique $\mathfrak{b}'$ s.t. $|\mathfrak{b}'| = |\mathfrak{b}| + 1$.

Given a configuration $\mathfrak{b}$, we access the *least unset bit* by $\mu(\mathfrak{b}) = \min\{j \mid \mathfrak{b}_j = 0\}$ and the *least set bit* by $\nu(\mathfrak{b}) = \min\{j \mid \mathfrak{b}_j = 1\}$. Let $\mathfrak{b}^\mu$ denote $\mathfrak{b}[\mu(\mathfrak{b}) \mapsto 1]$.

We are now ready to formulate the conditions for strategies that fulfill one of the five phases along with the improving edges. See Figure 5 for a complete description (with respect to a bit configuration $\mathfrak{b}$). We say that a strategy σ is a *phase p strategy with configuration* $\mathfrak{b}$ iff every node is mapped by σ to a choice included in the respective cell of the table.

Phases 1 and 5 are multi-step phases and can therefore be divided into sub-phases. Phase 1 can be sub-divided by a counting variable $j \geq \mu(\mathfrak{b})$ "looping" through all indices greater or equal to the least unset bit and a state variable $k \in \{1,2,3\}$ determining whether the respective d_i and e_i have already been switched. Phase 5 is sub-divided by a counting variable $j \leq \mu(\mathfrak{b})$.

Phase	1: $\exists j \geq \mu(\mathfrak{b}), k \in \{1,2,3\}$	2	3	4	5: $\exists j \leq \mu(\mathfrak{b})$
$\sigma(a_i)$	$\mathfrak{b}_i$	$\mathfrak{b}_i$	$\mathfrak{b}_i$	$\mathfrak{b}_i$	$\begin{cases}\mathfrak{b}_i & \text{if } i \leq j\\ \mathfrak{b}_i^{\oplus} & \text{if } i > j\end{cases}$
$\sigma(b_i)$	$\begin{cases}\mathbb{1}_{i=\mu(\mathfrak{b})} & \text{if } j > i\\ \mathbb{1}_{i=\nu(\mathfrak{b})} & \text{otherwise}\end{cases}$	$\mathbb{1}_{i=\mu(\mathfrak{b})}$	$\mathbb{1}_{i=\mu(\mathfrak{b})}$	$\mathbb{1}_{i=\mu(\mathfrak{b})}$	$\mathbb{1}_{i=\mu(\mathfrak{b})}$
$\sigma(c_i)$	$\mathbb{1}_{i=\nu(\mathfrak{b})}$	$\mathbb{1}_{i=\nu(\mathfrak{b})}$	$\mathbb{1}_{i=\mu(\mathfrak{b})}$	$\mathbb{1}_{i=\mu(\mathfrak{b})}$	$\mathbb{1}_{i=\mu(\mathfrak{b})}$
$\sigma(d_i)$	$\begin{cases}\mathfrak{b}_i^{\mu} & \text{if } j > i \vee (j = i \wedge k > 1)\\ 1 & \text{otherwise}\end{cases}$	$\mathfrak{b}_i^{\mu}$	$\mathfrak{b}_i^{\mu}$	$1,\ \mathfrak{b}_i^{\mu}$	1
$\sigma(e_i)$	$\begin{cases}1 & \text{if } j > i \vee (j = i \wedge k > 2)\\ \mathfrak{b}_i & \text{otherwise}\end{cases}$	1	$1,\ \mathfrak{b}_i^{\oplus}$	$\mathfrak{b}_i^{\oplus}$	$\mathfrak{b}_i^{\oplus}$

Fig. 5 Policy Phases

The following lemma computes an optimal counter-strategy along with the associated valuations of the nodes F_i given a strategy σ belonging to one of the phases.

Lemma 4 *Let $n \geq 3$ and σ be a strategy belonging to one of the phases P w.r.t. $\mathfrak{b}$. Let $k = \max(\{i \mid \sigma(a_i) \neq \mathfrak{b}_i^{\oplus}\} \cup \{1\})$. Then the following holds:* $(\tau_\sigma(F_i), \Xi_\sigma^{>6}(F_i)) =$

$$\begin{cases}
(h_i, \{g_j, h_j \mid \mathfrak{b}_j = 1, j > i\} \cup \{h_i\}) & \text{if } \sigma(d_i) = \sigma(e_i) = 1\\
(e_i, \{g_j, h_j \mid \mathfrak{b}_j = 1\} \cup \{s\}) & \text{if } \sigma(e_i) = 0 \wedge P = 1\\
(d_i, \{g_j, h_j \mid \mathfrak{b}_j^{\oplus} = 1\}) & \text{if } \sigma(d_i) = 0 \wedge (\sigma(e_i) = 1 \vee P \neq 1)\\
(e_i, \{g_j, h_j \mid \mathfrak{b}_j^{\oplus} = 1\} \cup \{s\}) & \text{if } \sigma(e_i) = 0 \wedge \sigma(d_i) = 1 \wedge P \neq 1 \wedge i \geq k\\
(h_i, \{g_j, h_j \mid \mathfrak{b}_j = 1, j > i\} \cup \{h_i\}) & \text{if } \sigma(e_i) = 0 \wedge \sigma(d_i) = 1 \wedge P \neq 1 \wedge i < k = \mu(\mathfrak{b})\\
(h_i, \{g_j, h_j \mid \mathfrak{b}_j^{\oplus} = 1 \vee i < j < k\} \cup \{s, h_i, g_k\}) & \text{if } \sigma(e_i) = 0 \wedge \sigma(d_i) = 1 \wedge P \neq 1 \wedge i < k < \mu(\mathfrak{b})
\end{cases}$$

Before we prove the lemma, we'll explain informally what the counter strategy and the corresponding valuation looks like. Since we are dealing with sink games, it suffices to only consider the path component. Particularly, we only consider the nodes in the path component that have a priority higher than 6. We will explain the lemma by going through the various cases above.

- If $\sigma(d_i) = \sigma(e_i) = 1$ it follows regardless of the phase and i that $\tau_\sigma(F_i) = h_i$. It is also easy to see that the a_j-ladder structure is well-behaved throughout all phases in the sense that if $\sigma(d_j) = \sigma(e_j) = 1$, a_{j+1} will only go to the right iff $\sigma(d_{j+1}) = \sigma(e_{j+1}) = 1$. This is certainly true at the beginning of phase 1, throughout phase 1 and in phase 2. In phase 3 we set some of

the $\sigma(e_j) = 0$, but in a bottom-up fashion, hence the a_j-ladder remains well-behaved. Phase 4 sets some of the $\sigma(d_j) = 1$, but none of those where $\sigma(a_j) = 1$. Phase 5 updates a_j in a top-down fashion, so everything remains well-behaved.

- If we are in phase 1 and $\sigma(e_i) = 0$, it means that the path starting in e_i goes down to s via the c-lane to the least set bit and then up through all set bits. During phase 1, the valuation of s will not change. The valuation of all a_{i+1} will be better than the valuation of h_i throughout the phase, hence, the counter strategy will never choose h_i over e_i. Now, if $\sigma(d_i) = \sigma(e_i) = 0$, the counter strategy will still prefer e_i over d_i, because by definition d_i will be accessing the set bit $\mu(\mathfrak{b})$ as well as all higher set bits and hence have a better valuation than s.
- Assume that $\sigma(d_i) = 0$. If we are in phase 1 and $\sigma(e_i) = 1$, it follows that the counter strategy goes from F_i to d_i, since exiting through h_i has a better valuation than coming from the bottom up through d_i. If we are not in phase 1, it similarly follows for the counter strategy to prefer d_i over h_i if $\sigma(e_i) = 1$.
- In all the remaining cases we have $\sigma(d_i) = 1$ and $\sigma(e_i) = 0$. Whether F_i prefers e_i over h_i now depends on a couple of subtle factors. We also assume that we are in a phase other than 1. Let k denote the greatest index for which the a-lane has not been fixed. Recall that this necessarily is a number less or equal to $\mu(\mathfrak{b})$ (since $\mathfrak{b}$ and $\mathfrak{b}^{\oplus}$ coincide in all higher bits). If the index i is greater than k, then the valuation of h_i is greater than of e_i since the lane has been fixed for this region already. For all other i, it is less, hence, F_i prefers h_i over e_i in that case.

Proof Let $n \geq 3$ and σ be a strategy belonging to one of the phases w.r.t. $\mathfrak{b}$. Let $k = \max(\{i \mid \sigma(a_i) \neq \mathfrak{b}_i^{\oplus}\} \cup \{1\})$.

- *Phase 1*: First, if $\sigma(d_i) = \sigma(e_i) = 1$, it follows that $\tau_\sigma(F_i) = h_i$ since we have a sink game. Furthermore it follows by definition of phase 1 that

$$\Xi_\sigma^{>6}(F_i) = \{g_j, h_j \mid \mathfrak{b}_j = 1, j > i\} \cup \{h_i\}$$

  Otherwise, $i \geq \mu(\mathfrak{b})$, since if one of $\sigma(d_i)$ and $\sigma(e_i)$ is 0, then $i \geq \mu(\mathfrak{b})$. It follows by definition of phase 1 that

$$\Xi_\sigma^{>6}(s) = \{g_j, h_j \mid \mathfrak{b}_j = 1\} \cup \{s\}$$
$$\Xi_\sigma^{>6}(h_i) = \{g_j, h_j \mid \mathfrak{b}_j = 1, j > i\} \cup \{h_i\}$$

  and if $\sigma(d_i) = 0$ that

$$\Xi_\sigma^{>6}(d_i) = \{g_j, h_j \mid \mathfrak{b}_j^{\oplus} = 1\}$$

  Second (a), if $\sigma(d_i) = 1$ and $\sigma(e_i) = 0$, it follows that $e_i \prec_\sigma h_i$ and therefore $\tau_\sigma(F_i) = e_i$ and

$$\Xi_\sigma^{>6}(F_i) = \{g_j, h_j \mid \mathfrak{b}_j = 1\} \cup \{s\}$$

Second (b), if $\sigma(d_i) = 0$ and $\sigma(e_i) = 1$, it follows that $d_i \prec_\sigma h_i$ and therefore $\tau_\sigma(F_i) = d_i$ and

$$\Xi_\sigma^{>6}(F_i) = \{g_j, h_j \mid \mathfrak{b}_j^\oplus = 1\}$$

Second (c), if $\sigma(d_i) = \sigma(e_i) = 0$, it follows that $e_i \prec_\sigma d_i \prec_\sigma h_i$ and therefore $\tau_\sigma(F_i) = e_i$ and

$$\Xi_\sigma^{>6}(F_i) = \{g_j, h_j \mid \mathfrak{b}_j = 1\} \cup \{s\}$$

– *Phase 2*: First, if $\mathfrak{b}_i^\mu = 1$, it follows that $\sigma(d_i) = \sigma(e_i) = 1$ by definition of phase 2. Since we have a sink game, it follows that $\tau_\sigma(F_i) = h_i$. Furthermore it follows by definition of phase 2 that

$$\Xi_\sigma^{>6}(F_i) = \{g_j, h_j \mid \mathfrak{b}_j = 1, j > i\} \cup \{h_i\}$$

Second, if $\mathfrak{b}_i^\mu = 0$, it follows that $i > \mu(\mathfrak{b})$ and from definition of phase 2 that $\sigma(d_i) = 0$, $\sigma(e_i) = 1$, and

$$\begin{aligned}\Xi_\sigma^{>6}(h_i) &= \{g_j, h_j \mid \mathfrak{b}_j = 1, j > i\} \cup \{h_i\}\\ \Xi_\sigma^{>6}(b_{i-1}) &= \{g_j, h_j \mid \mathfrak{b}_j^\oplus = 1\}\end{aligned}$$

Hence, $b_{i-1} \prec_\sigma h_i$ and therefore $\tau_\sigma(F_i) = d_i$ and

$$\Xi_\sigma^{>6}(F_i) = \{g_j, h_j \mid \mathfrak{b}_j^\oplus = 1\}$$

– *Phase 3 & 4*: First, if $\mathfrak{b}_i^\oplus = 1$, it follows that $\sigma(d_i) = \sigma(e_i) = 1$ by definition of phase 3 and phase 4. Since we have a sink game, it follows that $\tau_\sigma(F_i) = h_i$. Furthermore it follows by definition of phase 3 and phase 4 that

$$\Xi_\sigma^{>6}(F_i) = \{g_j, h_j \mid \mathfrak{b}_j = 1, j > i\} \cup \{h_i\}$$

Second, if $\mathfrak{b}_i^\mu = 0$, it follows that $i > \mu(\mathfrak{b})$. By definition of phase 3 and phase 4, we have

$$\begin{aligned}\Xi_\sigma^{>6}(h_i) &= \{g_j, h_j \mid \mathfrak{b}_j = 1, j > i\} \cup \{h_i\}\\ \Xi_\sigma^{>6}(b_{i-1}) &= \{g_j, h_j \mid \mathfrak{b}_j^\oplus = 1\}\\ \Xi_\sigma^{>6}(s) &= \{g_j, h_j \mid \mathfrak{b}_j^\oplus = 1\} \cup \{s\}\end{aligned}$$

It follows that $b_{i-1} \prec_\sigma s \prec_\sigma h_i$ and hence $\tau_\sigma(F_i) = d_i$ and

$$\Xi_\sigma^{>6}(F_i) = \{g_j, h_j \mid \mathfrak{b}_j^\oplus = 1\}$$

if $\sigma(d_i) = 0$, and $\tau_\sigma(F_i) = e_i$ and

$$\Xi_\sigma^{>6}(F_i) = \{g_j, h_j \mid \mathfrak{b}_j^\oplus = 1\} \cup \{s\}$$

if $\sigma(d_i) = 1$.

Third, if $i < \mu(\mathfrak{b}) = k$, we show by backward induction on i that $\tau_\sigma(F_i) = h_i$, which implies that

$$\Xi_\sigma^{>6}(F_i) = \{g_j, h_j \mid \mathfrak{b}_j = 1, j > i\} \cup \{h_i\}$$

By induction hypothesis or by considering the base case $i = k-1$ directly, we have

$$\Xi_\sigma^{>6}(h_i) = \{g_j, h_j \mid \mathfrak{b}_j = 1, j > i\} \cup \{h_i\}$$

and therefore $h_i \prec_\sigma s$, implying $\tau_\sigma(F_i) = h_i$.

– *Phase 5*: First, if $\mathfrak{b}_i^\oplus = 1$, it follows that $\sigma(d_i) = \sigma(e_i) = 1$ by definition of phase 5. Since we have a sink game, it follows that $\tau_\sigma(F_i) = h_i$. Furthermore it follows by definition of phase 5 that

$$\Xi_\sigma^{>6}(F_i) = \{g_j, h_j \mid \mathfrak{b}_j = 1, j > i\} \cup \{h_i\}$$

Second, if $\mathfrak{b}_i^\oplus = 0$ and $i \geq k$, it follows by definition of phase 5 that

$$\Xi_\sigma^{>6}(h_i) = \{g_j, h_j \mid \mathfrak{b}_j^\oplus = 1, j > i\} \cup \{h_i\}$$
$$\Xi_\sigma^{>6}(s) = \{g_j, h_j \mid \mathfrak{b}_j^\oplus = 1\} \cup \{s\}$$

It follows that $s \prec_\sigma h_i$ and hence $\tau_\sigma(F_i) = e_i$ and

$$\Xi_\sigma^{>6}(F_i) = \{g_j, h_j \mid \mathfrak{b}_j^\oplus = 1\} \cup \{s\}$$

Third, if $i < \mu(\mathfrak{b}) = k$, we show by backward induction on i that $\tau_\sigma(F_i) = h_i$, which implies that

$$\Xi_\sigma^{>6}(F_i) = \{g_j, h_j \mid \mathfrak{b}_j = 1, j > i\} \cup \{h_i\}$$

By induction hypothesis or by considering the base case $i = k-1$ directly, we have

$$\Xi_\sigma^{>6}(h_i) = \{g_j, h_j \mid \mathfrak{b}_j = 1, j > i\} \cup \{h_i\}$$

and therefore $h_i \prec_\sigma s$, implying $\tau_\sigma(F_i) = h_i$.
Fourth, if $i < k < \mu(\mathfrak{b})$, we show by backward induction on i that $\tau_\sigma(F_i) = h_i$, which implies that

$$\Xi_\sigma^{>6}(F_i) = \Xi_\sigma^{>6}(s) \cup \{g_{j+1}, h_j \mid i \leq j < k\}$$

By induction hypothesis or by considering the base case $i = k-1$ directly, we have

$$\Xi_\sigma^{>6}(h_i) = \Xi_\sigma^{>6}(s) \cup \{g_{j+1}, h_j \mid i < j < k\} \cup \{h_i, g_{i+1}\}$$

and therefore $h_i \prec_\sigma s$, implying $\tau_\sigma(F_i) = h_i$.

□

Figure 6 specifies the sets of improving switches for each phase p. It should be read as follows: an edge e is included in the set of improving switches $I(\sigma)$ iff $e \notin \sigma$ and the condition holds that is specified in the respective cell. If a cell contains a question mark, we do not specify whether the edge is included in the set.

The reason we do not specify some of the edges is that we do not need to know for our proof that the run of the algorithm takes exponentially many steps. In every step, the ordering of the edges as well as the currently selected edges narrows the set of improving edges that we actually have to consider.

Phase	1	2	3	4	5
$a_i^1 \in I(\sigma)$	?	?	?	?	$i = \mu(\mathfrak{b})$
$a_i^0 \in I(\sigma)$	?	?	?	?	$i < \mu(\mathfrak{b}) \wedge \sigma(a_{i+1}) = \mathfrak{b}_i^{\oplus}$
$b_i^1 \in I(\sigma)$	$i = \mu(\mathfrak{b}) \wedge \sigma(e_i) = 1$	?	?	?	?
$b_i^0 \in I(\sigma)$	$i > \mu(\mathfrak{b}) \wedge \sigma(e_{\mu(\mathfrak{b})}) = 1$	?	?	?	?
$c_i^1 \in I(\sigma)$	?	$i = \mu(\mathfrak{b})$	?	?	?
$c_i^0 \in I(\sigma)$	?	$i \neq \mu(\mathfrak{b})$	?	?	?
$d_i^1 \in I(\sigma)$	?	?	?	Yes	?
$d_i^0 \in I(\sigma)$	$\sigma(b_{\mu(\mathfrak{b})}) = 1 \wedge \sigma(b_{\nu(\mathfrak{b})}) = 0$	?	?	?	?
$e_i^1 \in I(\sigma)$	Yes	?	?	?	?
$e_i^0 \in I(\sigma)$	?	?	$\mathfrak{b}_i^{\oplus} = 0$	?	?

Fig. 6 Improving Switches

We finally arrive at the following main lemma describing the improving switches. It is based on the previous lemma as all valuations in our game are fundamentally based on the valuations of F_i and the counter-strategy τ_σ.

Lemma 5 *Let $n \geq 3$. The improving switches from strategies that belong to the phases in Figure 5 are as specified in Figure 6.*

Proof Let $n \geq 3$ and σ be a strategy belonging to one of the phases w.r.t. $\mathfrak{b}$.

- *Phase 1*: It follows from Lemma 4 that

$$\Xi_\sigma^{>6}(g_i) = \begin{cases} \{g_j, h_j \mid \mathfrak{b}_j = 1, j > i\} \cup \{g_i, h_i\} & \text{if } \sigma(d_i) = \sigma(e_i) = 1 \\ \{g_j, h_j \mid \mathfrak{b}_j = 1\} \cup \{g_i, s\} & \text{if } \sigma(e_i) = 0 \\ \{g_j, h_j \mid \mathfrak{b}_j^{\oplus} = 1\} \cup \{g_i\} & \text{if } \sigma(d_i) = 0 \wedge \sigma(e_i) = 1 \end{cases}$$

  Since $\tau_\sigma(F_i) = e_i$ if $\sigma(e_i) = 0$, it immediately follows that e_i^1 is an improving switch. It can furthermore be easily observed that d_i^0 is an improving switch for $\mathfrak{b}_i = 0$, $i > \nu(\mathfrak{b})$ iff $\sigma(e_{\nu(\mathfrak{b})}) = 1$ and $\sigma(b_{\nu(\mathfrak{b})}) = 1$.
- *Phase 2*: It follows from Lemma 4 that

$$\Xi_\sigma^{>6}(g_i) = \begin{cases} \{g_j, h_j \mid \mathfrak{b}_j = 1, j > i\} \cup \{g_i, h_i\} & \text{if } \mathfrak{b}_i^{\mu} = 1 \\ \{g_j, h_j \mid \mathfrak{b}_j^{\oplus} = 1\} \cup \{g_i\} & \text{if } \mathfrak{b}_i^{\mu} = 0 \end{cases}$$

  If $\mathfrak{b}_1 = 0$ and $\mathfrak{b}_2 = 1$, it follows that

$$\Xi_\sigma^{>6}(c_i) = \{g_j, h_j \mid \mathfrak{b}_j = 1\}$$

It immediately follows that c_2^0 is the only improving switch w.r.t. c. Otherwise, if $\mathfrak{b}_1 = 0$ and $\mathfrak{b}_2 = 0$, it follows that

$$\Xi_\sigma^{>6}(c_i) = \begin{cases} \{g_j, h_j \mid \mathfrak{b}_j = 1\} & \text{if } i \geq \nu(\mathfrak{b}) \\ \{g_j, h_j \mid \mathfrak{b}_j^\oplus\} & \text{if } i < \nu(\mathfrak{b}) \end{cases}$$

It immediately follows that $c^0_{\nu(\mathfrak{b})}$ is the only improving switch w.r.t. c. Otherwise, if $\mathfrak{b}_1 = 1$, it follows that

$$\Xi_\sigma^{>6}(c_i) = \{g_j, h_j \mid \mathfrak{b}_j = 1\}$$

Hence, it follows that $c^1_{\mu(bit)}$ is the only improving switch w.r.t. c.

– *Phase 3*: Let i s.t. $\sigma(e_i) = 1$. It follows from Lemma 4 that

$$\Xi_\sigma^{>6}(F_i) = \begin{cases} \{g_j, h_j \mid \mathfrak{b}_j = 1, j > i\} \cup \{h_i\} & \text{if } \sigma(d_i) = 1 \\ \{g_j, h_j \mid \mathfrak{b}_j^\oplus = 1\} & \text{if } \sigma(d_i) = 0 \end{cases}$$
$$\Xi_\sigma^{>6}(s) = \{g_j, h_j \mid \mathfrak{b}_j^\oplus = 1\} \cup \{s\}$$

Hence, we have $F_i \prec_\sigma s$.

– *Phase 4*: Let i s.t. $\sigma(d_i) = 0$. It follows from Lemma 4 that $\tau_\sigma(F_i) = d_i$ and hence $\Xi_\sigma(F_i) = \{F_i\} \cup \Xi_\sigma(d_i)$, i.e. $\sigma(d_i) \prec_\sigma F_i$.

– *Phase 5*: Let $k = \max(\{i \mid \sigma(a_i) \neq \mathfrak{b}_i^\oplus\} \cup \{1\})$. It follows from Lemma 4 that

$$\Xi_\sigma^{>6}(g_i) = \begin{cases} \{g_j, h_j \mid \mathfrak{b}_j = 1, j > i\} \cup \{g_i, h_i\} & \text{if } \mathfrak{b}_i^\oplus = 1 \vee (i < k = \mu(\mathfrak{b})) \\ \{g_j, h_j \mid \mathfrak{b}_j^\oplus = 1\} \cup \{g_i, s\} & \text{if } \mathfrak{b}_i^\oplus = 0 \wedge i \geq k \\ \{g_j, h_j \mid \mathfrak{b}_j^\oplus = 1 \vee i < j < k\} \cup \{s, g_k, g_i, h_i\} & \text{if } \mathfrak{b}_i^\oplus = 0 \wedge i < k < \mu(\mathfrak{b}) \end{cases}$$

Hence, it follows that if $k > 1$, the only improving switch is either a_k^1 or a_k^0 w.r.t. a.

□

We are now ready to formulate our main lemma describing the transitioning from an initial phase 1 strategy corresponding to $\mathfrak{b}$ to a successor initial phase 1 strategy corresponding to $\mathfrak{b}^\oplus$, complying with the given ordering selection.

Lemma 6 *Let σ be a phase 1 strategy with configuration $\mathbf{0}_n < \mathfrak{b} < \mathbf{1}_n$. Let z be an edge with $z \preceq e_1^1$ or $a_n^1 \preceq e$. Then, there is a phase 1 strategy σ' with configuration $\mathfrak{b}^\oplus$ and an edge z' with $z' \preceq e_1^1$ or $a_n^1 \preceq z'$ s.t. $(\sigma, z) \rightsquigarrow_\prec (\sigma', z')$.*

Proof The proof of the lemma is ultimately based on the five phases described in Figure 5, the corresponding improving switches given in Figure 6 (proven correct in Lemma 5) and the introduced selection ordering.

We prove the lemma by outlining the complete sequence of switches that are applied to σ in order to obtain σ' (we do not explicitly describe the intermediate strategies which can be derived by applying all mentioned switches up to that point).

Let $i_1, \ldots, i_k$ be the complete sequence of ascending indices s.t. $\mathfrak{b}_{i_j} = 0$ for $1 \leq j \leq k$. The following holds:

$$\overset{\text{P1}}{\rightsquigarrow} e^1_{i_1} \rightsquigarrow b^1_{i_1} \rightsquigarrow d^0_{i_2} \rightsquigarrow e^1_{i_2} \rightsquigarrow d^0_{i_3} \rightsquigarrow e^1_{i_3} \rightsquigarrow \ldots \rightsquigarrow d^0_{i_k} \rightsquigarrow e^m_{i_k} 1$$
$$\overset{\text{P2}}{\rightsquigarrow} c^1_{i_1}$$
$$\overset{\text{P3}}{\rightsquigarrow} \{e^0_i \mid \mathfrak{b}^{\oplus} = 0\}$$
$$\overset{\text{P4}}{\rightsquigarrow} \{d^1_i \mid \mathfrak{b}^{\nu} = 0\}$$
$$\overset{\text{P5}}{\rightsquigarrow} a^*_{i_1} \rightsquigarrow a^*_{i_1-1} \rightsquigarrow \ldots \rightsquigarrow a^*_2$$

□

It follows immediately that the parity games described here indeed simulate a binary counter by starting with the designated initial strategy and the $\prec$-minimal edge. This completes the proof of Theorem 4.

2.5 Application to Other Games

Theorem 4 can be applied to a variety of other games using known connections between these games. For completeness we give these results here. However we will not give definitions of the games concerned, referring instead to the relevant literature, as they will not be needed in the rest of the paper.

Parity games can be reduced to mean payoff games [19], mean payoff games to discounted payoff games, and the latter ones to turn-based stochastic games [25]. Friedmann showed [8] that the strategy improvement algorithm for payoff and stochastic games, when applied to the reduced game graphs of sink parity games, behaves exactly the same on the reduced games. In other words, the strategies in both the original game and the reduced game graphs coincide as well as the associated sets of improving switches.

Theorem 5 ([8]) *Let G be a sink parity game. Discrete strategy improvement requires the same number of iterations to solve G as strategy improvement for induced payoff games, as well as turn-based stochastic games, to solve the respective game G'. The game G' is induced by applying the standard reduction from G to the respective game class, assuming that the improvement rule solely depends on the combinatorial valuation-ordering of the improving edges.*

By this, we can conclude that the exponential lower bound for Cunningham's rule on parity games presented here also applies to payoff and turn-based stochastic games. We will see in the next section that it also applies to Markov decision processes, which can be seen as a one-player version of turn-based stochastic games.

Corollary 1 *There is a family of mean payoff games, discounted payoff games, respectively, turn-based stochastic games, on which the number of improving steps performed by Algorithm 2 is at least 2^n, where the size of the games are $O(n)$.*

3 Markov Decision Process Policy Iteration Lower Bound

Markov decision processes (MDPs) provide a mathematical model for sequential decision making under uncertainty. They are employed to model stochastic optimization problems in various areas ranging from operations research, machine learning, artificial intelligence, economics and game theory. For an in-depth coverage of MDPs, see the books of Howard [14], Derman [6], Puterman [20] and Bertsekas [3].

3.1 Markov Decision Processes and Policy Iteration

Formally, an MDP is defined by its *underlying graph* $G{=}(V_0, V_R, E_0, E_R, r, p)$. Here, V_0 is the set of vertices (states) operated by the controller, also known as *player* 0, and V_R is a set of *randomization* vertices corresponding to the probabilistic actions of the MDP. We let $V = V_0 \cup V_R$. The edge set $E_0 \subseteq V_0 \times V_R$ corresponds to the actions available to the controller. The edge set $E_R \subseteq V_R \times V_0$ corresponds to the probabilistic transitions associated with each action. The function $r : E_0 \to \mathbb{R}$ is the immediate reward function. The function $p : E_R \to [0, 1]$ specifies the transition probabilities. For every $u \in V_R$, we have $\sum_{v:(u,v)\in E_R} p(u, v) = 1$, i.e., the probabilities of all edges emanating from each vertex of V_R sum up to 1.

All MDPs considered in this paper satisfy the *unichain* condition (see [20]). It states that the Markov chain obtained from each policy σ has a single irreducible recurrent class (i.e. all states in the class have a finite hitting time from all states in the class, and the class is complete in that sense).

A policy σ is a function $\sigma : V_0 \to V$ that selects for each vertex $u \in V_0$ a target node v corresponding to an edge $(u, v) \in E_0$, i.e. $(u, \sigma(u)) \in E_0$. We assume that each vertex $u \in V_0$ has at least one outgoing edge. There are several *objectives* for MDPs and we consider the *expected total reward objective* here. The *values* $\mathtt{val}_\sigma(u)$ of the vertices under σ are defined as the unique solutions – if existing – of the following set of linear equations:

$$\mathtt{val}_\sigma(u) = \begin{cases} \mathtt{val}_\sigma(v) + r(u, v) & \text{if } u \in V_0 \text{ and } \sigma(u) = v \\ \sum_{v:(u,v)\in E_R} p(u, v)\, \mathtt{val}_\sigma(v) & \text{if } u \in V_R \end{cases}$$

together with the condition that $\mathtt{val}_\sigma(u)$ sum up to 0 on each irreducible recurrent class of the Markov chain defined by σ (making the solution unique).

As is the case for parity games, the policy iteration algorithm starts with some initial policy σ_0 and generates an improving sequence $\sigma_0, \sigma_1, \ldots, \sigma_N$ of policies, ending with an optimal policy σ_N. In each iteration the algorithm first *evaluates* the current policy σ_i, by computing the values $\mathtt{val}_{\sigma_i}(u)$ of all vertices. An edge $(u, v') \in E_0$, such that $\sigma_i(u) \neq v'$ is then said to be an *improving switch* if and only if either $\mathtt{val}_{\sigma_i}(v') > \mathtt{val}_{\sigma_i}(u)$. Given a policy σ, we again denote the *set of improving switches* w.r.t. v by $I_\sigma(v)$.

A crucial property of policy iteration is that σ is an optimal policy if and only if there are no improving switches with respect to it (see, e.g., [14], [20]).

Furthermore, if $v' \in I_\sigma(u)$ is an improving switch w.r.t. σ, and σ' is defined as $\sigma[(u,v')]$ (i.e., $\sigma'(u) = v'$ and $\sigma'(w) = \sigma(w)$ for all $w \neq u$), then σ' is *strictly* better than σ, in the sense that for every $u \in V_0$, we have $\mathtt{val}_{\sigma'}(u) \geq \mathtt{val}_\sigma(u)$, with a strict inequality for at least one vertex $u \in V_0$.

Lemma 7 ([20]) *Let σ be a policy and (v,w) be a σ-improving switch. Let $\sigma' = \sigma[(v,w)]$. Then the following holds:*

1. $\mathtt{val}_{\sigma'}(u) \geq \mathtt{val}_\sigma(u)$ *for all* $u \in V$,
2. $\mathtt{val}_{\sigma'}(v) > \mathtt{val}_\sigma(v)$, *and particularly*
3. $\sum_{u \in V_0} \mathtt{val}_{\sigma'}(u) > \sum_{u \in V_0} \mathtt{val}_\sigma(u)$.

3.2 Lower Bound Construction

We relate the description of the lower bound construction for Markov decision processes closely to the construction of the parity games. For that reason, we relax our definition of MDPs such that it corresponds almost directly to parity games.

As defined in the previous subsection, the underlying graph G is bipartite. However one can relax this condition and allow edges from V_0 to V_0 and from V_R to V_R. It suffices to subdivide these edges by inserting a node belonging to player 1 or player 0, respectively, with out-degree 1 and no reward. This leads to the following definition. We additionally allow rewards on edges played by nature. More formally:

A *relaxed MDP* is a tuple $M = (V, V_0, V_R, E, E_0, E_R, r, p)$, where $V = V_0 \cup V_R$, $E \subseteq V \times V$, $E_0 = E \cap (V_0 \times V)$, $E_R = E \cap (V_R \times V)$, $r : E \mapsto \mathbb{R}$ and $p : E_R \to [0,1]$ with $\sum_{w \in vE} p(v,w) = 1$ for all $v \in V_R$.

Given a relaxed MDP, we can easily reduce it to an equivalent MDP in the following fashion:

1. Edges $(v,w) \in E_0 \cap (V_0 \times V_0)$ can be realized by adding a randomization node (v,w) and by replacing the edge (v,w) with new edges $(v,(v,w))$ and $((v,w),w)$. The outgoing edge from (v,w) obviously has probability 1.
2. Edges $(v,w) \in E_R \cap (V_R \times V_R)$ can be realized by adding a player 0 node (v,w) and by replacing the edge (v,w) with new edges $(v,(v,w))$ and $((v,w),w)$. Note that player 0 does not obtain new choices by adding this node since the out-degree is one.
3. Randomization edges are not allowed to have rewards in MDPs. Hence we insert, for every outgoing edge of a randomization node with reward, a new node of player 0 connected to a new node of the randomizer connected to the original target node. We push the reward to the new player 0 edge.
4. Since we consider the expected total reward here, adding new intermediate nodes does not change the value of the nodes.

Relaxed MDPs allow us to show the close relationship between the original parity games and the corresponding MDPs.

For each $n \geq 3$ we define the underlying graph $M_n = (V, V_0, V_R, E, E_0, E_R, r, p)$ of a relaxed MDP as shown schematically in Figure 8 which the reader is invited to compare with Figure 1.

More formally:

$$V_0 := \{a_i, c_i, d_i \mid 1 < i \leq n\} \cup \{b_i \mid 1 < i < n\} \cup \{e_i \mid 1 \leq i \leq n\}$$
$$V_R := \{F_i \mid 1 \leq i \leq n\} \cup \{g_i, h_i \mid 1 \leq i \leq n\} \cup \{s, t\}$$

With M_n, we associate a large number $N \in \mathbb{N}$ and a small number $\varepsilon > 0$.

Some of the vertices are assigned integer *priorities*. If a vertex v has priority $\Omega(v)$ assigned to it, then a reward of $\langle v \rangle = (-N)^{\Omega(v)}$ is *added* to all edges emanating from v. This idea of using priorities is inspired by the reduction from parity games to mean payoff games, see [19].

We require N to be at least as large as the number of nodes with priorities, i.e. $N \geq 2n$ and ε^{-1} to be significantly larger than the largest occurring priority induced reward, i.e. $\varepsilon \leq N^{-2n}$.

Figure 7 defines the edge sets, the probabilities and the priorities of M_n. For convenience of notation, we identify the node names a_{n+1} with t, b_1 with g_1, and c_1 with g_1. Explicit constructions for small n are available online [11].

Node	Successors	Probability
F_i	h_i	ε
F_i	$d_{i>1}$	$0.5 \cdot (1 - \varepsilon)$
F_i	e_i	$0.5 \cdot (1 - \varepsilon)$

Node	Successors	Priority
g_i	F_i	$2 \cdot i - 1$
h_i	a_{i+1}	$2 \cdot i$
s	c_n	0

Node	Successors
a_i	g_i, a_{i+1}
b_i	g_i, b_{i-1}
c_i	g_i, c_{i-1}
d_i	F_i, b_{i-1}
e_i	F_i, s
t	t

Fig. 7 MDP Lower Bound Graph

In comparing Figure 8 with Figure 1 the connections between MDPs and parity games becomes clear. Blue edges show respectively the current policy for player 0, and red edges for player 1. Improving edges for player 0 are shown in dotted green and the other current non-policy and non-strategy edges are shown in dotted black. (Coloured edges show in bold on monochromatic printing.) Whereas Figure 1 shows the initial strategy for parity game G_3, Figure 8 shows the initial policy for the MDP M_3. The sequence P_n of improving switches we constructed in G_n starting at the initial strategy and following Algorithm 2 using the ordering (1) will be shown to correspond to an identical sequence of improving switches in M_n using policy iteration and the same ordering.

Lemma 8 *The Markov chains obtained by any policy reach the sink t almost surely (i.e. the sink t is the single irreducible recurrent class).*

It is not too hard to see that the absolute value of all nodes corresponding to policies are bounded by ε^{-1}. More formally we have:

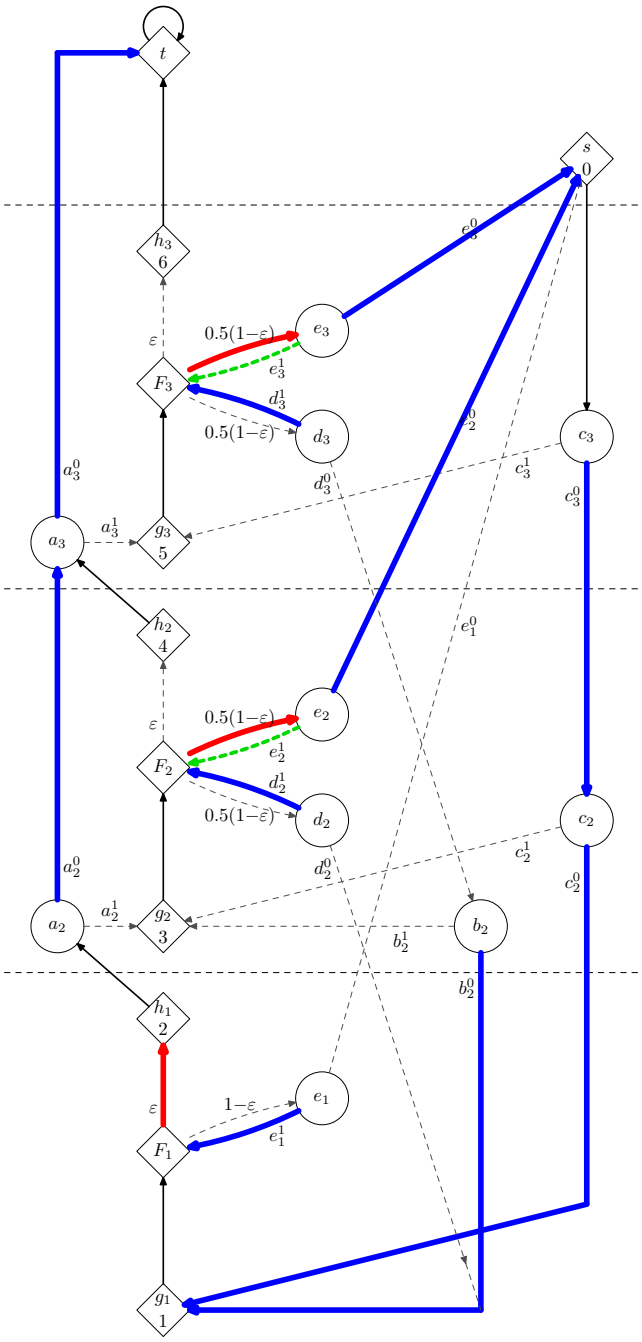

Fig. 8 (Relaxed) Markov Decision Process Lower Bound Graph

Lemma 9 *Let $P = \{s, g_*, h_*\}$ be the set of nodes with priorities. For a subset $S \subseteq P$, let $\sum(S) = \sum_{v \in S} \langle v \rangle$. For non-empty subsets $S \subseteq P$, let $v_S \in S$ be the node with the largest priority in S.*

1. *$|\sum(S)| < N^{2n}$ and $\varepsilon \cdot |\sum(S)| < 1$ for every subset $S \subseteq P$, and*
2. *$|v_S| < |v_{S'}|$ implies $|\sum(S)| < |\sum(S')|$ for non-empty subsets $S, S' \subseteq P$.*

3.3 Lower Bound Proof

In this section we prove the following theorem. The *initial policy*, $\{a_*^0, b_*^0, c_*^0, d_*^1, e_*^0\}$, corresponds exactly to the initial strategy for the parity game G_n.

Theorem 6 *The sequence P_n of improving switches followed by policy iteration in M_n from the initial policy to the terminal policy using the ordering (1) has length at least 2^n, where M_n has size $O(n)$.*

We will show that the sequence P_n in Theorem 6 is in one-to-one correspondence with the sequence P_n in Theorem 4. We first show that Figure 6 indeed specifies the sets of improving switches for each phase p.

Lemma 10 *Let $n > 1$. The improving switches from policies that belong to the phases in Figure 5 are as specified in Figure 6.*

Proof Let $n > 1$ and σ be a policy belonging to one of the phases w.r.t. $\mathfrak{b}$. Let $\mu = \mu(\mathfrak{b})$. Define $T_i = \langle h_i \rangle + \sum_{j>i, \mathfrak{b}_j = 1}(\langle g_j \rangle + \langle h_j \rangle)$ and $S_i = \langle g_i \rangle + T_i$.

– *Phase 1*: The following holds:

$$\mathtt{val}_\sigma(g_i) = \begin{cases} S_i & \text{if } \sigma(d_i) = 1, \sigma(e_i) = 1 \\ \varepsilon \cdot T_i + \langle g_i \rangle + (1-\varepsilon) \cdot (\langle s \rangle + S_\mu) & \text{if } \sigma(d_i) = 1, \sigma(e_i) = 0 \\ \varepsilon \cdot T_i + \langle g_i \rangle + (1-\varepsilon) \cdot S_\mu & \text{if } \sigma(d_i) = 0, \sigma(e_i) = 1 \\ \varepsilon \cdot T_i + \langle g_i \rangle + \frac{1-\varepsilon}{2} \cdot (\langle s \rangle + 2 \cdot S_\mu) & \text{if } \sigma(d_i) = 0, \sigma(e_i) = 0 \end{cases}$$

$$\mathtt{val}_\sigma(e_i) = \begin{cases} \langle s \rangle + S_\mu & \text{if } \sigma(e_i) = 0 \end{cases}$$

It is easy to see that e_i^1 are improving switches as $T_i > S_\mu$. It can furthermore be easily observed that d_i^0 is an improving switch for $\mathfrak{b}_i = 0$, $i > \nu(\mathfrak{b})$ iff $\sigma(e_{\nu(\mathfrak{b})}) = 1$ and $\sigma(b_{\nu(\mathfrak{b})}) = 1$.

– *Phase 2*: Similar to phase 5.
– *Phase 3*: Similar to phase 4.
– *Phase 4*: The following holds:

$$\mathtt{val}_\sigma(b_i) = \begin{cases} S_\mu & \text{if } i \geq \mu \\ S_1 & \text{otherwise} \end{cases}$$

$$\mathtt{val}_\sigma(F_i) = \begin{cases} T_i & \text{if } \mathfrak{b}_i^\oplus = 1 \\ \varepsilon \cdot \mathcal{O}(1) + \frac{1}{2} \cdot (\langle s \rangle + S_\mu + \mathtt{val}_\sigma(b_{i-1})) & \text{if } \mathfrak{b}_i^\mu = 0 \wedge \sigma(d_i) \neq F_i \\ \varepsilon \cdot \mathcal{O}(1) + \langle s \rangle + S_\mu & \text{otherwise} \end{cases}$$

We conclude: $\mathtt{val}_\sigma(F_i) - \mathtt{val}_\sigma(b_{i-1}) =$

$$\begin{cases} T_i - \mathtt{val}_\sigma(b_{i-1}) \geq T_i - S_\mu > 0 & \text{if } S_\mu \\ \varepsilon \cdot \mathcal{O}(1) + \frac{1}{2} \cdot (\langle s \rangle + S_\mu - \mathtt{val}_\sigma(b_{i-1})) \geq \varepsilon \cdot \mathcal{O}(1) + \frac{1}{2} \cdot \langle s \rangle > 0 & \text{if } \mathfrak{b}_i^\mu = 0 \wedge \sigma(d_i) \neq F_i \\ \varepsilon \cdot \mathcal{O}(1) + \langle s \rangle + S_\mu - \mathtt{val}_\sigma(b_{i-1}) \geq \varepsilon \cdot \mathcal{O}(1) + \langle s \rangle > 0 & \text{otherwise} \end{cases}$$

– *Phase 5*: The following holds:

$$\mathtt{val}_\sigma(g_i) = \begin{cases} S_i & \text{if } \mathfrak{b}_i^\oplus = 1 \\ \varepsilon \cdot \mathcal{O}(1) + \langle g_i \rangle + \langle s \rangle + S_\mu & \text{otherwise} \end{cases}$$

$$\mathtt{val}_\sigma(a_i) = \begin{cases} S_i & \text{if } \mathfrak{b}_i^\oplus = 1 \wedge (i > \mu \vee \sigma(a_\mu) = g_\mu) \\ \mathtt{val}_\sigma(a_{i+1}) & \text{if } i = \mu \wedge \sigma(a_\mu) \neq g_\mu) \\ S_\mu & \text{if } i < \mu \wedge \sigma(a_i) \neq g_i \\ \varepsilon \cdot \mathcal{O}(1) + \langle g_i \rangle + \langle s \rangle + S_\mu & \text{if } i < \mu \wedge \sigma(a_i) = g_i \end{cases}$$

By computing the difference we again see that the improving switches are as described.

Now Lemma 6 becomes applicable again and the proof of Theorem 6 follows.

4 Lower bound for the Simplex Method using Cunningham's rule

In this section we use the well known transformation from MDPs to linear programs to obtain an exponential lower bound for the simplex method using Cunningham's rule.

4.1 Linear Programs and the Simplex Method

We briefly give a few basic definitions and state our notation. For more information the reader is referred to any standard linear programming text, such as [4]. For the purpose of this paper, we assume LPs to be bounded.

Given an m by n matrix A with $m \leq n$, an n-vector c and m-vector b we consider the primal linear program in the standard form

$$\begin{aligned} \max\ & c^T x \\ \text{s.t.}\ & Ax = b \\ & x \geq 0 \end{aligned}$$

Let B and N be a partition of the indices $\{1, 2, \ldots, n\}$ such that $|B| = m$. We denote by A_B the submatrix of A with columns indexed by B, and x_B the m-subvector of x with indices in B. We say that B is a *feasible basis* if A_B is non-singular and

$$x_B = A_B^{-1} b \geq 0.$$

A corresponding *basic feasible solution(BFS)* x is obtained if we extend x_B to x by setting $x_N = 0$.

We call $c^T x$ the *objective function*. A *pivot* from a feasible basis B is defined by a pair of indices $i \in B$ and $j \in N$ for which $B \setminus \{i\} \cup \{j\}$ is also a feasible basis. If the corresponding BFS are x and x', then the pivot is *improving* if $c^T x \leq c^T x'$. If the inequality is strict then we call the pivot step *non-degenerate* otherwise it is called *degenerate*. An *optimal basis* is one for which the corresponding BFS maximizes the objective function. A *deterministic pivot rule* gives a unique pivot pair for every non-optimal feasible basis.

The simplex method starts from a given feasible basis and applies improving pivots until an optimal basis is obtained. The sequence of pivots depends on the specific pivot rule, and care must be taken to ensure that it does not cycle if there are degenerate pivots.

The corresponding dual LP is written

$$\begin{array}{l} \min\ b^T y \\ \text{s.t.}\ \ A^T y \geq c. \end{array}$$

4.2 Markov Decision Processes as LPs

Optimal policies for MDPs that satisfy the unichain condition can be found by solving the following primal linear program (see, e.g., [20].)

$$(P) \quad \begin{array}{ll} \max & \sum_{(u,v)\in E_0} r(u,v)x(u,v) \\ \text{s.t.} & \sum_{(u,v)\in E} x(u,v) - \sum_{(v,w)\in E_0,(w,u)\in E_R} p(w,u)x(v,w) = 1,\ u \in V_0 \\ & x(u,v) \ \geq\ 0 \quad , \quad (u,v) \in E_0 \end{array}$$

The variable $x(u,v)$, for $(u,v) \in E_0$, stands for the probability (frequency) of using the edge (action) (u,v). The constraints of the linear program are *conservation constraints* that state that the probability of entering a vertex u is equal to the probability of exiting u. It is not difficult to check that the BFS's of (P) correspond directly to policies of the MDP. For each policy σ we can define a feasible setting of primal variables $x(u,v)$, for $(u,v) \in E_0$, such that $x(u,v) > 0$ only if $\sigma(u) = (u,v)$. Conversely, for every BFS $(x(u,v))_{u,v}$ we can define a corresponding policy σ. It is well known that the policy corresponding to an optimal BFS of (P) is an optimal policy of the MDP. (See, e.g., [20].)

Lemma 11 *Let σ be a policy and $(x(u,v))_{u,v}$ be a corresponding BFS. Then the following holds:*

$$\sum_{u\in E_0} \mathtt{val}_\sigma(u) = \sum_{(u,v)\in E_0} r(u,v)x(u,v)$$

The dual linear program (for unichain MDPs) is:

$$(D) \quad \begin{array}{ll} \min & \sum_{u\in V} y(u) \\ \text{s.t.} & y(u) - \sum_{(v,w)\in E_R} p(v,w)y(w) \ \geq\ r(u,v) \quad , \quad (u,v) \in E_0 \end{array}$$

together with the condition that $y(u)$ sum up to 0 on the single irreducible recurrent class.

If y^* is an optimal solution of (D), then $y^*(u)$, for every $u \in V_0$, is the value of u under an optimal policy. An optimal policy σ^* can be obtained by letting $\sigma^*(u) = (u, v)$, where $(u, v) \in E_0$ is an edge for which the inequality constraint in (D) is tight, i.e., $y(u) - \sum_{w:(v,w)\in E_R} p(v, w)y(w) = r(u, v)$. Such a tight edge is guaranteed to exist.

4.3 Policy Iteration and Simplex Method

A policy iteration algorithm with policy σ that perform a single switch at each iteration – like Cunningham's rule – corresponds to a variation of the simplex method where the selection rule behaves like Cunningham's rule for LPs. Indeed every policy σ gives rise to a feasible solution $x(u, v)$ of the primal linear program (P) . We use σ to define a Markov chain and let $x(u, v)$ be the 'steady-state' probability that the edge (action) (u, v) is used. In particular, if $\sigma(u) \neq v$, then $x(u, v) = 0$.

We can also view the values corresponding to σ as settings of the variables $y(u)$ of the dual linear program (D). By linear programming duality, if $y(u)$ is feasible then σ is an optimal policy. It is easy to check that an edge $(u, v') \in E_0$ is an improving switch if and only if the dual constraint corresponding to (u, v') is violated. Furthermore, replacing the edge (u, v) by the edge (u, v') corresponds to a pivoting step, with a non-negative reduced cost, in which the column corresponding to (u, v') enters the basis, while the column corresponding to (u, v) leaves the basis.

4.4 Lower Bound Construction

Let $n > 1$. The variables of the LP correspond to the edges E_0 controlled by player 0, i.e. we have $10(n - 1)$ variables

$$\{a_i^1, a_i^0, c_i^1, c_i^0, d_i^1, d_i^0 \mid 1 < i \leq n\} \cup \{b_i^1, b_i^0 \mid 1 < i < n\} \cup \{e_i^1, e_i^0 \mid 1 \leq i \leq n\}.$$

The LP has $5(n - 1)$ constraints, corresponding to the nodes V_0 controlled by player 0, and labelled

$$\{a_i, c_i, d_i \mid 1 < i \leq n\} \cup \{b_i \mid 1 < i < n\} \cup \{e_i \mid 1 \leq i \leq n\}.$$

The linear program is defined as follows for each $n \geq 3$ (non-existent variables are assumed to be zero):

LP_n :
$\max \sum_{i=1}^{n} \left(\left(a_i^1 + b_i^1 + c_i^1\right)(\Omega(g_i) + \varepsilon\Omega(h_i)) + \varepsilon\left(d_i^1 + e_i^1\right)\Omega(h_i) + e_i^0\Omega(s)\right)$

subject to:

$$
\begin{array}{lll}
(a_2) & a_2^0 + a_2^1 = 1 + \varepsilon(b_2^0 + c_2^0 + d_2^0 + e_1^1) & \\
(a_i) & a_i^0 + a_i^1 = 1 + a_{i-1}^0 + \varepsilon(a_{i-1}^1 + b_{i-1}^1 + c_{i-1}^1 + d_{i-1}^1 + e_{i-1}^1) & 3 \le i \le n \\
(b_i) & b_i^0 + b_i^1 = 1 + b_{i+1}^0 + d_{i+1}^0 & 2 \le i < n \\
(c_i) & c_i^0 + c_i^1 = 1 + c_{i+1}^0 & 2 \le i < n \\
(c_n) & c_n^0 + c_n^1 = 1 + \sum_{j=1}^{n} e_j^0 & \\
(d_i) & d_i^0 + d_i^1 = 1 + \frac{1-\varepsilon}{2}(a_i^1 + b_i^1 + c_i^1 + d_i^1 + e_i^1) & 2 \le i \le n \\
(e_1) & e_1^0 + e_1^1 = 1 + (1-\varepsilon)(b_2^0 + c_2^0 + d_2^0 + e_1^1) & \\
(e_i) & e_i^0 + e_i^1 = 1 + \frac{1-\varepsilon}{2}(a_i^1 + b_i^1 + c_i^1 + d_i^1 + e_i^1) & 2 \le i \le n \\
 & \text{All variables non-negative} &
\end{array}
$$

Note that the size of LP_n is linear in n. Depending on the context we let LP_n denote both the linear program and the polytope defined by its constraints. For small values of n explicit constructions of LP_n and its dual are available online [11].

We now use the correspondence between the parity game G_n, the MDP M_n and linear program LP_n to get a lower bound for the simplex method using Cunningham's rule. The initial strategy for G_n and initial policy for M_n, $\{a_*^0, b_*^0, c_*^0, d_*^1, e_1^1, e_{*>1}^0\}$, defines a *starting basis* for LP_n. We construct a path on the polytope LP_n from this starting basis using the least recently considered rule with ordering (1). Using this construction, Theorem 6 implies that the path generated will be in one-to-one correspondence with the sequence P_n of improving switches generated in M_n (and hence G_n). We observe that the objective function strictly increases with each pivot due to Lemma 11 and Lemma 7. Therefore we have the following result.

Theorem 7 *The pivot path P_n for LP_n from the starting basis to the optimum basis followed by the least recently considered rule with ordering (1) has length at least 2^n. The objective function strictly increases with each pivot.*

5 Acyclic Unique Sink Orientations

Our final result concerns *acyclic unique sink orientations* (AUSOs), which are abstractions of various optimization problems including linear programming, linear complementarity and binary payoff games. In this section we extend our exponential lower bound to finding the sink of an AUSO using the least considered rule. For background information on AUSOs, see [22,12].

5.1 Definitions and previous results

AUSOs can be defined on arbitrary polytopes, but here we consider only hypercubes. An AUSO on a n-dimensional hypercube is an orientation of its edges that is acyclic and such that every face of the hypercube has a unique sink (vertex of outdegree 0). The goal is to find the unique sink of the AUSO. There is at present no known polynomial time algorithm for doing this, nor is it known to require superpolynomial time in the worst case.

A natural class of algorithms to find the sink of an AUSO are *path following algorithms*. Such an algorithm would start at any vertex v of the hypercube and repeatedly choose an outgoing edge according to some rule until the unique sink is located. Each edge of the path corresponds to flipping one bit of the current vertex.

There is a very natural analogy between path following algorithms and pivoting in linear programming. Pivot rules for LPs that involve only the *signs* of the coefficients of the objective function have natural analogues for AUSOs and a full discussion of this is contained in [1]. In particular the least recently considered rule can be adapted to give a path following algorithm to find the unique sink of an n-cube AUSO starting at any given vertex as follows.

We define, for each $i = 1, 2, ..., n$, the variable v_i to denote a flip of bit i from 0 to 1, and variable v_{n+i} to denote a flip of bit i from 1 to 0. We may now arrange the $2n$ variables $v_1, v_2, ..., v_{2n}$ in any cyclic order. For any vertex of the AUSO that is not the sink we find the first allowable flip in this cyclic order starting at the last chosen v_i.

Suppose we are given a polytope and an objective function that is not constant on any edge of the polytope. We can can then orient each edge in the direction of increasing objective function. The corresponding directed graph on the skeleton of the polytope can be shown to be an AUSO. The converse is not always true. Indeed, we call an AUSO *realizable* if there is a polytope and objective function that induces a directed graph on its skeleton which is graph isomorphic to the AUSO. Not all AUSOs are realizable and some small examples are given in [2].

The Klee-Minty examples[16] are realizations of AUSOs on hypercubes, so exponential lower bounds for most of the non-history based deterministic LP pivot rules immediately give similar bounds for AUSOs. Gärtner[12] gives an $exp(2n^{1/2})$ lower bound for random facet algorithms and Matoušek and Szabó[18] give an $exp(\Omega(n^{1/3}))$ lower bound for the random edge rule on AUSOs. In the next subsection we derive an exponential lower bound for finding the sink of a realizable AUSO using a path following algorithm based on Cunningham's rule.

5.2 Lower Bound Construction

As we saw in Section 2 there is a direct relationship between binary parity games and oriented hypercubes. Each vertex v corresponds to a strategy σ for player 0. A partial orientation of the hypercube's edges is given by the notion of improving switches: the orientation goes from σ to σ' iff there is a game edge e belonging to player 0 s.t. $\sigma' = \sigma[e]$ and e is σ-improving. This is only a partial orientation since there are strategies for the parity game that are not used in the lower bound construction and for which the notion of improving is not well defined. For each $n \geq 3$ we denote this partially oriented n-cube H_n.

Our goal is to embed H_n into an AUSO A_n whose edges orientations are consistent with those already set in H_n. Note that this is not a trivial oper-

ation, as even for $n = 3$ it can be readily be verified that there are partial acyclic orientations of the 3-cube which are USOs on every complete face but do not embed into an AUSO. We will achieve this embedding via the linear programming formulation of the last section, achieving the stronger result that the AUSO is realizable.

Lemma 12 *LP_n is a realization of an AUSO A_n which is consistent with the edge orientations of H_n.*

Proof We argued in Section 4.2 that each basic feasible solution of LP_n corresponded directly to a policy of the corresponding MDP M_n, and hence to strategy of the parity game G_n. Also each edge of LP_n corresponded to a switch in M_n and hence to an edge owned by player 0 in G_n. It follows that the vertices and edges of LP_n and H_n are in one-to-one correspondence. Since H_n is an n-cube so is the skeleton of LP_n. By applying symbolic dual perturbation if necessary to resolve ties in the objective function (see, e.g. [4]) A_n can be oriented to give an AUSO, which we denote as A_n. By definition A_n is realizable.

According to Theorem 7 the objective function of LP_n is non-constant on every edge and increases in the direction corresponding to an improving switch in M_n. However improving switches in M_n correspond to improving edges in G_n. The edges of H_n were directed in the same way as these improving edges. Since dual perturbation will not change the direction of any edges for which the objective function is strictly increasing, the directed edges in H_n maintain their directions in A_n. The lemma follows. □

The lemma implies that not only is H_n an AUSO but stronger properties, such as the Holt-Klee condition and the shelling property, also hold (see, e.g. [2]). For small values of n the AUSOs A_n are available online [11].

We may now apply the least recently considered rule to A_n as described in Section 5.1. The starting basis of LP_n $\{a^0_*, b^0_*, c^0_*, d^1_*, e^0_*\}$ defines a *starting vertex* of A_n. We construct a path from this vertex using the least recently considered rule with ordering (1). Using this construction, Theorem 7 implies that the path generated will be in one-to-one correspondence with the path P_n generated in LP_n.

Theorem 8 *The directed path P_n in A_n from the starting vertex to the unique sink followed by the least recently considered rule with ordering (1) has length at least 2^n.*

6 Conclusions

We have shown in this paper that Cunningham's least considered rule can lead to exponential worst case behaviour in parity and other games, Markov decision processes, linear programs and AUSOs. This appears to be the first such result for a history based rule for AUSOs. However Cunningham's rule

was in fact first proposed for the network simplex method. The LPs presented in Section 4 are not networks, but are structurally remarkably close to them. Our first open problem would be to extend our results to network LPs.

As remarked in the introduction, Zadeh's rule has recently been shown to have superpolynomial worst case behaviour on linear programs. The parity games behind these LPs were not binary, so it does not immediately follow that Zadeh's rule has similar behaviour on AUSOs. This is a second open problem. More generally it is of interest to determine whether all of the history based rules mentioned in [1] have exponential behaviour on AUSOs.

Finally it would be of interest to study the AUSOs presented in Section 5 to see if they have a simple enough structure to be stated explicitly. If so it may be possible to prove the existence of exponentially long paths in a simpler manner than that done in Section 2.

7 Acknowledgements

We would like to thank the referees for detailed suggestions that helped us improve the original version of this paper.

References

1. Aoshima, Y., Avis, D., Deering, T., Matsumoto, Y., Moriyama, S.: On the existence of hamiltonian paths for history based pivot rules on acyclic unique sink orientations of hypercubes. Discrete Applied Mathematics **160**(15), 2104–2115 (2012)
2. Avis, D., Moriyama, S.: On Combinatorial Properties of Linear Programming Digraphs. In: D. Avis, D. Bremner, A. Deza (eds.) Polyhedral Computation, CRM Proceedings and Lecture Notes 48, pp. 1–13. AMS (2009)
3. Bertsekas, D.: Dynamic programming and optimal control, second edn. Athena Scientific (2001)
4. Chvátal, V.: Linear Programmming. W.H. Freeman (1983)
5. Cunningham, W.H.: Theoretical properties of the network simplex method. In: Mathematics of Operations Research, pp. 196–208 (1979)
6. Derman, C.: Finite state Markov decision processes. Academic Press (1972)
7. Emerson, E., Jutla, C.: Tree automata, μ-calculus and determinacy. In: Proc. 32nd Symp. on Foundations of Computer Science, pp. 368–377. IEEE, San Juan (1991)
8. Friedmann, O.: An exponential lower bound for the latest deterministic strategy iteration algorithms. Logical Methods in Computer Science **7**(3) (2011)
9. Friedmann, O.: A subexponential lower bound for Zadeh's pivoting rule for solving linear programs and games. In: IPCO, pp. 192–206 (2011)
10. Friedmann, O.: A subexponential lower bound for the least recently considered rule for solving linear programs and games. In: GAMES'2012. Naples, Italy (2012)
11. Friedmann, O.: (2013). http://files.oliverfriedmann.de/add/cunningham_exponential_parity_game.pdf
12. Gärtner, B.: The random-facet simplex algorithm on combinatorial cubes. Random Structures & Algorithms **20**(3), 353–381 (2002)
13. Gärtner, B., Schurr, I.: Linear programming and unique sink orientations. In: Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 749–757 (2006)
14. Howard, R.: Dynamic programming and Markov processes. MIT Press (1960)
15. Kalai, G.: A subexponential randomized simplex algorithm (extended abstract). In: STOC, pp. 475–482 (1992)

16. Klee, V., Minty, G.J.: How Good is the Simplex Algorithm? In: O. Shisha (ed.) Inequalities III, pp. 159–175. Academic Press Inc., New York (1972)
17. Matousek, J., Sharir, M., Welzl, E.: A subexponential bound for linear programming. In: Symposium on Computational Geometry, pp. 1–8 (1992)
18. Matousek, J., Szabó, T.: Random edge can be exponential on abstract cubes. In: FOCS, pp. 92–100 (2004)
19. Puri, A.: Theory of hybrid systems and discrete event systems. Ph.D. thesis, University of California, Berkeley (1995). URL http://www.eecs.berkeley.edu/Pubs/TechRpts/1995/2950.html
20. Puterman, M.: Markov decision processes. Wiley (1994)
21. Stickney, A., Watson, L.: Digraph models of bard-type algorithms for the linear complementarity problem. Mathematics of Operations Research **3**(4), 322–333 (1978). DOI 10.1287/moor.3.4.322
22. Szabó, T., Welzl, E.: Unique sink orientations of cubes. In: Proceedings of the 42th FOCS, pp. 547–555 (2001)
23. Vöge, J., Jurdzinski, M.: A discrete strategy improvement algorithm for solving parity games. In: Proc. 12th Int. Conf. on Computer Aided Verification, CAV'00, *LNCS*, vol. 1855, pp. 202–215. Springer (2000)
24. Zadeh, N.: What is the worst case behavior of the simplex algorithm. In: Polyhedral Computation, pp. 131–143. American Mathematical Society (2009,1980)
25. Zwick, U., Paterson, M.: The complexity of mean payoff games on graphs. Theoretical Computer Science **158**(1-2), 343–359 (1996). DOI http://dx.doi.org/10.1016/0304-3975(95)00188-3