A subexponential lower bound for Zadeh's pivoting rule for solving linear programs and games

Oliver Friedmann

Department of Computer Science,
University of Munich,
Germany
Oliver.Friedmann@gmail.com

Abstract. The *simplex* algorithm is among the most widely used algorithms for solving *linear programs* in practice. Most pivoting rules are known, however, to need an exponential number of steps to solve some linear programs. No non-polynomial lower bounds were known, prior to this work, for *Zadeh's* pivoting rule [25].
Also known as the Least-Entered rule, Zadeh's pivoting method belongs to the family of memorizing improvement rules, which among all improving pivoting steps from the current basic feasible solution (or *vertex*) chooses one which has been entered least often. We provide the first *subexponential* (i.e., of the form $2^{\Omega(\sqrt{n})}$) lower bound for this rule.
Our lower bound is obtained by utilizing connections between pivoting steps performed by simplex-based algorithms and *improving switches* performed by *policy iteration* algorithms for 1-player and 2-player games. We start by building 2-player *parity games* (PGs) on which the policy iteration with the Least-Entered rule performs a subexponential number of iterations. We then transform the parity games into 1-player *Markov Decision Processes* (MDPs) which corresponds almost immediately to concrete linear programs.

1 Introduction

The *simplex method*, developed by Dantzig in 1947 (see [5]), is among the most widely used algorithms for solving linear programs. One of the most important parameterizations of a simplex algorithm is the *pivoting rule* it employs. It specifies which non-basic variable is to enter the basis at each iteration of the algorithm. Although simplex-based algorithms perform very well in practice, essentially all *deterministic* pivoting rules are known to lead to an *exponential* number of pivoting steps on some LPs [21], [18], [1] and [15].

Kalai [19, 20] and Matoušek, Sharir and Welzl [22] devised *randomized* pivoting rules that never require more than an expected *subexponential* number of pivoting steps to solve any linear program. The most prominent randomized pivoting rules probably are Random-Facet [19, 20, 22] and Random-Edge [4, 13,

14], for which, until recently [12], no non-trivial lower bounds given by concrete linear programs were known.

An interesting deterministic pivoting rule for which no subexponential lower bound is known yet was suggested by Zadeh [25] (see also [7]). Also known as the Least-Entered rule, Zadeh's pivoting method belongs to the family of memorizing improvement rules, which among all improving pivoting steps from the current basic feasible solution (or *vertex*) chooses one which has been entered least often.

Here, we provide the first *subexponential* (i.e., of the form $2^{\Omega(\sqrt{n})}$) lower bound for the this rule.

Techniques used. The linear program on which Least-Entered performs a subexponential number of iterations is obtained using the close relation between simplex-type algorithms for solving linear programs and *policy iteration* (also known as *strategy improvement*) algorithms for solving certain 2-player and 1-player games.

This line of work was started by showing that standard strategy iteration [24] for parity games [16] may require an exponential number of iterations to solve them [9]. Fearnley [8] transfered the lower bound construction for parity games to *Markov Decision Processes* (MDPs) [17], an extremely important and well studied family of stochastic 1-player games.

In [11], we recently constructed PGs on which the Random-Facet algorithm performs an expected subexponential number of iterations. In [12], we applied Fearnley's technique to transform these PGs into MDPs, and include an additional lower bound construction for the Random-Edge algorithm.

The problem of solving an MDP, i.e., finding the optimal control *policy* and the optimal *values* of all states of the MDP, can be cast as a linear program. More precisely, the improving switches performed by the (abstract) Least-Entered algorithm applied to an MDP corresponds directly to the steps performed by the Least-Entered pivoting rule on the corresponding linear program.

Our results. We construct a family of concrete linear programs on which the number of iterations performed by Least-Entered is $2^{\Omega(\sqrt{n})}$, where n is the number of variables.

Here, we follow our approach from [12] to obtain a subexponential lower bound for Zadeh's pivoting rule by constructing concrete parity games on which the policy iteration algorithm parameterized with Zadeh's rule requires a subexponential number of iterations. Then, we transform the PGs into MDPs, and the linear programs corresponding to our MDPs supply, therefore, concrete linear programs on which following the Least-Entered pivoting rule leads to an expected subexponential number of iterations.

As the translation of our PGs to MDPs is a relatively simple step, we directly present the MDP version of our construction. (The original PGs from which our MDPs were derived can be found in Appendix C.) As a consequence, our construction can be understood without knowing anything about PGs.

In high level terms, our PGs, MDPs, and the linear programs corresponding to them, are constructions of 'pairwise alternating' binary counters. Consider a

normal binary counter: less significant bits are switched more often than higher bits, when counting from 0 to $2^n - 1$. Zadeh's rule would not go through all steps from 0 to $2^n - 1$ on such a counter, because higher bits will be switched before they are supposed to be switched, as the switching times that are associated with higher bits will catch up with the switching times associated with lower bits. Zadeh's rule, in a sense, requires a "fair" counter that operates correctly when all bits are switched equally often.

Our solution to this problem is to represent each bit i in the original counter by *two* bits i' and i'' s.t. only one of those two is actively working as representative for i. After switching the representative for i – say i' – from 0 to 1 and back to 0, we change the roles of i' and i'' s.t. i'' becomes the active representative for i. The inactive i' can now, while i'' switches from 0 to 1 and back to 0, catch up with the rest of the counter in terms of switching fairness: while i' is inactive, we switch i' from 0 to 1 back and forth (without effecting the rest of the counter as i' is the inactive representative) until the number of switching times catches up with the number of switching times of the rest of the counter again.

Another viable approach could be to implement more sophisticated binary counters like Gray codes (see e.g. [3]). However, the construction of an MDP or PG that models the behavior of a Gray code-based counter seems to be a very difficult task.

The rest of this paper is organized as follows. In Section 2 we give a brief introduction to *Markov Decision Processes* (MDPs) and the primal linear programs corresponding to them. In Section 3 we review the policy iteration and the simplex algorithms, the relation between improving switches and pivoting steps, and Zadeh's Least-Entered pivoting rule. In Section 4, which is the main section of this paper, we describe our lower bound construction for Least-Entered. Many of the details are deferred, due t o lack of space, to appendices. Particularly all proofs of Section 4 can be found in Appendix B. We end in Section 5 with some concluding remarks and open problems.

2 Markov Decision Processes and their linear programs

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 [17], Derman [6], Puterman [23] and Bertsekas [2].

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. As defined, the graph G is *bipartite*, but one can relax this condition and allow edges from V_0 to V_0 that correspond to *deterministic* actions.

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. $(u, \sigma(u)) \in E_0$ wWe assume that each vertex $u \in V_0$ has at least one outgoing edge). There are several *objectives* for MDPs; we consider the *expected total reward objective* here. The *values* $\textsc{val}_\sigma(u)$ of the vertices under σ are defined as the unique solutions of the following set of linear equations:

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

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

All MDPs considered in this paper satisfy a *weak* version of the *unichain* condition. The normal unichain condition (see [23]) states that the Markov chain obtained from each policy σ has a single irreducible recurrent class. We discuss the weak version at the end of this section.

Optimal policies for MDPs that satisfy the unichain condition can be found by solving the following (primal) linear program

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

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 *basic feasible solutions* (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 $x(u, v)$ 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., [23].)

Our MDPs only satisfy a *weak* version of the unichain condition, saying that the optimal policy has a single irreducible recurrent class. It follows that the optimal policy can be found by the same LPs when being started with an initial basic feasible solution corresponding to a policy with the same single irreducible recurrent class as the optimal policy. Then, by monotonicity, we know that all considered basic feasible solutions will have the same irreducible recurrent class.

It should be noted that *all* pivoting steps performed on these linear programs are *non-degenerate*, due to the fact the we consider the expected total reward

criterion here. The lower bound construction of this paper also works when applied to the *discounted reward criterion* (for large enough discount factors), and also for the *limiting average reward criterion*. However, in the latter case, all pivoting steps performed on the induced linear programs are *degenerate*.

3 Policy iteration algorithms and simplex algorithms

Howard's [17] *policy iteration* algorithm is the most widely used algorithm for solving MDPs. It is closely related to the simplex algorithm.

The 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 $\textsc{val}_{\sigma_i}(u)$ 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 $\textsc{val}_{\sigma_i}(v') > \textsc{val}_{\sigma_i}(u)$. Given a policy σ, we denote the *set of improving switches* by I_σ.

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., [17], [23]). Furthermore, if $(u, v') \in I_\sigma$ 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 $\textsc{val}_{\sigma'}(u) \geq \textsc{val}_{\sigma}(u)$, with a strict inequality for at least one vertex $u \in V_0$.

Policy iteration algorithms that perform a single switch at each iteration – like Zadeh's rule – are, in fact, simplex algorithms. Each policy σ of an MDP immediately gives rise to a feasible solution $x(u, v)$ of the primal linear program (P); 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 $x(u, v) = 0$.

Zadeh's Least-Entered pivoting rule is a *deterministic*, *memorizing* improvement rule which among all improving pivoting steps from the current basic feasible solution (or *vertex*) chooses one which has been entered least often. When applied to the primal linear program of an MDP, it is equivalent to the variant of the policy iteration algorithm, in which the improving switch is chosen among all improving switches to be one, which has been chosen least often. This is the foundation of our lower bound for the Least-Entered rule.

We describe Zadeh's pivoting rule now formally in the context of MDPs. As a memorization structure, we introduce an *occurrence record*, which is a map $\phi : E_0 \to \mathbb{N}$ that specifies for every player 0 edge of the given MDP how often it has been used. Among all improving switches in the set I_σ for a given policy σ, we need to choose an edge $e \in I_\sigma$ that has been selected least often. We denote the set of *least occurred improving switches* by $I_\sigma^\phi = \{e \in I_\sigma \mid \phi(e) \leq \phi(e') \text{ for all } e' \in I_\sigma\}$.

See Algorithm 1 for a pseudo-code specification of the Least-Entered pivoting rule for solving MDPs.

In the original specification of Zadeh's algorithm [25], there is no clear objective how to *break ties* whenever $|I_\sigma^\phi| > 1$. In fact, we know that the asymptotic

Algorithm 1 Zadeh's Improvement Algorithm

```
1: procedure LEAST-ENTERED(G,σ)
2:     φ(e) ← 0 for every e ∈ E_0
3:     while I_σ ≠ ∅ do
4:         e ← select edge from I_σ^φ
5:         φ(e) ← φ(e) + 1
6:         σ ← σ[e]
7:     end while
8: end procedure
```

behavior of Zadeh's improvement rule highly depends on the method that is used to break ties, at least in the world of MDPs, PGs and policy iteration 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 policy in each step).

Theorem 1. *Let G be an MDP with n nodes and σ_0 be a policy. There is a sequence policies $\sigma_0, \sigma_1, \ldots, \sigma_N$ and a sequence of* different *switches $e_1, e_2, \ldots, e_N$ with $N \leq n$ s.t. σ_{N-1} 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 break ties that results in a linear number of pivoting steps to solve an MDP with Zadeh's improvement rule. However, there is no obvious method of breaking ties. The question whether Zadeh's pivoting rule solves MDPs (and LPs) in polynomial time should therefore be phrased *independently* of the heuristic of breaking ties. In other words, we as "lower bound designers" are the ones that choose a particular tie breaking rule

Formally, we write $(\sigma, \phi) \rightsquigarrow (\sigma', \phi')$ iff there is an edge $e \in I_\sigma^\phi$ s.t. $\sigma' = \sigma[e]$ and $\phi' = \phi[e \mapsto \phi(e) + 1]$. Let $\rightsquigarrow^+$ denote the transitive closure of $\rightsquigarrow$. The question, whether Zadeh's improvement rule admits a polynomial number of iterations independently of the method of breaking ties is therefore equivalent to the question, whether the length of any sequence $(\sigma_0, \phi_0) \rightsquigarrow^+ \ldots \rightsquigarrow^+ (\sigma_N, \phi_N)$ can be polynomially bounded in the size of the game.

We will not specify the tie-breaking rule used for our lower bound explicitly, due to the fact that the rule itself is not a natural one. Instead, our proof just relies on the $\rightsquigarrow$-relation, witnessing in every improvement step that we only select an improving switch that has been applied least often.

4 Lower bound for LEAST-ENTERED

We start with a high-level description of the MDPs on which LEAST-ENTERED performs an expected subexponential number of iterations. As mentioned in the introduction, the construction may be seen as an implementation of a 'fair' counter. A schematic description of the lower bound MDPs is given in Figure 1.

Circles correspond to vertices of V_0, i.e., vertices controlled by player 0, while small rectangles correspond to the randomization vertices of V_R. We use the notation $k_{[i;j]}$ to indicate that player 0 in fact has edges to every node k_l with $i \leq l \leq j$.

The MDP of Figure 1 emulates an n-bit counter. It is composed of n identical levels, each corresponding to a single bit of the counter. The i-th level ($i = 1 \dots n$) is shown explicitly in the figure. Levels are separated by dashed lines. The MDP includes one *source* s and one *sink* t.

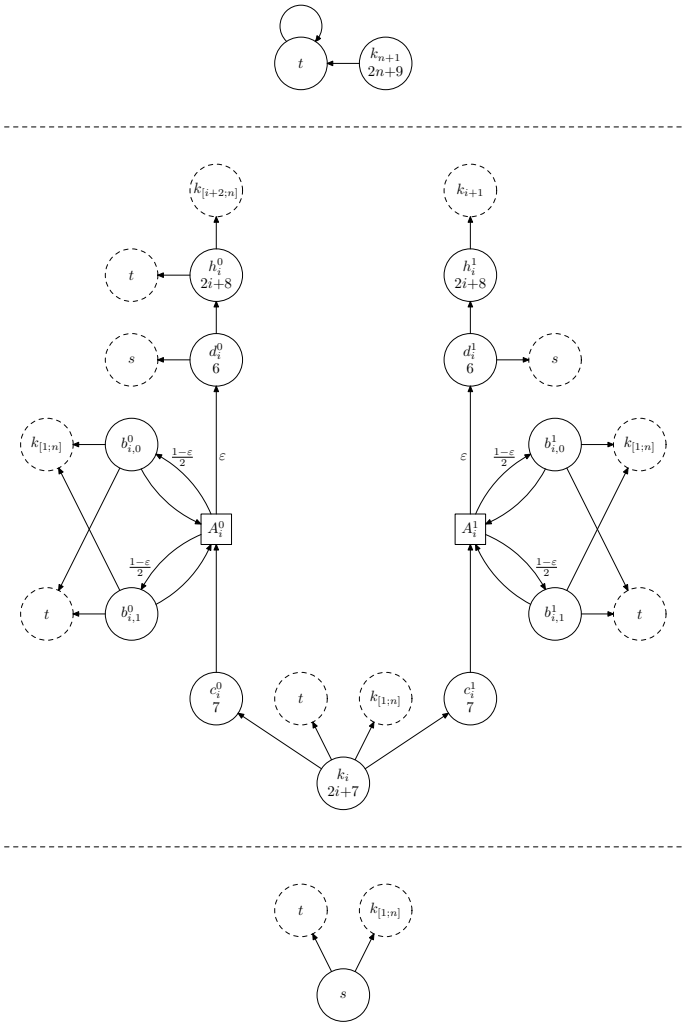

Fig. 1. Least Entered MDP Construction

All edges in Figure 1 have an immediate reward of 0 associated with them (such 0 rewards are not shown explicitly in the figure) unless stated otherwise as follows: 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, where N is a sufficiently large integer. We use $N \geq 7n+1$ and $\varepsilon \leq N^{-(2n+11)}$. Priorities, if present, are listed next to the vertex name. Note that it is profitable for the controller, to move through vertices of even priority and to avoid vertices of odd priority, and that vertices of higher numerical priority dominate vertices of lower priority (the idea of using priorities is inspired, of course, by the reduction from parity games to mean payoff games).

Each level i contains two (i.e. $j = 0, 1$) instances of a gadget that consists of a randomization vertex A_i^j and two (i.e. $l = 0, 1$) attached cycles with player 0 controlled nodes $b_{i,l}^j$. Therefore, we will call these gadgets from now on *bicycle gadgets*, and refer to the instance with $j = 0$ resp. $j = 1$ as to *bicycle* 0 resp. *bicycle* 1.

From A_i^j (with $j = 0, 1$), the edge $\mathrm{A}_i^j \to b_{i,l}^j$ (with $l = 0, 1$), is chosen with probability $\frac{1-\varepsilon}{2}$, while the edge $\mathrm{A}_i^j \to d_i^j$ is chosen with probability ε. Thus, if both $\sigma(b_{i,0}^j) = \mathrm{A}_i^j$ and $\sigma(b_{i,1}^j) = \mathrm{A}_i^j$, the MDP is guaranteed to eventually move from A_i^j to d_i^j (this is similar to the use of randomization by Fearnley [8]). We say that a bicycle gadget is

- *closed* iff both $\sigma(b_{i,0}^j) = \mathrm{A}_i^j$ and $\sigma(b_{i,1}^j) = \mathrm{A}_i^j$,
- *open* iff $\sigma(b_{i,0}^j) \neq \mathrm{A}_i^j$ or $\sigma(b_{i,1}^j) \neq \mathrm{A}_i^j$, and
- *completely open* iff $\sigma(b_{i,0}^j) \neq \mathrm{A}_i^j$ and $\sigma(b_{i,1}^j) \neq \mathrm{A}_i^j$.

Next, 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 $\mathcal{B}_n = \{\mathfrak{b} \in \{0,1\}^\infty \mid \forall i > n : \mathfrak{b}_i = 0\}$.

We start with index one, i.e. $\mathfrak{b} \in \mathcal{B}_n$ is essentially 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 $\mathfrak{b} \in \mathcal{B}$ and $k \in \mathbb{N}$ let $\mathfrak{b} + k$ denote the unique $\mathfrak{b}'$ s.t. $|\mathfrak{b}'| = \mathfrak{b} + k$. If $k \leq |\mathfrak{b}|$, let $\mathfrak{b} - k$ denote the unique $\mathfrak{b}'$ s.t. $|\mathfrak{b}'| + k = |\mathfrak{b}|$.

Given a configuration $\mathfrak{b}$, we access the *i-next set bit* by $\nu_i^n(\mathfrak{b}) = \min(\{n+1\} \cup \{j \geq i \mid \mathfrak{b}_j = 1\})$, and the *$i$-next unset bit* by $\mu_i(\mathfrak{b}) = \min\{j \geq i \mid \mathfrak{b}_j = 0\}$.

The i-th level of the MDP corresponds to the i-th bit. A set bit is represented by a *closed* bicycle gadget. Every level has two bicycle gadgets, but only one of them is *actively* representing the i-th bit.

Whether bicycle 0 or bicycle 1 is active in level i depends on the setting of the i+1-th bit. If it is set, i.e. $\mathfrak{b}_{i+1} = 1$, then bicycle 1 is active in the i-th level; otherwise, if $\mathfrak{b}_{i+1} = 0$, we have that bicycle 0 is active in the i-th level.

Our proof is conceptually divided into two parts. First we investigate the improving switches that can be performed from certain policies of the MDP. This allows us to prove the *existence* of a sequence of improving switches that indeed generates the sequence of policies $\sigma_{0\ldots00}, \sigma_{0\ldots01}, \sigma_{0\ldots10}, \ldots, \sigma_{1\ldots11}$. A transition from $\sigma_{\mathfrak{b}}$ to $\sigma_{\mathfrak{b}+1}$ involves many intermediate improvement steps. We partition the path leading from $\sigma_{\mathfrak{b}}$ to $\sigma_{\mathfrak{b}+1}$ into six sub-paths which we refer to as *phases*. In the following, we first give an informal description of the phases. The second part of our proof will be to show that the way we want to apply the improving switches is compliant with the associated occurrence records.

Before starting to describe what happens in the different phases, we describe the "ideal" configuration of a policy, which belongs to phase 1: (1) all active bicycles corresponding to set bits are closed, (2) all other bicycles are completely open, moving to the least set bit, (3) all entry points k_i move to the active bicycle if bit i is set and to the least set bit otherwise, (4) the source s moves to the least set bit, (5) all upper selection nodes h_i^0 move to the next *accessible* set bit (i.e. to the next set bit with index $\geq i$+2), and (6) the selection nodes d_i^j move higher up iff the immediately accessed bit is the next set bit (i.e. d_i^0 moves higher up iff $\mathfrak{b}_{i+1} = 0$ and d_i^1 moves higher up iff $\mathfrak{b}_{i+1} = 1$).

Note that the two upper selection nodes h_i^0 and h_i^1 cannot select the same entry points. The left node, h_i^0, can select from the entry points k_{i+2} up to k_n, while the right node, h_i^1, can only move to k_{i+1}. The intuition behind this is that bit i+1 is set every second time bit i is flipped, resulting in the alternating activation of the two bit representatives for i.

Now, we are ready to informally describe all phases.

1. At the beginning of the first phase, we only have open bicycles that are competing with each other to close. Inactive bicycles may have to catch up with active bicycles, and hence, are allowed to switch both player 0 edges inward, and therefore close the gadget. All active open bicycles move exactly one edge inward in this phase.
   So far, no active open bycycles have been closed. The last switch that is performed in this phase is to move the remaining edge of the active bicycle associated with the least unset bit inward, and therefore close the gadget.
2. In this phase, we need to make the recently set bit i accessible by the rest of the MDP, which will be via the k_i node. We switch here from k_i to c_i^j, where j denotes the active representative in this level.
   Note that k_i now has the highest value among all other k_*. Note that generally, k_l has a higher value than k_z for a set bit l and an unset bit z, and that k_l has a higher value than k_z for two set bits l and z iff $l < z$.
3. In the third phase, we perform the major part of the *resetting* process. By resetting, we mean to unset lower bits again, which corresponds to reopening the respective bicycles.

Also, we want to update all other inactive or active but not set bicycles again to move to the entry point k_i. In other words, we need to update the lower entry points k_z with $z < i$ to move to k_i, and the bicycle nodes $b^j_{z,l}$ to move to k_i.

We apply these switches by first switching the entry node k_z for some $z < i$, and then the respective bicycle nodes $b^j_{z,l}$.

4. In the fourth phase, we update the upper selection nodes h^0_z for all $z < i-1$ of the bits that have been reset. All nodes h^0_z should move to k_i.
5. In the fifth phase, we update the source node to finally move to the entry point corresponding to the recently set bit i.
6. In the last phase, we only have to update the selection nodes d^j_z for all $z < i$ of the bits that have been reset. We finally end up in a phase 1 policy again with the counter increased by one.

4.1 Full Construction

In this subsection, we formally describe the full construction of our MDPs. We define an underlying graph $G_n = (V_0, V_R, E_0, E_R, r, p)$ of an MDP as shown schematically in Figure 1 as follows:

$$V_0 := \{b^0_{i,0}, b^1_{i,0}, b^0_{i,1}, b^1_{i,1}, d^0_i, d^1_i, h^0_i, h^1_i, c^0_i, c^1_i \mid i \in [n]\} \cup \{k_i \mid i \in [n+1]\} \cup \{t, s\}$$
$$V_R := \{\mathrm{A}^0_i, \mathrm{A}^1_i \mid i \in [n]\}$$

With G_n, we associate a large number $N \in \mathbb{N}$ and a small number $0 < \varepsilon$. We require N to be at least as large as the number of nodes with priorities, i.e. $N \geq 7n+1$ and ε^{-1} to be significantly larger than the largest occurring priority induced reward, i.e. $\varepsilon \leq N^{-(2n+11)}$. Remember that node v having priority $\Omega(v)$ means that the cost associated with every outgoing edge of v is $\langle v \rangle = (-N)^{\Omega(v)}$.

Table 1 defines the edge sets, the probabilities, the priorities and the immediate rewards of G_n (note that h^0_i has the successors t, k_{i+2}, … k_n; particularly, h^0_n has only t as successor).

Node	Successors	Probability
A^j_i	d^j_i	ε
	$b^j_{i,0}$	$\frac{1}{2} \cdot (1-\varepsilon)$
	$b^j_{i,1}$	$\frac{1}{2} \cdot (1-\varepsilon)$

Node	Successors	Priority
t	t	-
s	$t, k_{[1;n]}$	-

Node	Successors	Priority
k_{n+1}	t	$2n$+9
k_i	$c^0_i, c^1_i, t, k_{[1;n]}$	$2i$+7
h^0_i	$t, k_{[i+2;n]}$	$2i$+8
h^1_i	k_{i+1}	$2i$+8
c^j_i	A^j_i	7
d^j_i	h^j_i, s	6
$b^j_{i,*}$	$t, \mathrm{A}^j_i, k_{[1;n]}$	-

Table 1. Least Entered MDP Construction

As designated *initial policy* σ^*, we use $\sigma^*(d_i^j) = h_i^j$, and $\sigma^*(\_) = t$ for all other player 0 nodes with non-singular out-degree. It is not hard to see that, starting with this initial policy, the MDP satisfies the weak unichain condition.

Lemma 1. *The Markov chains obtained by the initial and the optimal 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 belonging to the phases are bounded by ε^{-1}. More formally we have:

Lemma 2. *Let $P = \{k_*, h_*^*, c_*^*, d_*^*\}$ 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. *$|\sum(S)| < N^{2n+11}$ 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 $S, S' \subseteq P$.*

4.2 Lower Bound Proof

In this subsection, we formally describe the different *phases* that a policy can be in, as well as the improving switches in each phase. The increment of the binary counter by one is realized by transitioning through all the phases. Finally, we describe the corresponding occurrence records that appear in a run of the policy iteration on the MDPs.

We first introduce notation to succinctly describe policies. It will be convenient to describe the decision of a policy σ in terms of integers rather than concrete target vertices. Let σ be a policy. We define a function $\bar{\sigma}(v)$ as follows.

$\sigma(v)$	t	k_i	h_*^*	s	A_*^*	c_i^j
$\bar{\sigma}(v)$	$n+1$	i	1	0	0	$-j$

Additionally, we write $\bar{\sigma}(\mathrm{A}_i^j) = 1$ if $\sigma(b_{i,0}^j) = \mathrm{A}_i^j$ and $\sigma(b_{i,1}^j) = \mathrm{A}_i^j$, and $\bar{\sigma}(\mathrm{A}_i^j) = 0$ otherwise.

We are now ready to formulate the conditions for policies that fulfill one of the six phases along with the improving edges. See Table 2 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. Cells that contain more than one choice indicate that strategies of the respective phase are allowed to match any of the choices.

The following lemma tells us that all occurring values in the policy iteration are *small* compared to N^{2n+11}. Particularly, ε-times values are almost negligible.

Lemma 3. *Let σ be a policy belonging to one of the phases specified in Table 2. Then $|\mathrm{VAL}_\sigma(v)| < N^{2n+11}$ and $\varepsilon \cdot |\mathrm{VAL}_\sigma(v)| < 1$ for every node v.*

Phase	1	2	3	4	5	6
$\bar{\sigma}(s)$	r	r	r	r	r	r'
$\bar{\sigma}(d_i^0)$	$1-\mathfrak{b}_{i+1}$	$1-\mathfrak{b}_{i+1}$	$1-\mathfrak{b}_{i+1}$	$1-\mathfrak{b}_{i+1}$	$1-\mathfrak{b}_{i+1}$	$1-\mathfrak{b}_{i+1}$, $1-\mathfrak{b}'_{i+1}$
$\bar{\sigma}(d_i^1)$	$\mathfrak{b}_{i+1}$	$\mathfrak{b}_{i+1}$	$\mathfrak{b}_{i+1}$	$\mathfrak{b}_{i+1}$	$\mathfrak{b}_{i+1}$	$\mathfrak{b}_{i+1}$, $\mathfrak{b}'_{i+1}$
$\bar{\sigma}(h_i^0)$	$\nu^n_{i+2}(\mathfrak{b})$	$\nu^n_{i+2}(\mathfrak{b}$	$\nu^n_{i+2}(\mathfrak{b})$	$\nu^n_{i+2}(\mathfrak{b})$, $\nu^n_{i+2}(\mathfrak{b}')$	$\nu^n_{i+2}(\mathfrak{b}')$	$\nu^n_{i+2}(\mathfrak{b}')$
$\bar{\sigma}(b^*_{*,*})$	0, r	0, r	0, r, r'	0, r'	0, r'	0, r'
$\bar{\sigma}(\mathrm{A}_i^{\mathfrak{b}_{i+1}})$	$\mathfrak{b}_i$	$*$	$*$	$*$	$*$	$*$
$\bar{\sigma}(\mathrm{A}_i^{\mathfrak{b}'_{i+1}})$	$*$	$\mathfrak{b}'_i$	$\mathfrak{b}'_i$	$\mathfrak{b}'_i$	$\mathfrak{b}'_i$	$\mathfrak{b}'_i$

Phase	1–2	3	4–6
$\bar{\sigma}(k_i)$	$\begin{cases} r & \text{if } \mathfrak{b}_i = 0 \\ -\mathfrak{b}_{i+1} & \text{if } \mathfrak{b}_i = 1 \end{cases}$	$\begin{cases} r, r' & \text{if } \mathfrak{b}'_i = 0 \text{ and } \mathfrak{b}_i = 0 \\ -\mathfrak{b}_{i+1}, r' & \text{if } \mathfrak{b}'_i = 0 \text{ and } \mathfrak{b}_i = 1 \\ -\mathfrak{b}'_{i+1} & \text{if } \mathfrak{b}'_i = 1 \end{cases}$	$\begin{cases} r' & \text{if } \mathfrak{b}'_i = 0 \\ -\mathfrak{b}'_{i+1} & \text{if } \mathfrak{b}'_i = 1 \end{cases}$

Phase 3	Side Conditions
(a)	$\forall i.\ \Big(\big(\mathfrak{b}'_i = 0 \text{ and } (\exists j, l. \bar{\sigma}(b^j_{i,l}) = r')\big) \text{ implies } \bar{\sigma}(k_i) = r'\Big)$
(b)	$\forall i, j.\ \big(\mathfrak{b}'_i = 0, \mathfrak{b}'_j = 0, \bar{\sigma}(k_i) = r' \text{ and } \bar{\sigma}(k_j) \neq r'\big) \text{ implies } i > j\big)$

Table 2. Policy Phases (where $\mathfrak{b}' = \mathfrak{b} + 1$, $r = \nu^n_1(\mathfrak{b})$ and $r' = \nu^n_1(\mathfrak{b}')$)

Table 3 specifies the sets of improving switches by providing for each phase p a subset L^p_σ and a superset U^p_σ s.t. $L^p_\sigma \subseteq I_\sigma \subseteq U^p_\sigma$. The intuition behind this method of giving the improving switches is that we will only *use* switches from L^p_σ while making sure that *no* other switches from U^p_σ are applied.

We finally arrive at the following main lemma describing the improving switches.

Lemma 4. *The improving switches from policies that belong to the phases in Table 2 are bounded by those specified in Table 3, i.e. $L^p_\sigma \subseteq I_\sigma \subseteq U^p_\sigma$ for a phase p policy σ.*

Note that phase 1 policies do not say anything about the particular configuration of inactive or open bicycles. To specify that all bicycles are either closed or completely opened, we say that a phase 1 policy σ is an *initial phase 1* policy if $\bar{\sigma}(b^j_{i,l}) = 0$ iff $\mathfrak{b}_i = 1$ and $\mathfrak{b}_{i+1} = j$.

Next, we specify the occurrence records w.r.t. $\mathfrak{b} \in \mathcal{B}_n$ that we want to have for an initial phase 1 policy σ. As described earlier, the entries of the occurrence records essentially depend on the number of bit flips of a certain index that have happened while counting up to $\mathfrak{b}$.

More precisely, we need to be able to count the number of occurred bit settings that match a certain *scheme*, which is a description of how a certain bit configuration should look like. Formally, a *scheme* is a set $S \subseteq (\mathbb{N} \setminus \{0\}) \times \{0, 1\}$. Let $\mathfrak{b} \in \mathcal{B}$. We write $S \models \mathfrak{b}$ iff $\mathfrak{b}_i = q$ for all $(i, q) \in S$. We can now define the set of bit configurations leading to $\mathfrak{b}$ that *match* the scheme. Formally, we define the *match set* as $M(\mathfrak{b}, S) = \{\mathfrak{b}' \leq \mathfrak{b} \mid S \models \mathfrak{b}'\}$.

Ph. p	Improving Switches Subset L^p_σ	Improving Switches Superset U^p_σ
1	$\{(b^j_{i,l}, \mathrm{A}^j_i) \mid \sigma(b^j_{i,l}) \neq \mathrm{A}^j_i\}$	L^1_σ
2	$\{(k_{r'}, c^{\mathfrak{b}'_{r'+1}}_{r'})\}$	$L^1_\sigma \cup L^2_\sigma$
3	$\{(k_i, k_{r'}) \mid \bar{\sigma}(k_i) \neq r' \wedge \mathfrak{b}'_i = 0\} \cup$ $\{(b^j_{i,l}, k_{r'}) \mid \bar{\sigma}(b^j_{i,l}) \neq r' \wedge \mathfrak{b}'_i = 0\} \cup$ $\{(b^j_{i,l}, k_{r'}) \mid \bar{\sigma}(b^j_{i,l}) \neq r' \wedge \mathfrak{b}'_{i+1} \neq j\}$	$U^4_\sigma \cup \{(k_i, k_z) \mid \bar{\sigma}(k_i) \notin \{z, r'\}, z \leq r' \wedge \mathfrak{b}'_i = 0\} \cup$ $\{(b^j_{i,l}, k_z) \mid \bar{\sigma}(b^j_{i,l}) \notin \{z, r'\}, z \leq r' \wedge \mathfrak{b}'_i = 0\} \cup$ $\{(b^j_{i,l}, k_z) \mid \bar{\sigma}(b^j_{i,l}) \notin \{z, r'\}, z \leq r' \wedge \mathfrak{b}'_{i+1} \neq j\}$
4	$\{(h^0_i, k_{\nu^n_{i+2}(\mathfrak{b}')}) \mid \bar{\sigma}(h^0_i) \neq \nu^n_{i+2}(\mathfrak{b}')\}$	$U^5_\sigma \cup \{(h^0_i, k_l) \mid l \leq \nu^n_{i+2}(\mathfrak{b}')\}$
5	$\{(s, k_{r'})\}$	$U^6_\sigma \cup \{(s, k_i) \mid \bar{\sigma}(s) \neq i \wedge i < r'\} \cup$ $\{(d^j_i, x) \mid \sigma(d^j_i) \neq x \wedge i < r'\}$
6	$\{(d^0_i, x) \mid \sigma(d^0_i) \neq x \wedge \bar{\sigma}(d^0_i) \neq \mathfrak{b}'_{i+1}\} \cup$ $\{(d^1_i, x) \mid \sigma(d^1_i) \neq x \wedge \bar{\sigma}(d^1_i) = \mathfrak{b}'_{i+1}\}$	$L^1_\sigma \cup L^6_\sigma$

Table 3. Improving Switches (where $\mathfrak{b}' = \mathfrak{b} + 1$ and $r' = \nu^n_1(\mathfrak{b}')$)

We are most interested in schemes that correspond to flipping the i-th bit to one, i.e. schemes that demand for every bit $j < i$ to be zero. We define the *flip set* w.r.t. an index i and an additional scheme S by $F(\mathfrak{b}, i, S) = M(\mathfrak{b}, S \cup \{(i, 1)\} \cup \{(j, 0) \mid 0 < j < i\})$. We drop the parameter S if $S = \emptyset$.

We use the flip set to specify two numbers. First, we define the number of *bit flips* as the cardinality of the flip set by $f(\mathfrak{b}, i, S) = |F(\mathfrak{b}, i, S)|$. Second, we compute the *maximal flip number* representation in the flip set by $g(\mathfrak{b}, i, S) = \max(\{0\} \cup \{|\mathfrak{b}'| \mid \mathfrak{b}' \in F(\mathfrak{b}, i, S)\})$.

Table 4 specifies the occurrence record of an initial phase 1 policy. The technical conditions for the cycle components essentially say that (1) both cycle edges attached to A^j_i differ at most by one, that (2) the addition of both edges belonging to an active unset cycle equal $|\mathfrak{b}|$, that (3) the addition of both edges belonging to an active set cycle equal the maximal flip number when the respective bit was set, and that (4) recently opened inactive cycles are in the process of catching up with $|\mathfrak{b}|$ again.

Edge e	$(*, t)$	(s, k_r)	(h^0_*, k_r)	$(b^j_{i,*}, k_r)$
$\phi^{\mathfrak{b}}(e)$	0	$f(\mathfrak{b}, r)$	$f(\mathfrak{b}, r)$	$f(\mathfrak{b}, r, \{(i, 0)\}) + f(\mathfrak{b}, r, \{(i, 1), (i{+}1, 1{-}j)\})$

Edge e	(k_i, k_r)	(k_i, c^j_i)	(d^j_i, s)	(d^j_i, h^j_i)
$\phi^{\mathfrak{b}}(e)$	$f(\mathfrak{b}, r, \{(i, 0)\})$	$f(\mathfrak{b}, i, \{(i{+}1, j)\})$	$f(\mathfrak{b}, i{+}1) - j \cdot \mathfrak{b}_{i+1}$	$f(\mathfrak{b}, i{+}1) - (1{-}j) \cdot \mathfrak{b}_{i+1}$

Complicated Conditions

$$|\phi^{\mathfrak{b}}(b^j_{i,0}, \mathrm{A}^j_i) - \phi^{\mathfrak{b}}(b^j_{i,1}, \mathrm{A}^j_i)| \leq 1$$

$$\phi^{\mathfrak{b}}(b^j_{i,0}, \mathrm{A}^j_i) + \phi^{\mathfrak{b}}(b^j_{i,1}, \mathrm{A}^j_i) = \begin{cases} g^* + 1 & \text{if } \mathfrak{b}_i = 1 \text{ and } \mathfrak{b}_{i+1} = j \\ g^* + 1 + 2 \cdot z & \text{if } \mathfrak{b}_{i+1} \neq j \text{ and } z := |\mathfrak{b}| - g^* - 2^{i-1} < \frac{1}{2}(|\mathfrak{b}| - 1 - g^*), \\ |\mathfrak{b}| & \text{otherwise} \end{cases}$$

where $g^* = g(\mathfrak{b}, i, \{(i{+}1, j)\})$

Table 4. Occurrence Records

We are now ready to specify our main lemma describing the transitioning from an initial phase 1 policy corresponding to $\mathfrak{b}$ to a successor initial phase 1 policy corresponding to $\mathfrak{b}'$, complying with the respective occurrence records.

Lemma 5. *Let σ be an initial phase 1 policy with configuration $\mathfrak{b} < \mathbf{1}_n$. There is an initial phase 1 policy σ' with configuration $\mathfrak{b}' = \mathfrak{b}+1$ s.t. $(\sigma, \phi^{\mathfrak{b}}) \rightsquigarrow^+ (\sigma', \phi^{\mathfrak{b}'})$.*

It follows immediately that the MDPs provided here indeed simulate a binary counter.

Theorem 2. *The number of improving steps performed by* Least-Entered *on the MDPs constructed in this section, which contain $\mathcal{O}(n^2)$ vertices and edges, is $\Omega(2^n)$.*

The primal linear programs corresponding to the MDPs constructed in this section are thus linear programs on which the simplex algorithm with Zadeh's pivoting rule performs a subexponential number of iterations.

5 Concluding remarks and open problems

We have shown that Zadeh's Least-Entered rule [25] may lead to a subexponential number of iterations by constructing explicit linear programs with n variables on which the expected number of iterations performed by Least-Entered is $2^{\Omega(\sqrt{n})}$.

The lower bound for linear programming has been obtained by constructing explicit parity games and subsequently MDPs on which we have the same expected number of iterations when solved by policy iteration. The lower bound result immediately transfers to mean payoff games, discounted payoff games and turned-based simple stochastic games [10].

The tie-breaking rule that we employed to prove the lower bound was non-explicit and definitely not a natural one. It would be interesting to see, whether it is easily possible to transform the MDPs presented here, in order to obtain an exponential lower bound for Zadeh's rule with a natural tie-breaking rule.

The most interesting open problems are, perhaps, whether linear programs can be solved in strongly polynomial time, whether the weak Hirsch conjecture holds, and whether there is a polynomial time algorithm for solving parity games or related game classes.

Acknowledgments. I would like to thank Uri Zwick and Thomas Dueholm Hansen for pointing me to this challenging pivoting rule and for numerous inspiring discussions on the subject. Also, I would like to thank the anonymous referees for their thorough reports that me helped to improve the presentation of this paper.

References

1. Avis, D., Chvátal, V.: Notes on Bland's pivoting rule. In: Polyhedral Combinatorics, Mathematical Programming Studies, vol. 8, pp. 24–34. Springer (1978), http://dx.doi.org/10.1007/BFb0121192
2. Bertsekas, D.: Dynamic programming and optimal control. Athena Scientific, second edn. (2001)
3. Bhat, G.S., Savage, C.D.: Balanced gray codes. Electronic Journal of Combinatorics 3, 2–5 (1996)
4. Broder, A., Dyer, M., Frieze, A., Raghavan, P., Upfal, E.: The worst-case running time of the random simplex algorithm is exponential in the height. Inf. Process. Lett. 56(2), 79–81 (1995)
5. Dantzig, G.: Linear programming and extensions. Princeton University Press (1963)
6. Derman, C.: Finite state Markov decision processes. Academic Press (1972)
7. Fathi, Y., Tovey, C.: Affirmative action algorithms. Math. Program. 34(3), 292–301 (1986)
8. Fearnley, J.: Exponential lower bounds for policy iteration. In: Proc. of 37th ICALP. pp. 551–562 (2010)
9. Friedmann, O.: An exponential lower bound for the parity game strategy improvement algorithm as we know it. In: Proc. of 24th LICS. pp. 145–156 (2009)
10. Friedmann, O.: An exponential lower bound for the latest deterministic strategy iteration algorithms. Selected Papers of the Conference "Logic in Computer Science 2009" (to appear) (2010), a preprint available from http://www.tcs.ifi.lmu.de/~friedman
11. Friedmann, O., Hansen, T., Zwick, U.: A subexponential lower bound for the random facet algorithm for parity games. In: Proc. of 22nd SODA (2011), to appear
12. Friedmann, O., Hansen, T., Zwick, U.: Subexponential lower bounds for randomized pivoting rules for solving linear programs (2011), a preprint available from http://www.tcs.ifi.lmu.de/~friedman
13. Gärtner, B., Henk, M., Ziegler, G.: Randomized simplex algorithms on Klee-Minty cubes. Combinatorica 18(3), 349–372 (1998), http://dx.doi.org/10.1007/PL00009827
14. Gärtner, B., Tschirschnitz, F., Welzl, E., Solymosi, J., Valtr, P.: One line and n points. Random Structures & Algorithms 23(4), 453–471 (2003), http://dx.doi.org/10.1002/rsa.10099
15. Goldfarb, D., Sit, W.: Worst case behavior of the steepest edge simplex method. Discrete Applied Mathematics 1(4), 277 – 285 (1979), http://www.sciencedirect.com/science/article/B6TYW-45GVXJ1-2B/2/a7035da2cf84d35e9503c69f883c23f7
16. Grädel, E., Thomas, W., Wilke, T. (eds.): Automata, Logics, and Infinite Games. A Guide to Current Research, LNCS, vol. 2500. Springer (2002)
17. Howard, R.: Dynamic programming and Markov processes. MIT Press (1960)
18. Jeroslow, R.G.: The simplex algorithm with the pivot rule of maximizing criterion improvement. Discrete Mathematics 4(4), 367–377 (1973), http://www.sciencedirect.com/science/article/B6V00-45FSNXP-1H/2/0968f0b25d2d8a2e0e160a8a248d06de
19. Kalai, G.: A subexponential randomized simplex algorithm (extended abstract). In: Proc. of 24th STOC. pp. 475–482 (1992)

20. Kalai, G.: Linear programming, the simplex algorithm and simple polytopes. Mathematical Programming 79, 217–233 (1997)
21. Klee, V., Minty, G.J.: How good is the simplex algorithm? In: Shisha, O. (ed.) Inequalities III, pp. 159–175. Academic Press, New York (1972)
22. Matoušek, J., Sharir, M., Welzl, E.: A subexponential bound for linear programming. Algorithmica 16(4-5), 498–516 (1996)
23. Puterman, M.: Markov decision processes. Wiley (1994)
24. Vöge, J., Jurdziński, M.: A discrete strategy improvement algorithm for solving parity games (Extended abstract). In: International Conference on Computer-Aided Verification, CAV 2000. LNCS, vol. 1855, pp. 202–215 (2000)
25. Zadeh, N.: What is the worst case behavior of the simplex algorithm? Tech. Rep. 27, Department of Operations Research, Stanford (1980)

A Proofs of Section 3

Theorem 1.

Proof. Let G be an MDP and σ^* be the optimal policy. We define a *distance* function d that maps every policy to a natural number, counting in how many edges it differs from $\sigma*$. Formally:

$$d(\sigma) = |\{v \mid \sigma(v) \neq \sigma^*(v)\}|$$

To show the claim of the theorem, it suffices to prove that we can improve any non-optimal strategy σ by an improving edge e in one step s.t. $d(\sigma) > d(\sigma[e])$. Let therefore σ be a strategy with $d(\sigma) > 0$. Consider now the MDP G' restricted to σ and σ^*, i.e. $E'_0 = \{(v, w) \in E_0 \mid \sigma(v) = w \text{ or } \sigma^*(v) = w\}$.

It is easy to see that G' is a well-defined MDP, both σ and σ^* are policies in G' and that σ^* is still the optimal policy in G'. Since σ is not optimal in G', there must be an improving edge $e \in E'_0 \setminus \sigma$. Hence, $d(\sigma) > d(\sigma[e])$. □

B Proofs of Section 4

Lemma 3.

Proof. Let σ be a policy belonging to one of the phases specified in Table 2 and with configuration $\mathfrak{b}$. Let $\mathfrak{b}' = \mathfrak{b}+1$. Let $S_i = \sum_{j \geq i,\, \mathfrak{b}_j = 1} \left(\langle k_j \rangle + \langle c_j^0 \rangle + \langle d_j^0 \rangle + \langle h_j^0 \rangle\right)$ and $T_i = \sum_{j \geq i,\, \mathfrak{b}'_j = 1} \left(\langle k_j \rangle + \langle c_j^0 \rangle + \langle d_j^0 \rangle + \langle h_j^0 \rangle\right)$.

It suffices to show that $|\textsc{val}_\sigma(v)| < N^{2n+11}$ for every node v. Obviously, $\textsc{val}_\sigma(t) = 0$.

It is not too hard to see that the following holds:

$$\textsc{val}_\sigma(s) \in [S_1; T_1] \qquad\qquad \textsc{val}_\sigma(k_i) \in [\langle k_i \rangle + S_1; T_i]$$

We derive for all the other nodes that the following holds:

$$\begin{aligned}
\textsc{val}_\sigma(h_i^j) &\in [\langle h_i^j \rangle + \langle k_{i+1} \rangle + S_1; \langle h_i^j \rangle + T_{i+1}] \\
\textsc{val}_\sigma(d_i^j) &\in [\langle d_i^j \rangle + S_1; \langle d_i^j \rangle + \langle h_i^j \rangle + T_{i+1}] \\
\textsc{val}_\sigma(\mathrm{A}_i^j) &\in [S_1; \langle d_i^j \rangle + \langle h_i^j \rangle + T_{i+1}] \\
\textsc{val}_\sigma(b_{i,l}^j) &\in [S_1; \langle d_i^j \rangle + \langle h_i^j \rangle + T_{i+1}] \\
\textsc{val}_\sigma(c_i^j) &\in [\langle c_i^j \rangle + S_1; \langle c_i^j \rangle + \langle d_i^j \rangle + \langle h_i^j \rangle + T_{i+1}]
\end{aligned}$$

By Lemma 2, we have $|\textsc{val}_\sigma(v)| < N^{2n+11}$ for every node v. □

Next, we will specify and prove an auxiliary lemma that describes the exact behavior of all the bicycles appearing in the construction.

The idea behind the bicycles is to have a gate that controls the access of other nodes of the graph to the *escape node* of the bicycle (d_i^j) to which the randomized node moves with very low probability.

First, assume that both cycles attached to a node A_i^j are moving inward. Although the randomized node circles through the cycles with very high probability (without accumulating any costs), it eventually moves out to the escape node, resulting in the same value as the value of the escape node itself.

Second, assume that the bicycle is open, i.e. one of the V_0-controlled nodes of the bicycle decides to move out of the gadget to some *reset node*. Now, the randomized node selects to move into the cycle with very large probability and therefore leaves the cycle to the reset node with high probability as well. The resulting value of the randomized node essentially matches the value of the reset node.

The following lemma formalizes the intuition of the behavior of the bicycles. If the escape node has better valuation than the reset nodes, it should be profitable to close the bicycle, and otherwise, it should be profitable to open the bicycle again.

Lemma 6. *Let σ be a policy belonging to one of the phases specified in Table 2. Let $U = \{t, k_*\}$ and $u \in U$.*

1. $\bar{\sigma}(b_{i,l}^j) = 0$ *and* $\bar{\sigma}(b_{i,1-l}^j) = 0 \Rightarrow \mathrm{VAL}_\sigma(u) > \mathrm{VAL}_\sigma(d_i^j)$ *iff* $(b_{i,l}^j, u) \in I_\sigma$,
2. $\bar{\sigma}(b_{i,l}^j) \neq 0$, $\bar{\sigma}(b_{i,1-l}^j) \neq 0$ *and* $\bar{\sigma}(b_{i,l}^j) \neq \bar{\sigma}(b_{i,1-l}^j) \Rightarrow \mathrm{VAL}_\sigma(\sigma(b_{i,1-l}^j)) > \mathrm{VAL}_\sigma(\sigma(b_{i,l}^j))$ *iff* $(b_{i,l}^j, \mathrm{A}_i^j) \in I_\sigma$,
3. $\bar{\sigma}(b_{i,l}^j) \neq 0$ *and* $\bar{\sigma}(b_{i,1-l}^j) = \bar{\sigma}(b_{i,l}^j) \Rightarrow \mathrm{VAL}_\sigma(d_i^j) > \mathrm{VAL}_\sigma(\sigma(b_{i,l}^j))$ *iff* $(b_{i,l}^j, \mathrm{A}_i^j) \in I_\sigma$,
4. $\bar{\sigma}(b_{i,l}^j) \neq 0$ *and* $\bar{\sigma}(b_{i,1-l}^j) = 0 \Rightarrow \mathrm{VAL}_\sigma(d_i^j) > \mathrm{VAL}_\sigma(\sigma(b_{i,l}^j))$ *iff* $(b_{i,l}^j, \mathrm{A}_i^j) \in I_\sigma$,
5. $\bar{\sigma}(b_{i,l}^j) = 0$, $\bar{\sigma}(b_{i,1-l}^j) \neq 0$ *and* $\mathrm{VAL}_\sigma(d_i^j) > \mathrm{VAL}_\sigma(\sigma(b_{i,1-l}^j)) \Rightarrow \mathrm{VAL}_\sigma(u) > \mathrm{VAL}_\sigma(\sigma(b_{i,1-l}^j))$ *iff* $(b_{i,l}^j, u) \in I_\sigma$, *and*
6. $\bar{\sigma}(b_{i,l}^j) = 0$, $\bar{\sigma}(b_{i,1-l}^j) \neq 0$ *and* $\mathrm{VAL}_\sigma(d_i^j) < \mathrm{VAL}_\sigma(\sigma(b_{i,1-l}^j)) \Rightarrow \mathrm{VAL}_\sigma(u) \geq \mathrm{VAL}_\sigma(\sigma(b_{i,1-l}^j))$ *iff* $(b_{i,l}^j, u) \in I_\sigma$.

Proof. Let σ be a policy belonging to one of the phases specified in Table 2.

1. It follows that $\mathrm{VAL}_\sigma(\mathrm{A}_i^j) = \mathrm{VAL}_\sigma(d_i^j)$.
2. It follows that $\mathrm{VAL}_\sigma(\mathrm{A}_i^j) = \frac{1}{2}\mathrm{VAL}_\sigma(\sigma(b_{i,l}^j)) + \frac{1}{2}\mathrm{VAL}_\sigma(\sigma(b_{i,1-l}^j)) + \mathcal{O}(1)$.
3. It follows that $\mathrm{VAL}_\sigma(\mathrm{A}_i^j) = (1-\varepsilon)\mathrm{VAL}_\sigma(\sigma(b_{i,l}^j)) + \varepsilon\mathrm{VAL}_\sigma(d_i^j)$.
4. It follows that $\mathrm{VAL}_\sigma(\mathrm{A}_i^j) = \frac{1-\varepsilon}{1+\varepsilon}\mathrm{VAL}_\sigma(\sigma(b_{i,l}^j)) + \frac{2\varepsilon}{1+\varepsilon}\mathrm{VAL}_\sigma(d_i^j)$.
5. This can be shown the same way.
6. This can be shown the same way.

□

Finally, we prove that the improving switches are indeed exactly as specified. The simple but tedious proof uses Lemma 3 and Lemma 6 to compute the values of all important nodes in the game to determine whether a successor of V_0-controlled node is improving or not.

Lemma 4.

Proof. Let σ be a policy belonging to one of the phases with configuration $\mathfrak{b}$. We assume that σ is a phase 1 policy. The improving switches for the other phases can be shown the same way.

Let $S_i^l = \sum_{l \geq j \geq i,\, \mathfrak{b}_j = 1} \left(\langle k_j \rangle + \langle c_j^0 \rangle + \langle d_j^0 \rangle + \langle h_j^0 \rangle \right)$ and $S_i = S_i^n$.

First, we apply Lemma 2 and compute the values of all nodes.

Node	t	s	c_i^j	h_i^0	h_i^1
Value	0	S_1	$\langle c_i^j \rangle + \textsc{val}_\sigma(\mathrm{A}_i^j)$	$\langle h_i^0 \rangle + S_{i+2}$	$\langle h_i^1 \rangle + \textsc{val}_\sigma(k_{i+1})$

Node	k_i		d_i^j	
	$\mathfrak{b}_i{=}1$	$\mathfrak{b}_i{=}0$	$\mathfrak{b}_{i+1}{=}j$	$\mathfrak{b}_{i+1}{\neq}j$
Value	S_i	$\langle k_i \rangle + S_1$	$\langle d_i^j \rangle + \textsc{val}_\sigma(h_i^j)$	$\langle d_i^j \rangle + S_1$

Node	A_i^j		$b_{i,l}^j$	
	$\bar{\sigma}(\mathrm{A}_i^j){=}1$	$\bar{\sigma}(\mathrm{A}_i^j){\neq}1$	$\bar{\sigma}(\mathrm{A}_i^j){=}1$	$\bar{\sigma}(\mathrm{A}_i^j){\neq}1$
Value	$\textsc{val}_\sigma(d_i^j)$	$S_1 + \mathcal{O}(1)$	$\textsc{val}_\sigma(d_i^j)$	$S_1 + \mathcal{O}(1)$

Second, we observe the following ordering on the values of all "entry points" in the game graph.

1. $\mathfrak{b}_i = 1$ implies $\textsc{val}_\sigma(k_i) > \textsc{val}_\sigma(t)$,
2. $\mathfrak{b}_i = 1$ and $\mathfrak{b}_j = 0$ implies $\textsc{val}_\sigma(k_i) > \textsc{val}_\sigma(k_j)$,
3. $\mathfrak{b}_i = 1$, $\mathfrak{b}_j = 1$ and $i < j$ implies $\textsc{val}_\sigma(k_i) > \textsc{val}_\sigma(k_j)$.

Third, we derive that there are no improving switches for s and h_i^0. Fourth, we compute the differences between the values of the successors of d_i^j to see that there are no improving switches for these nodes.

Difference	$\textsc{val}_\sigma(h_i^0) - \textsc{val}_\sigma(s)$		$\textsc{val}_\sigma(h_i^1) - \textsc{val}_\sigma(s)$	
	$\mathfrak{b}_{i+1}{=}1$	$\mathfrak{b}_{i+1}{=}0$	$\mathfrak{b}_{i+1}{=}1$	$\mathfrak{b}_{i+1}{=}0$
Value	$\langle h_i^0 \rangle - S_1^{i+1} < 0$	$\langle h_i^0 \rangle - S_1^i > 0$	$\langle h_i^1 \rangle - S_1^i > 0$	$\langle h_i^1 \rangle + \langle k_{i+1} \rangle < 0$

Fifth, we show that there are no improving switches for the entry points k_i by computing the value differences between S_1 and c_i^j if $\mathfrak{b}_i = 0$ and additionally between c_i^j and c_i^{1-j} if $\mathfrak{b}_i = 1$.

Difference	$\textsc{val}_\sigma(c_i^j) - \textsc{val}_\sigma(c_i^{1-j})$ $\mathfrak{b}_i = 1, \mathfrak{b}_{i+1} = j$	
	$\bar{\sigma}(\mathrm{A}_i^{1-j}){=}1$	$\bar{\sigma}(\mathrm{A}_i^{1-j}){=}0$
Value	$\langle h_i^0 \rangle - S_1^i > 0$	$\langle h_i^0 \rangle + \langle d_i^j \rangle - S_1^i + \mathcal{O}(1) > 0$

Difference	$\textsc{val}_\sigma(c_i^j) - S_1$		
	$\mathfrak{b}_i = 1, \mathfrak{b}_{i+1} = j$	$\mathfrak{b}_i = 0$	
		$\bar{\sigma}(\mathrm{A}_i^j){=}1$	$\bar{\sigma}(\mathrm{A}_i^j){=}0$
Value	$\langle h_i^j \rangle + \langle d_i^j \rangle - S_1^i > 0$	$\langle c_i^j \rangle + \langle d_i^j \rangle < 0$	$\langle c_i^j \rangle + \mathcal{O}(1) < 0$

Finally, we consider all $b_{i,l}^j$ nodes and show that the set of improving switches is indeed $\{(b_{i,l}^j, \mathrm{A}_i^j) \mid \sigma(b_{i,l}^j) \neq \mathrm{A}_i^j\}$. Therefore, we compute the value difference between d_i^j and S_1, and apply Lemma 6.

Difference	$\text{VAL}_\sigma(d_i^j) - S_1$	
	$\mathfrak{b}_{i+1} = j$	$\mathfrak{b}_{i+1} \neq j$
Value	$\langle d_i^j \rangle + \langle h_i^j \rangle - S_1^i > 0$	$\langle d_i^j \rangle > 0$

□

Lemma 5.

Proof. Let σ_1 be an initial phase 1 policy with configuration $\mathfrak{b} < \mathbf{1}_n$. Let $\mathfrak{b}' = \mathfrak{b} + 1$ and $r = \mu_1(\mathfrak{b})$. Let $\phi_1 = \phi^{\mathfrak{b}}$.

The idea of this lemma is to undergo all six phases of Table 2 while performing improving switches towards the desired subsequent occurrence record.

More formally: we construct additional $(\sigma_2, \phi_2), \ldots, (\sigma_7, \phi_7)$ s.t.

- $(\sigma_p, \phi_p) \rightsquigarrow^+ (\sigma_{p+1}, \phi_{p+1})$,
- σ_p is in phase p with configuration $\mathfrak{b}$ if $p < 7$, and
- $\phi_7 = \phi^{\mathfrak{b}'}$ and σ_7 is an initial phase 1 policy with configuration $\mathfrak{b}'$

The construction is now as follows. We implicitly apply Lemma 4 when referring to the improving switches of a phase.

1. The only improving switches in this phase are from $b_{i,l}^j$ to A_i^j. This will be the only phase in which we will be making any switches of this kind.
   The first observation to make is that $g(\mathfrak{b}, i, \{(i{+}1, j)\}) = g(\mathfrak{b}', i, \{(i{+}1, j)\})$ if $i \neq r$.
   First, there are bicycles s.t. $\mathfrak{b}_i = 1$ and $\mathfrak{b}_{i+1} = j$, hence they are already closed, hence we cannot increase their respective occurrence records. In other words, we need to show that $\phi_1(b_{i,l}^j, \mathrm{A}_i^j) = \phi_7(b_{i,l}^j, \mathrm{A}_i^j)$.
   If $\mathfrak{b}'_i = 1$, i.e. $i > r$, it follows by $g(\mathfrak{b}, i, \{(i{+}1, j)\}) = g(\mathfrak{b}', i, \{(i{+}1, j)\})$ that $\phi_1(b_{i,l}^j, \mathrm{A}_i^j) = \phi_7(b_{i,l}^j, \mathrm{A}_i^j)$.
   Otherwise, if $\mathfrak{b}'_i = 0$, i.e. $i < r$, it follows that $\phi_7(b_{i,l}^j, \mathrm{A}_i^j) = g(\mathfrak{b}', i, \{(i{+}1, j)\}) + 1 + 2 \cdot (|\mathfrak{b}'| - g(\mathfrak{b}', i, \{(i{+}1, j)\}) - 2^{i-1})$. In other words, we need to show that $|\mathfrak{b}'| - g(\mathfrak{b}', i, \{(i{+}1, j)\}) = 2^{i-1}$. And this is true, because it required 2^{i-1} counting steps to count with all the lower bits.
   Second, there are bicycles s.t. $\mathfrak{b}_{i+1} \neq j$ and $\phi_1(b_{i,0}^j, \mathrm{A}_i^j) + \phi_1(b_{i,1}^j, \mathrm{A}_i^j) < |\mathfrak{b}|$. We will see that, $i \neq r$. Hence, we know that $g(\mathfrak{b}, i, \{(i{+}1, j)\}) = g(\mathfrak{b}', i, \{(i{+}1, j)\})$. In this case, we have $\phi_1(b_{i,l}^j, \mathrm{A}_i^j) + 2 = \phi_7(b_{i,l}^j, \mathrm{A}_i^j)$. Hence, by flipping both edges of these bicycles, we can make sure that we comply to the objective occurrence record.
   Third, there are bicycles s.t. $\mathfrak{b}_i = 0$ or $\mathfrak{b}_{i+1} \neq j$ that have $\phi_1(b_{i,0}^j, \mathrm{A}_i^j) + \phi_1(b_{i,1}^j, \mathrm{A}_i^j) = |\mathfrak{b}|$. Obviously, r belongs to this class of bicycles. It is easy to see that $\phi_1(b_{i,l}^j, \mathrm{A}_i^j) + 1 = \phi_7(b_{i,l}^j, \mathrm{A}_i^j)$ for $i \neq r$ and $\phi_1(b_{i,l}^j, \mathrm{A}_i^j) + 2 = \phi_7(b_{i,l}^j, \mathrm{A}_i^j)$ for $i = r$. Hence, by switching one edge for all $i \neq r$ and both edges for $i = r$, we can make sure that we comply to the objective occurrence record.
   The order in which all switches are to be performed is therefore as follows. We close both edges of all second class bicycles and one edge of every third class bicycle. Finally, we close the second edge of bicycle r.
   We now have, for all *open* bicycles, that $\phi_2(b_{i,0}^j, \mathrm{A}_i^j) + \phi_2(b_{i,1}^j, \mathrm{A}_i^j) = |\mathfrak{b}'|$.

2. The only improving switches in this phase are still from $b^j_{i,l}$ to A^j_i, and from k_r to $c_r^{\mathfrak{b}'_{r+1}}$.
   Is easy to see that $2f(\mathfrak{b}', r, \{(r{+}1, \mathfrak{b}'_{r+1})\}) \leq |\mathfrak{b}'|$, hence we can ensure to make that switch without closing any additional bicycles.
   Also note that $\phi_2(k_r, c_r^{\mathfrak{b}'_{r+1}}) + 1 = \phi_7(k_r, c_r^{\mathfrak{b}'_{r+1}})$, and for all other edges $(i,j) \neq (r, \mathfrak{b}'_{r+1})$ of this kind we have $\phi_2(k_i, c^j_i) = \phi_7(k_i, c^j_i)$.
3. In this phase, there are many improving switches. In order to fulfill all side conditions for phase 3, we need to perform all switches from higher indices to smaller indices, and k_i to k_r before $b^j_{i,l}$ with $\mathfrak{b}'_{i+1} \neq j$ or $\mathfrak{b}'_i = 0$ to k_r.
   The reader can easily check from that we can perform the switches in the desired ordering.

4.-6. These can be shown similarly.

□

C Parity Games

In this section, we show how the lower bound graphs can be turned into a parity game to provide a lower bound for this class of games as well.

We just give a formal specification of parity games to fix the notation. For a proper description of parity games, related two-player game classes and policy iteration on these games, please refer to [11] and [10].

A *parity game* is a tuple $G = (V_0, V_1, E, \Omega)$, where V_0 is the set of vertices controlled by player 0, V_1 is the set of vertices controlled by player 1, $E \subseteq V \times V$, where $V = V_0 \cup V_1$, is the set of edges, and $\Omega : V \to \mathbb{N}$ is a function that assigns a *priority* to each vertex. We assume that each vertex has at least one outgoing edge.

We say that G is a *sink parity game* iff there is a node $v \in V$ such that $\Omega(v) = 1$, $(v, v) \in E$, $\Omega(w) > 1$ for every other node $w \in V$, v is the only cycle in G that is won by player 1, and player 1 has a winning policy for the whole game.

Theorem 3 ([10]). *Let G be a sink parity game. Discrete policy iteration requires the same number of iterations to solve G as the policy iteration for the induced payoff games as well as turn-based stochastic games to solve the respective game G' induced by applying the standard reduction from G to the respective game class, assuming that the improvement policy solely depends on the ordering of the improving edges.*

Essentially, the graph is exactly the same. Randomization nodes are replaced by player 1 controlled nodes s.t. the cycles are won by player 0. We assign low unimportant priorities to all nodes that have currently no priority.

The correspondence between a vertex A controlled by player 1 in the parity game and the vertex controlled by the randomization player in the MDP is shown in Figure 2. Suppose player 1 does not move left unless player 0 moves from both

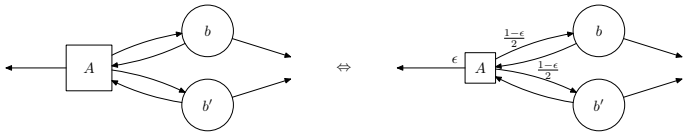

Fig. 2. Conversion of a vertex controlled by player 1 to a randomization vertex and vice versa.

b and b' to A. This behavior of player 1 can be simulated by a randomization vertex that moves left with very low, but positive probability.

We define the underlying graph $G_n = (V_0, V_1, E, \Omega)$ of a parity game as shown schematically in Figure 3. More formally:

$$V_0 := \{b^0_{i,0}, b^1_{i,0}, b^0_{i,1}, b^1_{i,1}, d^0_i, d^1_i, h^0_i, h^1_i, c^0_i, c^1_i \mid i \in [n]\} \cup \{k_i \mid i \in [n+1]\} \cup \{t, s\}$$
$$V_1 := \{\mathrm{A}^0_i, \mathrm{A}^1_i \mid i \in [n]\}$$

Table 5 defines the edge sets and the priorities of G_n.

Node	Successors	Priority
d^j_i	h^j_i, s	6
A^j_i	$d^j_i, b^j_{i,0}, b^j_{i,1}$	4
$b^j_{i,*}$	$t, \mathrm{A}^j_i, k_{[1;n]}$	3
t	t	3
s	$t, k_{[1;n]}$	3

Node	Successors	Priority
k_{n+1}	t	$2n{+}9$
k_i	$c^0_i, c^1_i, t, k_{[1;n]}$	$2i{+}7$
h^0_i	$t, k_{[i+2;n]}$	$2i{+}8$
h^1_i	k_{i+1}	$2i{+}8$
c^j_i	A^j_i	7

Table 5. Least Entered PG Construction

The first important observation to make is that the parity game is a sink game, which helps us to transfer our result to mean payoff games, discounted payoff games as well as turned-based simple stochastic games. The following lemma corresponds to Lemma 1 in the MDP world.

Lemma 7. *Starting with the designated initial policy of Section 4, we have that G_n is a sink parity game.*

All other definitions are exactly as in Section 4. Particularly, Table 2 and Table 3 become applicable again. The following lemma has the exact same formulation as Lemma 4 in the MDP world.

Lemma 8. *The improving switches from policies that belong to the phases in Table 2 are bounded by those specified in Table 3, i.e. $L^p_\sigma \subseteq I_\sigma \subseteq U^p_\sigma$ for a phase p policy σ.*

The reason why this lemma holds is that the valuations of the parity game nodes are essentially the same as the values in the MDP by dropping unimportant $\mathcal{O}(1)$ terms.

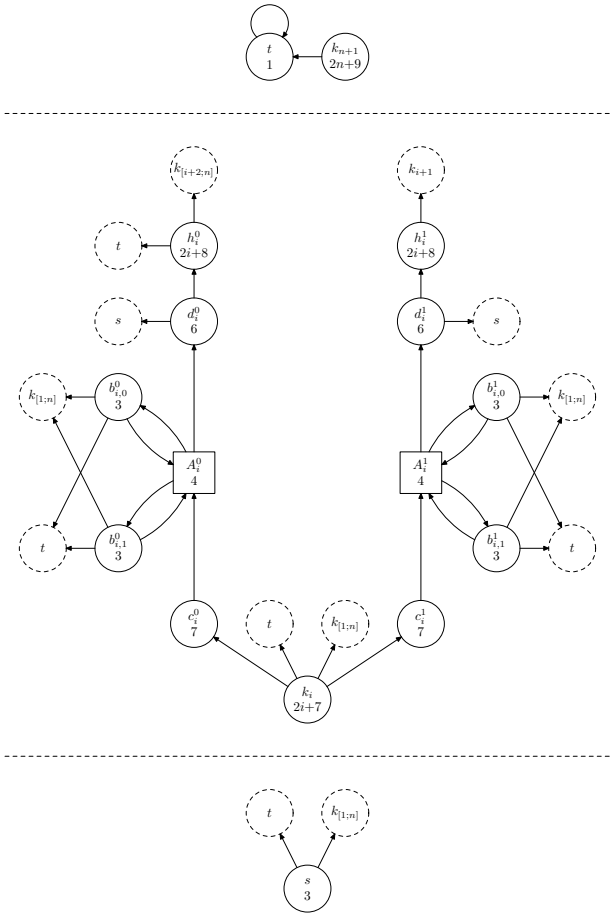

Fig. 3. Least Entered Parity Game Construction

All other proofs in Section 4 rely on Table 2, Table 3 and Lemma 4, hence we transfer our main theorem to the parity game world.

Theorem 4. *The worst-case expected running time of the* Least-Entered *algorithm for n-state parity games, mean payoff games, discounted payoff games and turn-based simple stochastic games is subexponential.*