Subexponential lower bounds for randomized pivoting rules for solving linear programs

Oliver Friedmann [*] *Thomas Dueholm Hansen* [†] *Uri Zwick* [‡]

Abstract

The *simplex* algorithm is among the most widely used algorithms for solving *linear programs* in practice. Most *deterministic* 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 *randomized* pivoting rules. We provide the first *subexponential* (i.e., of the form $2^{\Omega(n^\alpha)}$, for some $\alpha > 0$) lower bounds for the two most natural, and most studied, randomized pivoting rules suggested to date.

The first randomized pivoting rule we consider is RANDOM-EDGE, which among all improving pivoting steps (or *edges*) from the current basic feasible solution (or *vertex*) chooses one uniformly at random. The second randomized pivoting rule we consider is RANDOM-FACET, a more complicated randomized pivoting rule suggested by Matoušek, Sharir and Welzl [MSW96]. Our lower bound for the RANDOM-FACET pivoting rule essentially matches the subexponential upper bound of Matoušek *et al.* [MSW96]. Lower bounds for RANDOM-EDGE and RANDOM-FACET were known before only in *abstract* settings, and not for concrete linear programs.

Our lower bounds are 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 suitable randomized policy iteration algorithms perform a subexponential number of iterations. We then transform these 2-player games into 1-player *Markov Decision Processes* (MDPs) which correspond almost immediately to concrete linear programs.

[*]Department of Computer Science, University of Munich, Germany. E-mail: `Oliver.Friedmann@gmail.com`.

[†]Department of Computer Science, Aarhus University, Denmark. Supported by the Center for Algorithmic Game Theory, funded by the Carlsberg Foundation. E-mail: `tdh@cs.au.dk`.

[‡]School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel. Research supported by grant no. 1306/08 of the Israel Science Foundation. E-mail: `zwick@tau.ac.il`.

1 Introduction

Linear programming is one of the most important computational problems studied by researchers in computer science, mathematics and operations research. Though our understanding of linear programming improved vastly in the last 60 years, there are still many extremely intriguing and important open problems. Dozens of books and thousands of articles were written on linear programming. See, e.g., Chvátal [Chv83], Schrijver [Sch86] and Matoušek and Gärtner [MG07] and the references there in.

The *simplex method*, developed by Dantzig in 1947 (see [Dan63]), and its many variants, are still among the most widely used algorithms for solving linear programs. One of the most important characteristics of a simplex algorithm is the *pivoting rule* it employs. (The pivoting rule determines 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. This was first established for Dantzig's original pivoting rule by Klee and Minty [KM72]. Similar results for many other rules were obtained by [Jer73],[AC78] and [GS79]. For a unified view of these constructions see Amenta and Ziegler [AZ96]. An interesting deterministic pivoting rule for which no exponential lower bound is known yet was suggested by Zadeh [Zad80] (see also [FT86]).

It is not known whether there exists a pivoting rule that requires a polynomial number of pivoting steps on *any* linear program. This is, perhaps, the most important open problem in the field of linear programming. The existence of such a polynomial pivoting rule would imply, of course, that the *diameter* of the edge-vertex graph of any polytope is polynomial in the number of facets defining it. The Hirsch conjecture (see, e.g., [Dan63], pp. 160,168), which states that the diameter of the graph defined by an n-facet d-dimensional polytope is at most $n - d$ has recently been refuted by Santos [San10]. The best upper bound known on the diameter is a quasi-polynomial bound (i.e., of the form $n^{\mathcal{O}(\log n)}$) obtained by Kalai and Kleitman [KK92]. The *polynomial Hirsch conjecture*, the weaker form of the conjecture that just stipulates that the diameter is polynomial, is now the focus of the *polymath3* project.

Although no polynomial versions of the simplex algorithm are known, linear programs can be solved in polynomial time using either the *ellipsoid* algorithm of Khachiyan [Kha79], or the *interior-point* algorithm of Karmarkar [Kar84]. (For more on the ellipsoid algorithms and its combinatorial consequences, see Grötschel *et al.* [GLS88]. For more on interior-point algorithms, see Nesterov and Nemirovskii [NN94] and Ye [Ye97].) The ellipsoid and interior-point algorithms are polynomial, but are not *strongly polynomial*, i.e., their running times depend on the numerical values of the coefficients appearing in the program, even in the *unit-cost* model in which each arithmetical operation is assumed to take constant time. Furthermore, the ellipsoid and the interior-point algorithms have a strong numerical flavor, as opposed to the more combinatorial flavor of simplex algorithms. It is another major open problem whether there exists a strongly polynomial time algorithm for solving linear programs. A polynomial pivoting rule for the simplex algorithm would also provide a positive answer to this open problem.

Kalai [Kal92, Kal97] and Matoušek, Sharir and Welzl [MSW96] devised *randomized* pivoting rules that never require more than an expected *subexponential* number of pivoting steps to solve any linear program. More specifically, the expected number of steps performed by their algorithms is at most $2^{O(\sqrt{n})}$, where n is the number of constraints in the linear program. Their algorithms can, in fact, be used to solve a more general class of problems known as *LP-type* problems. (See also Gärtner [Gär95]). In this more general setting, Matoušek [Mat94] (see also [Gär02]) constructed *Acyclic Unique Sink Orientations* (AUSOs) of combinatorial cubes, which form a subfamily of LP-type problems, on which the algorithm of [MSW96] may require an almost matching expected subexponential number of iterations to find the *sink*. (The sink in this abstract setting corresponds

to the optimal vertex of a linear program.) It was not known, however, whether such subexponential behavior may also occur on linear programs. Gärtner [Gär02] showed that on all linear programs that belong to the abstract family of problems constructed by Matoušek [Mat94], the algorithm of [MSW96] actually runs in quadratic time.

The pivoting rules of Kalai [Kal92, Kal97] and of Matoušek *et al.* [MSW96] are in a sense dual to each other (see Goldwasser [Gol95]). We focus here on the pivoting rule of Matoušek *et al.* [MSW96] and refer to it as the RANDOM-FACET rule.

Perhaps the most natural randomized pivoting rule is RANDOM-EDGE, which among all improving edges from the current vertex chooses one uniformly at random. The upper bounds currently known for RANDOM-EDGE are still exponential (see Gärtner and Kaibel [GK07]). (For additional results regarding RANDOM-EDGE, see [BDF$^+$95],[GHZ98],[GTW$^+$03],[BP07].) RANDOM-EDGE is also applicable in a much wider abstract setting. Matoušek and Szabó [MS06] showed that it can be subexponential on AUSOs. It was again not known whether such subexponential behavior could also occur on actual linear programs.

Our results. We show that both RANDOM-EDGE and RANDOM-FACET may lead to an expected subexponential number of iterations on actual linear programs. More specifically, we construct concrete linear programs on which the expected number of iterations performed by RANDOM-EDGE is $2^{\Omega(n^{1/4})}$, where n is the number of variables, and (different) linear programs on which the expected number of iterations performed by RANDOM-FACET is $2^{\Omega(\sqrt{n}/\log^c n)}$, for some fixed $c > 0$.

The study of randomized pivoting rules should not be confused with the study of the *average case* behavior of the simplex algorithm, with deterministic pivoting rules, under various probability distributions. (For more on this subject, see, e.g., [Sma83], [AM85], [Tod86], [AKS87], [Bor91].) Our results are also orthogonal to the *smoothed analysis* of the simplex algorithm performed by Spielman and Teng [ST04], which may be used to explain why the simplex algorithm behaves so well in practice. Kelner and Spielman [KS06] use the results of [ST04] to obtain a new polynomial time algorithm for linear programming. Their algorithm applies the simplex algorithm to a suitably transformed linear program, so it does not resolve the two major open problems mentioned above.

Techniques used. The linear programs on which RANDOM-EDGE and RANDOM-FACET perform an expected subexponential number of iterations are 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.

Friedmann [Fri09] started the line of work pursued here by showing that the standard strategy iteration algorithm, which performs all improving switches *simultaneously*, may require an exponential number of iterations to solve certain *parity games* (PGs). Parity games (see, e.g., [EJ91],[EJS93],[Sti95],[GTW02]) form an intriguing family of deterministic 2-player games whose solution is equivalent to the solution of important problems in automatic verification and automata theory.

Fearnley [Fea10] adapted Friedmann's construction to work for *Markov Decision Processes* (MDPs), an extremely important and well studied family of stochastic 1-player games. (For more on MDPs, see Howard [How60], Derman [Der72] and Puterman [Put94].) (The paper of Melekopoglou and Condon [MC94] may be seen as a precursor of the papers of Friedmann [Fri09] and Fearnley [Fea10]. It deals, however, only with deterministic policy iteration algorithms that perform one switch at a time, and proving exponential lower bounds for such algorithms is a much easier task.)

In [FHZ11], we recently constructed PGs on which the RANDOM-FACET algorithm performs an expected subexponential number of iterations. Here, we use Fearnley's technique to transform these PGs into MDPs. The problem of solving an MDP, i.e., finding the optimal control *policy* and the optimal *values* and *potentials* of all states of the MDP, can be cast as a linear program. Furthermore, the improving switches performed by the (abstract) RANDOM-FACET algorithm applied

to an MDP correspond directly to the steps performed by the RANDOM-FACET pivoting rule on the corresponding linear program. (Assuming, of course, that the same random choices are made by both algorithms.) The linear programs corresponding to our MDPs supply, therefore, concrete linear programs on which following the RANDOM-FACET pivoting rule leads to an expected subexponential number of iterations.

To obtain concrete linear programs on which the simplex algorithm with the RANDOM-EDGE pivoting rule performs an expected subexponential number of pivoting steps, we follow a similar path. We start by constructing PGs on which the (abstract) RANDOM-EDGE algorithm performs an expected subexponential number of iterations. We convert these PGs into MDPs, and then to concrete linear programs. Although the conceptual path followed is similar, the concrete constructions used for RANDOM-EDGE are completely different, and somewhat more complicated, than the ones used here and in [FHZ11] for RANDOM-FACET. We view the construction used for RANDOM-EDGE as the *main novel result* of the paper and most of this extended abstract is devoted to its description.

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 D.) As a consequence, our construction can be described and understood without knowing anything about PGs. We would like to stress, however, that most of our intuition about the problem was obtained by thinking in terms of PGs. Thinking in terms of MDPs seems harder, and we doubt whether we could have obtained our results by thinking directly in terms of linear programs.

In high level terms, our MDPs, and the linear programs corresponding to them, are constructions of 'fault tolerant' *randomized counters*. The challenge in designing such counters is making sure that they count 'correctly' under most sequences of random choices made by the RANDOM-FACET and RANDOM-EDGE pivoting rules. Our constructions are very different from the constructions used to obtain lower bounds for deterministic pivoting rules.

Significance. RANDOM-EDGE is perhaps the most natural randomized pivoting rule. RANDOM-FACET is the theoretically fastest pivoting rule currently known. Prior to this work there were no non-polynomial lower bounds on their performance. We show that both rules may require an expected subexponential number of iterations on some linear programs, resolving a major open problem.

The rest of this extended abstract is organized as follows. In Section 2 we give a brief introduction to *Markov Decision Processes* (MDPs) and the primal and dual linear programs corresponding to them. In Section 3 we review the policy iteration and the simplex algorithms, and the relation between improving switches and pivoting steps. In Section 4, which is the main section of this extended abstract, we describe our lower bound construction for RANDOM-EDGE. Many of the details are deferred, due to lack of space, to appendices. In Section 5 we briefly sketch our lower bound construction for RANDOM-FACET. (The results in this section rely heavily on our results from [FHZ11].) We end in Section 6 (and Section A) 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 widely used to model stochastic optimization problems in various areas ranging from operations research, machine learning, artificial intelligence, economics and game theory. The study of MDPs started with the seminal work of Bellman [Bel57]. (More general *stochastic games* were previously considered by Shapley [Sha53].) For a thorough treatment of MDPs, see the books of Howard [How60], Derman [Der72], Puterman [Put94] and Bertsekas [Ber01].

An MDP is composed of a finite set of *states*. Each state has a set of *actions* associated with it. Each action has an immediate *reward* and a probability distribution according to which the next

state of the process is determined if this action is taken. In each time unit, the *controller* of the MDP has to choose an action associated with the current state of the process. The goal of the controller is to maximize the long-term (infinite horizon) *expected reward per turn* of the actions taken.

A *policy* for the controller of an MDP is a rule that specifies which action should be taken in each situation. The decision may depend on the current state of the process and possibly on the *history*. A policy is *positional* if it is deterministic and history independent. A policy is *optimal* if it maximizes the expected cost per turn. One of the fundamental results concerning MDPs (see, e.g., [Put94]), says that every infinite horizon MDP has an optimal positional policy. Furthermore, there is always a single positional policy which is optimal from every starting state. All policies considered here are positional.

Formally, an MDP is defined by specifying its *underlying graph* $G = (V_0, V_R, E_0, E_R, r, p)$. Here, V_0 is the set of vertices (states) controlled 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*. (See Figure 3 on page 17 for a small example.)

A policy σ is a function $\sigma : V_0 \to E_0$ that selects for each vertex $u \in V_0$ and edge $\sigma(u) = (u, v) \in E_0$. (We assume that each vertex $u \in V_0$ has at least one outgoing edge.) The *values* $\textsc{val}_\sigma(u)$ and *potentials* $\textsc{pot}_\sigma(u)$ of the vertices under σ are defined as the unique solutions of the following set of linear equations:

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

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

together with the condition that $\textsc{pot}_\sigma(u)$ sum up to 0 on each irreducible recurrent class of the Markov chain defined by σ. All MDPs considered in this paper satisfy the *unichain* condition (see [Put94]) that states that the Markov chain obtained from each policy σ has a single irreducible recurrent class. This condition implies, in particular, that all vertices have the same value. It is not difficult to check that $\textsc{val}_\sigma(u)$ is indeed the expected reward per turn, when the process starts at u and policy σ is used. The potentials $\textsc{pot}_\sigma(u)$ represent *biases*. Loosely speaking, the expected reward after N steps, when starting at u and following σ, and when N is sufficiently large, is about $N\, \textsc{val}_\sigma(u) + \textsc{pot}_\sigma(u)$.

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

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

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. (Due to possible degeneracies, the policy, i.e., basis, corresponding to a given bfs is not necessarily unique. If for some $u \in V_0$ we have $x(u,v) = 0$ for every $(u,v) \in E_0$, then the choice of $\sigma(u)$ is arbitrary.) It is well known that the policy corresponding to an optimal bfs of (P) is an optimal policy of the MDP. (See, e.g., [Put94].)

The dual linear program (for unichain MDPs) is:

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

If (y^*, z^*) is an optimal solution of (D), then z^* is the common value of all vertices, and $y^*(u)$, for every $u \in V_0$, is the potential 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., $z + y(u) - \sum_{w:(v,w)\in E_R} p(v,w)y(w) = r(u,v)$. Such a tight edge is guaranteed to exist.

3 Policy iteration algorithms and simplex algorithms

The most widely used algorithm for solving MDPs is Howard's [How60] *policy iteration* algorithm. The policy iteration algorithm is closely related to the simplex algorithm. It can, however, exploit the special structure of the LPs that correspond to MDPs and perform many pivoting steps simultaneously.

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 and potentials $\textsc{val}_{\sigma_i}(u)$ and $\textsc{pot}_{\sigma_i}(u)$ of all vertices. An edge $(u,v') \in E_0$, such that $(u,v') \neq \sigma_i(u)$ is then said to be an *improving switch* if and only if either $\textsc{val}_{\sigma_i}(v') > \textsc{val}_{\sigma_i}(u)$ or $\textsc{val}_{\sigma_i}(v') = \textsc{val}_{\sigma_i}(u)$ and $r(u,v') - \textsc{val}_{\sigma_i}(v') + \textsc{pot}_{\sigma_i}(v') > \textsc{pot}_{\sigma_i}(u)$.

It is well known (see, e.g., [How60], [Put94]) that σ is an optimal policy if and only if there are no improving switches with respect to it. Furthermore, if $(u_1, v_1'), \ldots, (u_k, v_k')$ is an arbitrary collection of improving switches with respect to σ, where $u_1, u_2, \ldots, u_k$ are distinct, and σ' is defined as $\sigma'(u_i) = (u_i, v_i')$, for $i = 1, 2, \ldots, k$, and $\sigma'(u) = \sigma(u)$, for all other vertices, then σ' is *strictly* better than σ, in the sense that for every $u \in V_0$ either $\textsc{val}_{\sigma'}(u) > \textsc{val}_{\sigma}(u)$ or $\textsc{val}_{\sigma'}(u) = \textsc{val}_{\sigma}(u)$ and $\textsc{pot}_{\sigma'}(u) \geq \textsc{pot}_{\sigma}(u)$, with a strict inequality for at least one vertex $u \in V_0$.

There is, in fact, a whole family of policy iteration algorithms, differing in the improving switches performed in each iteration. The most natural variant is, perhaps, the one in which the algorithm selects the best improving switch from each vertex and performs all these switches simultaneously. The worst-case complexity of this extremely natural algorithm, which is usually referred to as Howard's algorithm, was open for almost 50 years until it was shown by Fearnley [Fea10] that it may require a (sub-)exponential number of iterations. On the other hand, it was recently shown by Ye [Ye10] and Hansen *et al.* [HMZ11] that Howard's algorithm is strongly polynomial for *discounted* MDPs, and even discounted 2-player stochastic games, when the discount factor is fixed.

Policy iteration algorithms that perform a single switch at each iteration 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 $\sigma(u) \neq (u,v)$, then $x(u,v) = 0$. We can also view the values and potentials corresponding to σ as settings of the variables $y(u)$ and z of the dual linear program (D). (We again assume the unichain condition, so the values of all vertices is the same.) By linear programming duality, if $(y(u), z)$ is feasible then σ is an optimal

strategy. 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 $\sigma(u) = (u, v)$ by the edge (u, v') correspond 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. The simplex algorithm with the RANDOM-EDGE pivoting rule, when applied to the primal linear program of an MDP, is thus equivalent to the variant of the policy iteration algorithm in which the improving switch used in each iteration is chosen uniformly at random among all improving switches. This is the foundation of our lower bound for the RANDOM-EDGE rule. A similar observation holds for RANDOM-FACET.

4 Lower bound for RANDOM-EDGE

We start with a high-level description of the MDPs on which RANDOM-EDGE performs an expected subexponential number of iterations. The exact details are fairly intricate. Due to lack of space, some of the details, and most of the proofs, are deferred to appendices. As mentioned in the introduction, the construction may be seen as an implementation of 'fault tolerant' randomized counters.

A schematic description of the lower bound MDPs is given in Figure 1 (on page 16). The MDPs are described formally in Appendix B. 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. The shaded octagons enclosing some of the vertices stand for *cycle gadgets* shown in Figure 2 (on page 17). It is useful to assume, at first, these octagons stand for standard vertices. (When we adopt this point of view, we refer to $a_{i,j}$ simply as a_i, and similarly for $b_{i,j}$ and $c_{i,j}$.) We shall explain later why they need to be replaced by the cycle gadgets.

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 1-st, i-th and $(i+1)$-th levels are shown explicitly in the figure. Levels are separated by dashed lines. n, ℓ_i, h, g, for $1 \leq i \leq n$, are integer parameters for the construction. The MDP includes two *sources* r and s, and one *sink* t. The i-th level contains 7 vertices of V_0, namely, $a_i, b_i, c_i, d_i, u_i, w_i, x_i$, and two randomization vertices A_i and B_i. (When the cycle gadgets are used, a_i, b_i and c_i are replaced by collections of vertices $a_{i,j}$, $b_{i,j}$ and $c_{i,j}$.) We refer to the vertices a_i, b_i and c_i (and $a_{i,j}, b_{i,j}$ and $c_{i,j}$) as *cycle vertices*, and refer to the corresponding cycles as the A_i-, B_i- and C_i-cycles, respectively. The vertices u_i form the *right lane*, while the vertices w_i form the *left lane*. We use U and W to refer collectively to the vertices of the right and left lanes, respectively. In each of $a_{i,j}, b_{i,j}, c_{i,j}, u_i, w_i$, player 0, the controller, has two outgoing edges to choose from. Vertices d_i, x_i and y_i have only one outgoing edge, so no decision is made at them. (The role of d_i, x_i and y_i will become clear later.)

Most edges in Figure 1 have an immediate reward of 0 associated with them. (Such 0 rewards are not shown explicitly in the figure.) The only edges that have non-zero rewards associated with them are the edges $a_{i,j} \to x_i$, $b_{i,j} \to s$ and $c_{i,j} \to r$ that have reward $j\epsilon$, where ϵ is a sufficiently small number to be chosen later.

In addition to the rewards assigned to some of the edges, 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 $N = 3n + 1$ and $\varepsilon = N^{-(4n+8)}$. Priorities, if present, are listed next to the vertex name. (In particular, $\Omega(d_i) = 4i+5$, $\Omega(x_i) = 4i + 3$, $\Omega(y_i) = 4i + 6$, for $1 \leq i \leq n$, and $\Omega(r) = 6$. All other vertices have no priorities assigned to them.) Rewards and priorities are chosen such that priorities are always of higher importance. Note that it is desirable 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 has only two randomization vertices. From $\textsc{A}_i$, the edge $\textsc{A}_i \to a_i$ (or more specifically $\textsc{A}_i \to a_{i,\ell_i}$), is chosen with probability $1-\epsilon$, while the edge $\textsc{A}_i \to d_i$ is chosen with probability ϵ. Thus, if the $\textsc{A}_i$-cycle is *closed*, the MDP is guaranteed to eventually move to d_i. From $\textsc{B}_i$, each of the two edges $\textsc{B}_i \to b_i$ and $\textsc{B}_i \to c_i$ are chosen with probability $\frac{1-\epsilon}{2}$, while the edge $\textsc{B}_i \to y_i$ is chosen with probability ϵ. Again, if both $\textsc{B}_i$- and $\textsc{C}_i$-cycles are closed, an eventual transition to y_i is made. (This is similar to the use of randomization nodes by Fearnley [Fea10].)

To each state $\mathfrak{b} = (\mathfrak{b}_n, \ldots, \mathfrak{b}_1) \in \{0,1\}^n$ of an n-bit binary counter, we define a corresponding policy $\sigma_{\mathfrak{b}}$ of the MDP. If $\mathfrak{b}_i = 1$, then all three cycles in the i-th level are *closed*, and u_i and w_i point into the level, while if $\mathfrak{b}_i = 0$, then the three cycles are *open*, and u_i and w_i point to the next level. Our ultimate goal is to show that a run of RANDOM-EDGE, that starts with $\sigma_{0\ldots00}$, visits all 2^n policies $\sigma_{0\ldots00}, \sigma_{0\ldots01}, \sigma_{0\ldots10}, \ldots, \sigma_{1\ldots11}$, with high probability.

Our proof is conceptually divided into two parts. First we investigate the improving switches that can be performed from *well-behaved* policies of the MDP. This allows us to prove that there *exists* a sequence of improving switches that does indeed generate the sequence $\sigma_{0\ldots00}, \sigma_{0\ldots01}, \sigma_{0\ldots10}, \ldots, \sigma_{1\ldots11}$. This is true even if the cycle gadgets of Figure 2 are *not* used. A transition from $\sigma_{\mathfrak{b}}$ to $\sigma_{\mathfrak{b}+1}$ involves many improving switches. We partition the path leading from $\sigma_{\mathfrak{b}}$ to $\sigma_{\mathfrak{b}+1}$ into seven sub-paths which we refer to as *phases*. In the following we first give an informal description of the phases, and then describe how the cycle gadgets of Figure 2 increase the transition probabilities. Note that some of the mentioned improving switches exist during several phases. We present here the sequences of updates enforced by the gadgets with high probability. A more formal description of the phases is given in Section 4.1.

Let $\mathfrak{b}$ be a state of the bit-counter, and let the *least significant unset bit* be $k_{\mathfrak{b}} := \min(\{i \leq n \mid \mathfrak{b}_i = 0\} \cup \{n+1\})$. The phases are as follows:

1. At the beginning of the first phase the policy corresponds to $\sigma_{\mathfrak{b}}$, except that some of the $\textsc{C}_i$-cycles of unset bits are closed. It is improving, however, to open these cycles, since opening the cycle leads through r, which has priority 6, to the lowest set bit. If a $\textsc{C}_i$-cycle is closed it instead moves via the $\textsc{B}_i$-cycle to s to the lowest set bit. Hence, all $\textsc{C}_i$-cycles open during this phase.

2. The initial strategy for the second phase is exactly $\sigma_{\mathfrak{b}}$. It is desirable for all open $\textsc{B}_i$-cycles to close, because this implies moving via the $\textsc{C}_i$-cycles to r. The gadgets indicated by the octagons ensure that only the B-cycle of the least 0-bit $k_{\mathfrak{b}}$ is closed.

3. Since the $\textsc{B}_{k_{\mathfrak{b}}}$-cycle is now closed, the $\textsc{C}_{k_{\mathfrak{b}}}$-cycle at level $k_{\mathfrak{b}}$ also closes as this gives access to $y_{k_{\mathfrak{b}}}$, which has a large even priority.

4. Since the $\textsc{A}_{k_{\mathfrak{b}}}$-cycle has not yet closed there is (essentially) not access from $\textsc{A}_{k_{\mathfrak{b}}}$ to $d_{k_{\mathfrak{b}}}$. This implies that lower set bits are unable to reach the dominating even priority at $y_{k_{\mathfrak{b}}}$. In particular, the u_i vertices for $i \leq k_{\mathfrak{b}}$ are updated to provide access from the source s to $y_{k_{\mathfrak{b}}}$.

5. Next, all $\textsc{A}_i$- and $\textsc{B}_i$-cycles at levels $i < k_{\mathfrak{b}}$ open to reach $y_{k_{\mathfrak{b}}}$, and in particular to reach $y_{k_{\mathfrak{b}}}$ through a vertex x_i with as low a priority as possible. Note that it is also desirable for $\textsc{B}_i$-cycles of unset bits at higher levels to open (although they are currently already open). This property is critical for resetting the gadgets.

6. The $\textsc{A}_{k_{\mathfrak{b}}}$-cycle now closes since it is then able to avoid the odd priority at $x_{k_{\mathfrak{b}}}$.

7. Finally, since there is now access from $\textsc{A}_{k_{\mathfrak{b}}}$ to $d_{k_{\mathfrak{b}}}$ the w_i vertices for $i \leq k_{\mathfrak{b}}$ are updated accordingly, and after the phase is over it is again desirable to close lower $\textsc{B}_i$-cycles. Note also that lower $\textsc{C}_i$-cycles remained open.

Proving that a long sequence of switches exists is of course not enough. We need to prove that such a long sequence occurs with a sufficiently high probability. To do that we introduce the cycle gadgets of Figure 2.

The idea is to make the $\textsc{A}_i$-, $\textsc{B}_i$- and $\textsc{C}_i$-cycles longer such that they are difficult to close. The purpose of the small rewards on the edges is to make sure that only one edge at a time is an improving switch when closing a cycle. Hence, closing a cycle requires a very specific sequence of improving switches. Furthermore, we can use Chernoff bounds to bound the probability of a longer cycle closing before a shorter cycle. By increasing the length of the $\textsc{B}_i$-cycles for increasing i, we make sure that in phase 2 the $\textsc{B}_i$-cycle of the lowest unset bit closes first.

On the other hand, when opening a cycle all outgoing edges are simultaneously improving switches. This allows lower bits to reset very fast during phase 5 before the $\textsc{A}_i$ cycle closes in phase 6. The lengths of the cycles are determined in Section 4.2.

4.1 Counting phases

In this subsection, we formally describe the different *phases* that a strategy 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. We first introduce notation to succinctly describe strategies.

Note that all vertices of V_0 have at most binary out-degree. It will be convenient to describe the decision of a strategy in terms of $\{0,1\}$-values for all vertices of V_0 with binary out-degree. Let σ be a policy and $u \in \{u_i, w_i, a_{i,*}, b_{i,*}, c_{i,*} \mid i \in [n]\}$. We write:

$$\overline{\sigma}(u) = \begin{cases} 1 & \text{if } \sigma(u) = (u,v), \text{ such that } v \notin \{r,s\} \cup \{u_{i+1}, w_{i+1}, x_i \mid i \in [n]\} \\ 0 & \text{otherwise} \end{cases}$$

In other words, $\overline{\sigma}(u) = 1$ iff the node u moves *into* the corresponding level of the construction. We, furthermore, define the *total* number of edges of σ going into the respective cycles as:

$$\alpha_i(\sigma) = \sum_{j \in [h]} \overline{\sigma}(a_{i,j}) \qquad \beta_i(\sigma) = \sum_{j \in [\ell_i]} \overline{\sigma}(b_{i,j}) \qquad \gamma_i(\sigma) = \sum_{j \in [g]} \overline{\sigma}(c_{i,j})$$

We say that a cycle is:

1. *Closed*, if $\alpha_i(\sigma) = h$, $\beta_i(\sigma) = \ell_i$ or $\gamma_i(\sigma) = g$, respectively.
2. *Open*, if it is not closed.
3. *Completely open*, if $\alpha_i(\sigma) = 0$, $\beta_i(\sigma) = 0$ or $\gamma_i(\sigma) = 0$, respectively.
4. *Consecutive*, if the frontmost k vertices move into the cycle, for some k, and all remaining vertices move out of the cycle.

To describe the set of improving edges, we say that a cycle is:

1. *Opening*, if every unused edge moving out of the cycle is an improving switch.
2. *Closing*, if either the cycle is closed and there are no improving switches, or the cycle is consecutive and the only improving switch is $(a_{i,\alpha_i(\sigma)+1}, a_{i,\alpha_i(\sigma)})$, $(b_{i,\beta_i(\sigma)+1}, b_{i,\beta_i(\sigma)})$ or $(c_{i,\gamma_i(\sigma)+1}, c_{i,\gamma_i(\sigma)})$, respectively.

For every $i \in [n]$, we use a succinct notation tuple to provide all necessary information describing the i'th level: `b c a u w`, where `b` describes the B-cycle, `c` describes the C-cycle, `a` describes the A-cycle, `u` describes the right lane, and `w` describes the left lane.

The first three components, describing the cycles, take one of the following values:

1	cycle is *closed* and *closing*
0	cycle is *completely open* and *opening*

$\uparrow$	cycle is *open*, *consecutive* and *closing*
$\downarrow$	cycle is *opening*

The last two components describe the setting and improving switches of w_i and u_i. For concreteness we give the definitions for w_i, the definitions for u_i are similar.

1	$\overline{\sigma}(w_i) = 1$ and switching is no improvement
0	$\overline{\sigma}(w_i) = 0$ and switching is no improvement
$\searrow$	$\overline{\sigma}(w_i) = 1$ and switching is an improvement
$\nearrow$	$\overline{\sigma}(w_i) = 0$ and switching is an improvement

We write $*$ if we neither care about the current setting nor about any improving switches.
To describe the progress of reassembling the right and left lanes in phases 4 and 7, respectively, we define the index of the lowest level with an incorrect setting as follows:

$$\delta(\sigma, k) = \max\{i{\leq}k \mid i{<}k \iff \overline{\sigma}(u_i){=}1\} \qquad \eta(\sigma, k) = \max\{i{\leq}k \mid i{<}k \iff \overline{\sigma}(w_i){=}1\}$$

We are now ready to formulate the conditions for strategies that fulfill one of the seven phases along with the improving edges. See Table 1 for a complete description (with respect to a given strategy σ and global counter state $\mathfrak{b}$). Using a number of technical lemmas, given in Appendix B, we arrive at the following main lemma.

Lemma 4.1 *The improving switches from strategies that belong to the phases are exactly those specified in Table 1.*

Phase	$i > k_{\mathfrak{b}}$		$i = k_{\mathfrak{b}}$	$0 < i < k_{\mathfrak{b}}$	Side Conditions
	$\mathfrak{b}_i = 1$	$\mathfrak{b}_i = 0$			
1	1 1 1 1 1	$\uparrow\downarrow$ 0 0 0	$\uparrow\downarrow$ 0 0 0	1 1 1 1 1	$\mathtt{u}_2 = \begin{cases} \searrow & \text{if } i = \delta(\sigma, k_{\mathfrak{b}}) \\ 1 & \text{if } \delta(\sigma, k_{\mathfrak{b}}) > i \\ 0 & \text{otherwise} \end{cases}$
2	1 1 1 1 1	$\uparrow$ 0 0 0 0	$\uparrow$ 0 0 0 0	1 1 1 1 1	
3	1 1 1 1 1	$\uparrow$ 0 0 0 0	1 $\uparrow$ 0 0 0	1 1 1 1 1	
4	1 1 1 1 1	$\uparrow$ 0 0 0 0	1 1 $\uparrow$ $\mathtt{u}_1$ 0	1 1 1 $\mathtt{u}_2$ 1	
5	1 1 1 1 1	$\downarrow\uparrow$ 0 0 0	1 1 $\uparrow$ 1 $*$	$\downarrow$ 1 $\downarrow$ 0 $*$	$\mathtt{w}_2 = \begin{cases} \searrow & \text{if } i = \eta(\sigma, k_{\mathfrak{b}}) \\ 0 & \text{if } i > \eta(\sigma, k_{\mathfrak{b}}) \\ * & \text{otherwise} \end{cases}$
6	1 1 1 1 1	0 $\uparrow$ 0 0 0	1 1 $\uparrow$ 1 $*$	0 1 0 0 $*$	
7	1 1 1 1 1	0 $\uparrow$ 0 0 0	1 1 1 1 $\mathtt{w}_1$	$\mathfrak{b}$ 1 0 0 $\mathtt{w}_2$	

$$\mathtt{b} = \begin{cases} \uparrow & \text{if } i \geq \eta(\sigma, k_{\mathfrak{b}}) \\ 0 & \text{otherwise} \end{cases} \qquad \mathtt{u}_1 = \begin{cases} 1 & \text{if } k_{\mathfrak{b}} \neq \delta(\sigma, k_{\mathfrak{b}}) \\ \nearrow & \text{otherwise} \end{cases} \qquad \mathtt{w}_1 = \begin{cases} 1 & \text{if } k_{\mathfrak{b}} \neq \eta(\sigma, k_{\mathfrak{b}}) \\ \nearrow & \text{otherwise} \end{cases}$$

Table 1: Strategies and improving switches of the seven phases.

4.2 Transition probabilities

Let $\Sigma_{\mathfrak{b},p}$ be the set of policies that belong to phase p, where $p \in [7]$, with respect to a given setting $\mathfrak{b}$ of the counter. The sets $\Sigma_{\mathfrak{b},p}$ are defined by Table 1, where the improving switches from each such policy are also specified. Our goal in this section is to show that if Random-Edge is run on a policy from $\Sigma_{\mathfrak{b},p}$, then with an extremely high probability a policy from $\Sigma_{\mathfrak{b},p+1}$, or from $\Sigma_{\mathfrak{b}+1,1}$, if $p = 7$, is encountered after polynomially many steps. We show that the probability that this does not hold is $O(\mathrm{e}^{-n})$. The probability that one of the $7 \cdot 2^n$ phases fails is thus $O((2/\mathrm{e})^n)$, i.e., exponentially small.

Trans.	Competition	Noise Bounds		
		ξ^B	ξ^C	ξ^A
$1 \to 2$	$\Downarrow(\{\text{B}(i) \mid \mathfrak{b}_i{=}0\})$	$\nu(n) + \rho$	0	0
$2 \to 3$	$\Uparrow(\text{B}(k_\mathfrak{b}), \{\text{B}(i) \mid i{>}k_\mathfrak{b}, \mathfrak{b}_i{=}0\})$	$\nu(n) + \rho + \nu(\ell_{k_\mathfrak{b}})$	0	0
$3 \to 4$	$\Uparrow(\text{C}(k_\mathfrak{b}), \{\text{B}(i) \mid i{>}k_\mathfrak{b}, \mathfrak{b}_i{=}0\})$	$\nu(n){+}\rho{+}\nu(\ell_{k_\mathfrak{b}}){+}\nu(g)$	0	0
$4 \to 5$	$\Uparrow(\text{U}, \{\text{A}(k_\mathfrak{b})\}{\cup}\{\text{B}(i) \mid i{>}k_\mathfrak{b}, \mathfrak{b}_i{=}0\})$	$2\nu(n){+}\rho{+}\nu(\ell_{k_\mathfrak{b}}){+}\nu(g)$	0	$\nu(n)$
$5 \to 6$	$\Downarrow(\{\text{A}(k_\mathfrak{b})\}{\cup}\{\text{C}(i) \mid i{>}k_\mathfrak{b}, \mathfrak{b}_i{=}0\})$	0	ρ	$\nu(n){+}\rho$
$6 \to 7$	$\Uparrow(\text{A}(k_\mathfrak{b}), \{\text{C}(i) \mid i{>}k_\mathfrak{b}, \mathfrak{b}_i{=}0\})$	0	$\rho + \nu(h)$	0
$7 \to 1$	$\Uparrow(\text{W}, \{\text{C}(i) \mid i{>}k_\mathfrak{b}, \mathfrak{b}_i{=}0\}{\cup}\{\text{B}(i) \mid i{<}k_\mathfrak{b}\})$	$\nu(n)$	$\rho{+}\nu(h){+}\nu(n)$	0

Table 2: Noise bounds and phase transition competitions

The vertices of the MDP are partitioned into $3n$ cycles, which we refer to as the A_i-cycle, B_i-cycle and C_i-cycle, for $i \in [n]$, and the two lanes W and U. We use Z as a generic name for each one of these cycles or lanes. Table 1 specifies the behavior of each cycle or lane Z during a phase. A cycle Z is in one of the four states $1, 0, \uparrow, \downarrow$, as explained above. Recall that $\uparrow$ means that the cycle is closing, and that $\downarrow$ means that the cycle is opening. A lane Z is either fixed during a stage, or is being realigned.

A phase ends when a specified component Z completely opens, completely closes, or is completely realigned. By looking at Table 1 we see, for example, that phase 1 ends when all C_i-cycles, with $\mathfrak{b}_i = 0$, which are opening during the phase, open completely. Note that the B_i-cycles, with $\mathfrak{b}_i = 0$, are closing during the phase, and none of them is allowed to close completely before all the C_i-cycles open completely. Phase 1 *fails* only if one of the B_i-cycles closes completely before all required C_i-cycles open completely.

Similarly, in phase 2, all B_i-cycles with $\mathfrak{b}_i = 0$ are opening. The phase ends successfully if the first such cycle to close completely is the $\text{B}_{k_\mathfrak{b}}$-cycle (recall that $k_\mathfrak{b}$ is the index of the least significant unset bit). The phase thus fails only if some B_i-cycle, with $i > k_\mathfrak{b}$ and $\mathfrak{b}_i = 0$ closes completely before the $\text{B}_{k_\mathfrak{b}}$-cycle.

As a final example, note that phase 4 ends when the right lane U realigns, and that this should happen before the $\text{A}_{k_\mathfrak{b}}$-cycle and the B_i-cycles, with $i > k_\mathfrak{b}$ and $\mathfrak{b}_i = 0$, close completely.

We can thus view each phase as being composed of several simultaneous *competitions* between various components, some of which are trying to open while others are trying to close. In each phase, we either like all cycles that are trying to open, to open completely before any other cycle closes completely, as it is the case in phase 1. In other cases, we would like some specified cycle, like the $\text{B}_{k_\mathfrak{b}}$-cycle in phase 2, or the right lane U in phase 4, to completely close, or realign itself, before any other cycle closes completely.

The competitions carried out during each phase are shown in Table 2. We let $\Uparrow(\text{Z}, \mathcal{Z})$ denote a competition in which we want Z to completely close before any other component $\text{Z}' \in \mathcal{Z}$ closes completely. We let $\Downarrow(\mathcal{Z})$ denote the competition in which we want all cycles that are currently opening to open completely before any cycle $\text{Z} \in \mathcal{Z}$ closes completely. (Note that in $\Downarrow(\mathcal{Z})$ we do not specify which cycles are opening.)

Competitions between closing cycles and opening cycles are heavily biased towards the opening cycles. This is because a closing cycle has only one improving edge associated with it, while an opening cycle has, in general, many improving switches associated with it. As an improving switch is chosen uniformly at random, an improving switch that belongs to the opening cycle is much more likely to be selected.

In competitions between closing cycles, shorter cycles, or more precisely cycles with less 'missing' edges, clearly have an advantage. (Both cycles close at the same 'speed'.) To make it much more likely that the B_i-cycles that belong to less significant bits close before those corresponding to more

significant bits, we use longer B_i-cycles for the more significant bit positions. (The A_i-cycles and C_i-cycles, in contrast, are all of the same length.)

We rely on the following two simple probabilistic lemmas whose proofs are deferred to Appendix C.

Lemma 4.2 *Let a be the total length of all the cycles that are currently opening. Then, the probability that a closing cycle acquires at least b new edges before all opening cycles open completely is at most $\frac{a}{2^b}$.*

Lemma 4.3 *The probability that a closing cycle acquires b new edges before a different closing cycle of length a closes completely is at most $\mathrm{e}^{-\frac{1}{2}(b-a)^2/(b+a)}$.*

We let $\nu(a)$ be the value of b for which the probability $\mathrm{e}^{-\frac{1}{2}(b-a)^2/(b+a)}$ of Lemma 4.3 is at most e^{-n}. It is not difficult to check that $\nu(a) = a+n+\sqrt{n^2+4an}$. In particular, we have $\nu(n) = (2+\sqrt{5})n < 5n$, and $\nu(a) \leq a + 3\sqrt{an}$, for $a \geq 2n$. We also let $\rho = 2n$.

If Z is a cycle that is closing at a certain phase, but is not supposed to win the competition of the phase, we refer to the number of edges currently pointing into Z as the *noise level* of Z. To prove that competitions are won by the intended candidates, we prove that the probability that the noise level of any of the other cycles exceeds the noise bound specified on the right of Table 2, at any time during the phase, is exponentially small. Three different noise bounds ξ^B, ξ^C, ξ^A are specified for B_i-cycles, A_i-cycles and C_i-cycles, respectively. A phase ends successfully if the noise level of each cycle never reaches the length of that cycle.

It is not difficult to prove by induction that the probability that the noise level of a cycle exceeds the noise bound given in Table 2 is exponentially small. Let us look, for example, at the noise levels of the B_i-cycles. In phase 5, no B_i-cycle is closing, so $\xi^B = 0$ is a (vacuous) upper bound on the noise level of closing B_i-cycles. The same holds for phase 6. Some B_i-cycles are closing in phase 7. All these cycles, however, are completely open at the beginning of phase 7. The competition in phase 7 is with the left lane W. The realignment of a lane may be viewed as the closing of a cycle of length at most n. (Both lanes and cycles close one edge at a time.) Thus, by Lemma 4.3, the probability that the noise level of any of the B_i-counters exceeds $\nu(n)$ is at most e^{-n}. As mentioned $\nu(n) < 5n$. Phase 1 is a cycle opening competition. By Lemma 4.2, the probability that the noise level of a given B_i-cycle increases by more than $\rho = 2n$ is $O(n^4/2^{2n}) = o(\mathrm{e}^{-n})$. The other noise bound can be verified in a similar manner.

We are now in a position to choose the length of the various cycles. The length h of all the A_i-cycles should satisfy $h > \nu(n) + \rho$. This is satisfied by choosing $h = 8n$. The length g of the C_i-cycles should satisfy $g > \rho+\nu(8n)+\nu(n)$. As $\rho = 2n$, $\nu(8n) < 15n$ and $\nu(n) < 5n$, we can choose $g = 22n$. Finally, the length ℓ_{k+1} of the B_{k+1}-cycle should satisfy $\ell_{k+1} > 2\nu(n) + \rho + \nu(\ell_k) + \nu(22n)$. As $\nu(22n) < 33n$ and $\nu(\ell_k) \leq \ell_k + 3\sqrt{\ell_k n}$, it is enough to require that $\ell_{k+1} > \ell_k + 3\sqrt{\ell_k n} + 45n$. It is easy to check that this is satisfied by the choice $\ell_k = 25k^2n$. Putting everything together, we get

Theorem 4.4 *The expected number of improving switches performed by* RANDOM-EDGE *on the MDPs constructed in this section, which contain $\mathcal{O}(n^4)$ 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 the RANDOM-EDGE pivoting rule performs an expected subexponential number of iterations. It should be noted that *all* pivoting steps performed on these linear programs are *degenerate*, as the bfs corresponding to *any* policy has $x(t,t) = 1$, where t is the sink, while $x(u,v) = 0$, for any other edge $(u,v) \in E_0$. (Progress is still being made in each iteration as some potentials, i.e., dual variables, strictly increase.) It is not difficult to introduce small perturbations that would make all pivoting steps non-degenerate, without affecting the expected number of iterations. One way of doing it is to use a *discount factor* extremely close to 1.

5 Lower bound for Random-Facet

In [FHZ11], we recently produced a subexponential lower bound for the Random-Facet algorithm for parity games. By applying the same techniques that were used to convert the lower bound construction for the Random-Edge algorithm from parity games to MDPs, we show that the Random-Facet algorithm may also require subexponentially many steps to solve MDPs. Due to the connection between MDPs and LPs the same bound then follows for LPs. More precisely we prove the following theorem:

Theorem 5.1 *The worst-case expected running time of the* Random-Facet *algorithm for n-state MDPs is at least $2^{\Omega(\sqrt{n}/\log n)}$, even when at most $\mathcal{O}(\log n)$ actions are associated with each state.*

Corollary 5.2 *The worst-case expected running time of the* Random-Facet *algorithm for LPs of dimension n with $\mathcal{O}(n \log n)$ constraints is at least $2^{\Omega(\sqrt{n}/\log n)}$.*

The main observation required for the conversion is that, in the parity games of [FHZ11], the role of the second player, referred to as player 1, is very limited. A simplified transformation of a vertex b controlled by player 1 is shown in Figure 4. Suppose player 1 does not move left unless player 0 moves from both b_1 and b_2 to b. This behavior of player 1 can be simulated by a randomization vertex that moves left with very low, but positive probability. In addition to this modification we introduce vertices with large odd priority, similar to vertices d_i and x_i in the lower bound construction for the Random-Edge algorithm. The details of the construction can be found in Appendix E.

6 Concluding remarks and open problems

See Appendix A.

References

[AC78] D. Avis and V. Chvátal. Notes on Bland's pivoting rule. In *Polyhedral Combinatorics*, volume 8 of *Mathematical Programming Studies*, pages 24–34. Springer, 1978.

[AKS87] I. Adler, R.M. Karp, and R. Shamir. A simplex variant solving an m times d linear program in $O(\min(m^2, d^2))$ expected number of pivot steps. *Journal of Complexity*, 3(4):372–387, 1987.

[AM85] I. Adler and N. Megiddo. A simplex algorithm whose average number of steps is bounded between two quadratic functions of the smaller dimension. *Journal of the ACM*, 32(4):871–895, 1985.

[AZ96] N. Amenta and G.M. Ziegler. Deformed products and maximal shadows of polytopes. In *Advances in Discrete and Computational Geometry*, pages 57–90, Providence, 1996. Amer. Math. Soc.

[BDF+95] A.Z. Broder, M.E. Dyer, A.M. Frieze, P. Raghavan, and E. Upfal. The worst-case running time of the random simplex algorithm is exponential in the height. *Information Processing Letters*, 56(2):79–81, 1995.

[Bel57] R.E. Bellman. *Dynamic programming.* Princeton University Press, 1957.

[Ber01] D.P. Bertsekas. *Dynamic programming and optimal control.* Athena Scientific, second edition, 2001.

[Bor91] K.H. Borgwardt. Probabilistic analysis of the simplex method. In J.C. Lagarias and M.J. Todd, editors, *Mathematical Developments Arising from Linear Programming*, Contemporary Mathematics, pages 21–34. American Mathematical Society, 1991.

[BP07] J. Balogh and R. Pemantle. The Klee-Minty random edge chain moves with linear speed. *Random Structures & Algorithms*, 30(4):464–483, 2007.

[Chv83] V. Chvátal. *Linear programming.* A Series of Books in the Mathematical Sciences. W. H. Freeman and Company, New York, 1983.

[Dan63] G.B. Dantzig. *Linear programming and extensions.* Princeton University Press, 1963.

[Der72] C. Derman. *Finite state Markov decision processes.* Academic Press, 1972.

[EJ91] E.A. Emerson and C. Jutla. Tree automata, μ-calculus and determinacy. In *Proceedings of the 32nd FOCS*, pages 368–377. IEEE Computer Society Press, 1991.

[EJS93] E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model-checking for fragments of μ-calculus. In *International Conference on Computer-Aided Verification, CAV'93*, volume 697 of *LNCS*, pages 385–396, 1993.

[Fea10] J. Fearnley. Exponential lower bounds for policy iteration. In *Proc. of 37th ICALP*, pages 551–562, 2010.

[FHZ11] O. Friedmann, T.D. Hansen, and U. Zwick. A subexponential lower bound for the random facet algorithm for parity games. In *Proc. of 22nd SODA*, 2011. To appear, preliminary version available at `http://www.cs.au.dk/~tdh/papers/random_facet_lower_bound.pdf`.

[Fri09] O. Friedmann. An exponential lower bound for the parity game strategy improvement algorithm as we know it. In *Proc. of 24th LICS*, pages 145–156, 2009.

[Fri10] O. Friedmann. 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`.

[FS09] P. Flajolet and R. Sedgewick. *Analytic Combinatorics.* Cambridge University Press, 2009.

[FT86] Y. Fathi and C. Tovey. Affirmative action algorithms. *Math. Program.*, 34(3):292–301, 1986.

[Gär95] B. Gärtner. A subexponential algorithm for abstract optimization problems. *SIAM Journal on Computing*, 24:1018–1035, 1995.

[Gär02] B. Gärtner. The random-facet simplex algorithm on combinatorial cubes. *Random Structures & Algorithms*, 20(3):353–381, 2002.

[GHZ98] B. Gärtner, M. Henk, and G. Ziegler. Randomized simplex algorithms on Klee-Minty cubes. *Combinatorica*, 18(3):349–372, 1998.

[GK07] B. Gärtner and V. Kaibel. Two new bounds for the random-edge simplex-algorithm. *SIAM J. Discrete Math.*, 21(1):178–190, 2007.

[GLS88] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer Verlag, 1988.

[Gol95] M. Goldwasser. A survey of linear programming in randomized subexponential time. *SIGACT News*, 26(2):96–104, 1995.

[GS79] D. Goldfarb and W.Y. Sit. Worst case behavior of the steepest edge simplex method. *Discrete Applied Mathematics*, 1(4):277 – 285, 1979.

[GTW02] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games. A Guide to Current Research*, volume 2500 of *LNCS*. Springer, 2002.

[GTW+03] B. Gärtner, F. Tschirschnitz, E. Welzl, J. Solymosi, and P. Valtr. One line and n points. *Random Structures & Algorithms*, 23(4):453–471, 2003.

[HMZ11] T.D. Hansen, P.B. Miltersen, and U. Zwick. Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. In *Proc. of 2nd Symposium on Innovations in Computer Sciece (ICS)*, 2011. To appear.

[How60] R.A. Howard. *Dynamic programming and Markov processes*. MIT Press, 1960.

[Jer73] R. G. Jeroslow. The simplex algorithm with the pivot rule of maximizing criterion improvement. *Discrete Mathematics*, 4(4):367–377, 1973.

[Kal92] G. Kalai. A subexponential randomized simplex algorithm (extended abstract). In *Proc. of 24th STOC*, pages 475–482, 1992.

[Kal97] G. Kalai. Linear programming, the simplex algorithm and simple polytopes. *Mathematical Programming*, 79:217–233, 1997.

[Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[Kha79] L.G. Khachiyan. A polynomial time algorithm in linear programming. *Soviet Math. Dokl.*, 20:191–194, 1979.

[KK92] G. Kalai and D. Kleitman. Quasi-polynomial bounds for the diameter of graphs and polyhedra. *Bull. Amer. Math. Soc.*, 26:315–316, 1992.

[KM72] V. Klee and G. J. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities III*, pages 159–175. Academic Press, New York, 1972.

[KS06] J.A. Kelner and D.A. Spielman. A randomized polynomial-time simplex algorithm for linear programming. In *Proc. of 38th STOC*, pages 51–60, 2006.

[Mat94] J. Matoušek. Lower bounds for a subexponential optimization algorithm. *Random Structures and Algorithms*, 5(4):591–608, 1994.

[MC94] M. Melekopoglou and A. Condon. On the complexity of the policy improvement algorithm for Markov decision processes. *ORSA Journal on Computing*, 6(2):188–192, 1994.

[MG07] J. Matoušek and B. Gärtner. *Understanding and using linear programming*. Springer, 2007.

[MS99] Y. Mansour and S.P. Singh. On the complexity of policy iteration. In *Proc. of the 15th UAI*, pages 401–408, 1999.

[MS06] J. Matoušek and T. Szabó. RANDOM EDGE can be exponential on abstract cubes. *Advances in Mathematics*, 204(1):262–277, 2006.

[MSW96] J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4-5):498–516, 1996.

[MU05] M. Mitzenmacher and E. Upfal. *Probability and computing, Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[NN94] Y. Nesterov and A. Nemirovskii. *Interior Point Polynomial Methods in Convex Programming*. SIAM, 1994.

[Put94] M.L. Puterman. *Markov decision processes*. Wiley, 1994.

[San10] F. Santos. A counterexample to the hirsch conjecture. *CoRR*, abs/1006.2814v1, 2010.

[Sch86] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1986.

[Sch08] S. Schewe. An optimal strategy improvement algorithm for solving parity and payoff games. In *Proc. of 17th CSL*, pages 369–384, 2008.

[Sha53] L.S. Shapley. Stochastic games. *Proc. Nat. Acad. Sci. U.S.A.*, 39:1095–1100, 1953.

[Sma83] S. Smale. On the average number of steps of the simplex method of linear programming. *Mathematical Programming*, 27(3):241–262, 1983.

[ST04] D.A. Spielman and S. Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, 2004.

[Sti95] C. Stirling. Local model checking games. In *Proceedings of the 6th International Conference on Concurrency Theory (CONCUR'95)*, volume 962 of *LNCS*, pages 1–11. Springer, 1995.

[Tod86] M.J. Todd. Polynomial expected behavior of a pivoting algorithm for linear complementarity and linear programming problems. *Mathematical Programming*, 35(2):173–192, 1986.

[Ye97] Y. Ye. *Interior Point Algorithms: Theory and Analysis*. Wiley-Interscience, 1997.

[Ye10] Y. Ye. The simplex method is strongly polynomial for the Markov decision problem with a fixed discount rate. Available at `http://www.stanford.edu/~yyye/simplexmdp1.pdf`, 2010.

[Zad80] N. Zadeh. What is the worst case behavior of the simplex algorithm? Technical Report 27, Department of Operations Research, Stanford, 1980.

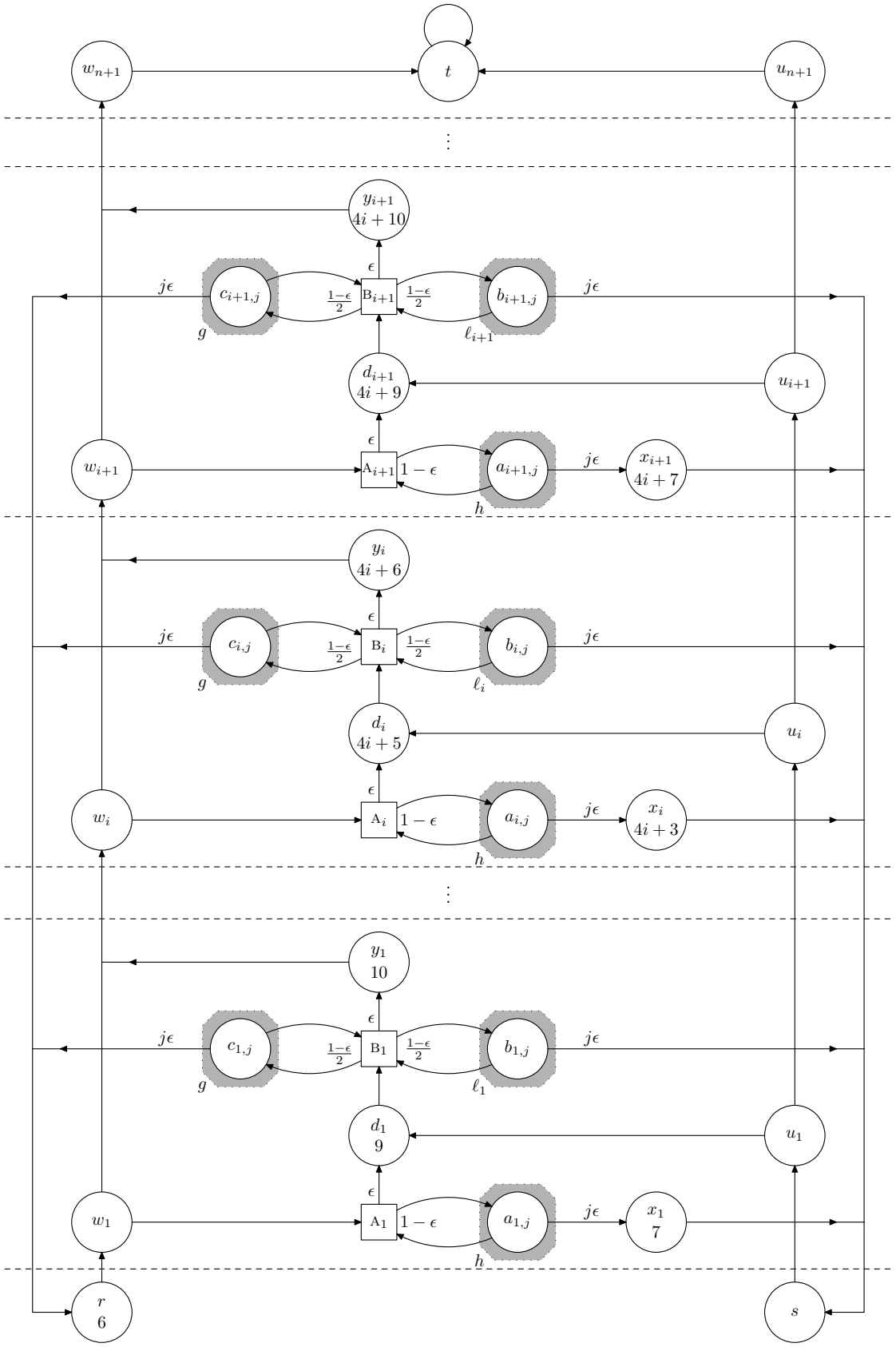

Figure 1: Random Edge MDP Construction. The interpretation of the shaded octagons is shown in Figure 2.

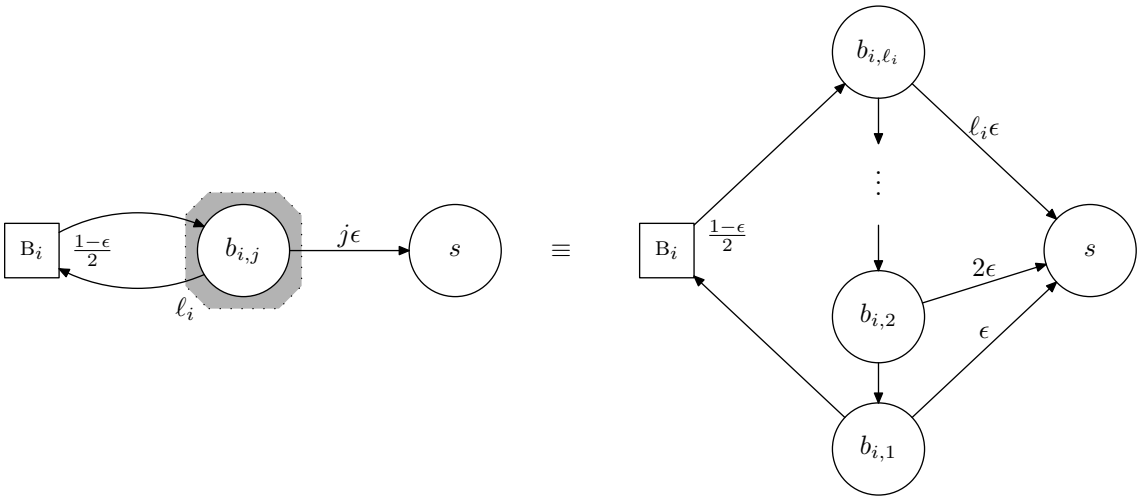

Figure 2: A cycle gadget used by the lower bound MDPs for Random-Edge.

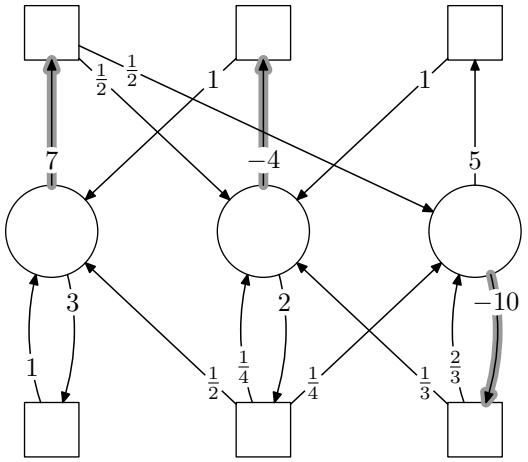

Figure 3: A simple MDP. Circles are controlled by player 0, and small squares are randomization vertices. The highlighted edges form a policy.

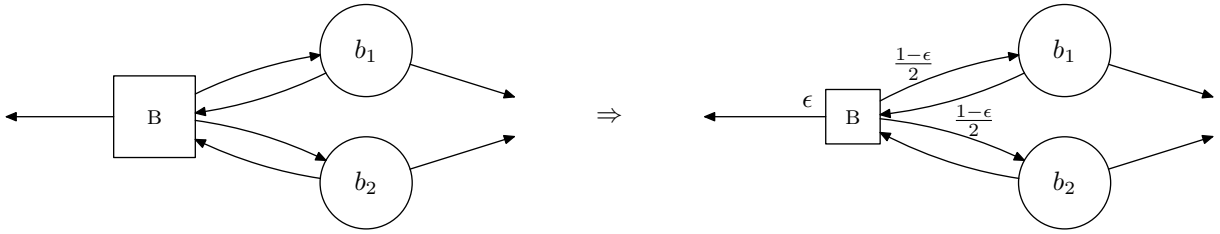

Figure 4: Conversion of a vertex controlled by player 1 to a randomization vertex.

A Concluding remarks and open problems

We have shown that both RANDOM-EDGE and RANDOM-FACET may lead to an expected subexponential number of iterations by constructing explicit linear programs with n variables on which the expected number of iterations performed by RANDOM-EDGE is $2^{\Omega(n^{1/4})}$, and (different) linear programs on which the expected number of iterations performed by RANDOM-FACET is $2^{\Omega(\sqrt{n}/\log^c n)}$, for some fixed $c > 0$.

Both lower bounds for linear programming have 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 results immediately transfer to mean payoff games, discounted payoff games and turned-based simple stochastic games [Fri10].

The analysis of the counter as well as the construction for RANDOM-EDGE can be improved greatly. First, the probabilistic analysis is based on the desire, that the binary counter operates without *any* faults, meaning that we really want to perform 2^n increment steps. However, counting "good enough", i.e. skipping a small number of increment steps from time to time, would still yield an exponential number of increment steps. Hence, our probabilistic analysis could be relaxed in such a way that the length of the cycles could be reduced. Additional details will appear in the final version of the paper.

Second, using a more complicated construction for parity games, we can decrease the lengths of the cycles to be linear in n, resulting in a $2^{\Omega(\sqrt{n})}$ lower bound. The main idea is to have downgoing edges from a node $b_{i,j}$ to nodes $b_{i',j}$ with $i' < i$; which of these downgoing edges is chosen will be controlled by player 1. The difference in the behaviour of RANDOM-EDGE on the improved construction is that when all B_i-cycles are competing with each other, higher B_i-cycles actually open all their nodes again that are already subsumed by the least open B_i-cycle. Hence, it is sufficient to have the same length for all B_i-cycles. The improved result is likely to transfer to MDPs and linear programs as well. Additional details will appear in the final version of the paper.

Our lower bound for the RANDOM-EDGE policy iteration for parity games and related two-player games can be extended to arbitrary *randomized multi-switch* improvement rules which select in each iteration step a subset with a certain cardinality of the improving switches arbitrarily at random. RANDOM-EDGE, for instance, always selects subsets with cardinality one, and the deterministic SWITCH-ALL rule always selects the subset with maximal cardinality. Another important randomized multi-switch improvement rule is SWITCH-HALF [MS99], which applies every improving switch with probability $1/2$. The lower bound transfers to all randomized multi-switch improvement rules due to the fact that the two kinds of competitions that we have in the analysis are won with even higher probability, when the cardinality of the number of switches that are to be made at the same time is greater than one.

We also mention without proof that the construction gives a subexponential lower bound for Schewe's globally "optimal" improvement rule [Sch08], although this result was previously known [Fri10].

A related open problem is to determine whether Zadeh's single-switch pivoting rule [Zad80] has a subexponential lower bound. If that is the case, the most convenient way to obtain a lower bound probably is to find a family of PGs again on which this pivoting rule requires a subexponential number of iterations.

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.

B Proofs and Constructions of Section 4

For a tuple $\zeta = (n, (\ell_i)_{0 \le i \le n}, h, g)$, with $n, \ell_i, h, g > 0$, define an underlying graph $G_\zeta = (V_0, V_R, E_0, E_R, r, p)$ of an MDP as shown schematically in Figure 1. More formally:

$$\begin{aligned} V_0 &:= \{a_{i,j} \mid i \in [n], j \in [h]\} \cup \{b_{i,j} \mid i \in [n], j \in [\ell_i]\} \cup \{c_{i,j} \mid i \in [n], j \in [g]\} \cup \\ &\quad \{d_i, y_i, x_i \mid i \in [n]\} \cup \{w_i, u_i \mid i \in [n+1]\} \cup \{t, r, s\} \\ V_R &:= \{\mathrm{A}_i, \mathrm{B}_i \mid i \in [n]\} \end{aligned}$$

With G_ζ, 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. $N \ge 3n+1$ and ε^{-1} to be significantly larger than the largest occurring priority induced reward, i.e. $\varepsilon \le N^{-(4n+8)}$. 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 3 defines the edge sets, the probabilities, the priorities and the immediate rewards of G_ζ.

Node	Successors	Probability
A_i	d_i	ε
	$a_{i,h}$	$1-\varepsilon$
B_i	y_i	ε
	b_{i,ℓ_i}	$\frac{1}{2} \cdot (1-\varepsilon)$
	$c_{i,g}$	$\frac{1}{2} \cdot (1-\varepsilon)$

Node	Successors	Priority
r	w_1	6
x_i	s	$4i+3$
d_i	B_i	$4i+5$
y_i	w_{i+1}	$4i+6$
w_{n+1}	t	-
u_{n+1}	t	-
w_i	w_{i+1}, A_i	-
u_i	u_{i+1}, d_i	-

Node	Successors	Cost
$a_{i,1}$	A_i	0
	x_i	1ε
$a_{i,j+1}$	$a_{i,j}$	0
	x_i	$(j+1)\varepsilon$
$b_{i,1}$	B_i	0
	s	1ε
$b_{i,j+1}$	$b_{i,j}$	0
	s	$(j+1)\varepsilon$
$c_{i,1}$	B_i	0
	r	1ε
$c_{i,j+1}$	$c_{i,j}$	0
	r	$(j+1)\varepsilon$
t	t	-
s	u_1	-

Table 3: Random Edge MDP Construction

Lemma B.1 *For every strategy σ, the MDP described by G_ζ ends in the sink t with probability 1.*

Proof: Let σ be a strategy. We write $v \rightsquigarrow v'$ to denote that the MDP conforming with σ starting in v reaches v' with positive probability. Note that $v \rightsquigarrow v'$ and $v' \rightsquigarrow v''$ implies $v \rightsquigarrow v''$.

We need to show that $v \rightsquigarrow t$ for every node v. Obviously, $t, w_{n+1}, u_{n+1} \rightsquigarrow t$.

First, it is easy to see by backwards induction on $i \le n$ that $\mathrm{A}_i \rightsquigarrow d_i \rightsquigarrow \mathrm{B}_i \rightsquigarrow y_i \rightsquigarrow w_{i+1} \rightsquigarrow t$, and hence also that $w_i, u_i \rightsquigarrow t$.

Second, it follows immediately that $r, s \rightsquigarrow t$, and hence also $x_i \rightsquigarrow t$. Finally, $a_i, b_i, c_i \rightsquigarrow t$.

Since all vertices reach t with positive probability, and t is an absorbing state, the statement of the lemma follows. □

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

Lemma B.2 *Let $P = \{r, y_i, x_i, d_i \mid i \le n\}$ be the set of nodes with priorities. For a subset $S \subseteq P$, let $\sum(S) = \sum_{v=S} \langle v \rangle$. For non-empty subsets $S \subseteq P$, let $v_S \in S$ be the node with the largest priority in S.*

1. $|\sum(S)| < N^{4n+8}$ *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* $S, S' \subseteq P$.

Lemma B.3 *Let* σ *be a strategy belonging to one of the phases specified in Table 1. Then* $|\textsc{pot}_\sigma(v)| < N^{4n+8}$ *and* $\varepsilon \cdot |\textsc{pot}_\sigma(v)| < 1$ *for every node* v.

Proof: Let $\mathfrak{b}$ be a global bit state, $k = k_\mathfrak{b}$ and σ be a strategy belonging to one of the phases with global bit state $\mathfrak{b}$. Let $\mathfrak{b}' = \mathfrak{b} + 1$. Let $\delta = \delta(\sigma, k)$, $\eta = \eta(\sigma, k)$, $S_i = \sum_{j \geq i,\, \mathfrak{b}_j = 1} (\langle d_j \rangle + \langle y_j \rangle)$, and $T_i = \sum_{j \geq i,\, \mathfrak{b}'_j = 1} (\langle d_j \rangle + \langle y_j \rangle)$.

It suffices to show that $|\textsc{pot}_\sigma(v)| < N^{4n+8}$ for every node v. Obviously, $\textsc{pot}_\sigma(t) = 0$.

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

$$\begin{aligned} \textsc{pot}_\sigma(s) &\in [S_1; T_1] & \textsc{pot}_\sigma(r) &\in [S_1; T_1] + \langle r \rangle \\ \textsc{pot}_\sigma(w_i) &\in [S_i; T_i] & \textsc{pot}_\sigma(u_i) &\in [S_i; T_i] \\ \textsc{pot}_\sigma(x_i) &\in [S_i; T_i] + \langle x_i \rangle & \textsc{pot}_\sigma(y_i) &\in [S_{i+1}; T_{i+1}] + \langle y_i \rangle \end{aligned}$$

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

$$\begin{aligned} \textsc{pot}_\sigma(\textsc{b}_i) &\in [S_1; T_{i+1} + \langle y_i \rangle] & \textsc{pot}_\sigma(d_i) &\in [S_1; T_{i+1} + \langle y_i \rangle] + \langle d_i \rangle \\ \textsc{pot}_\sigma(b_{i,*}) &\in [S_1; T_{i+1} + \langle y_i \rangle] & \textsc{pot}_\sigma(c_{i,*}) &\in [S_1; T_{i+1} + \langle y_i \rangle] \\ \textsc{pot}_\sigma(\textsc{a}_i) &\in [S_1 + \langle x_i \rangle; T_{i+1} + \langle y_i \rangle + \langle d_i \rangle] & \textsc{pot}_\sigma(a_{i,*}) &\in [S_1 + \langle x_i \rangle; T_{i+1} + \langle y_i \rangle + \langle d_i \rangle] \end{aligned}$$

By Lemma B.2, we have $|\textsc{pot}_\sigma(v)| < N^{4n+8}$ for every node v. □

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

The idea behind the cycles is to have a gate that controls the access of other nodes of the graph to the *escape node* of the cycle (d_i resp. y_i) to which the randomized node moves with very low probability.

First, assume that a cycle (or both cycles if there are two) is closed. 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 potential as the potential of the escape node itself.

Second, assume that a cycle is open, i.e. one of the V_0-controlled nodes of the cycle decides to move out of the cycle 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 potential of the randomized node essentially matches the potential of the reset node.

The critical property of cycles is that closing a cycle is a very slow process while opening proceeds at a rapid pace. Closing a cycle takes place when the potential of the escape node is better than the potential of the reset node. However, in every policy iteration step, there is only one improving edge associated with the cycle, namely the first edge pointing out of the cycle. Therefore, closing a cycle can only be performed one edge at a time. Opening a cycle happens in the reverse situation in which the potential of the reset node is better than the potential of the escape node. Here, every node that is currently moving into the cycle has an improving edge to move out of the cycle.

The following lemma formalizes the intuition of the behaviour of the cycles. If the escape node has better valuation than the reset nodes, it should be profitable to close the cycle, and otherwise, it should be profitable to open the cycle again. This idea generalizes to the setting in which two cycles are attached to the randomization node. Since both reset nodes necessarily have different

potentials, it is always the case that it is profitable to close one of the two cycles (the one with the worse reset node) and while it is closing, the other one is opening. If one of the two cycles is completely closed, the problem is essentially reduced to the case in which only one cycle is attached to the randomization node.

Lemma B.4 *Let σ be a strategy belonging to one of the phases specified in Table 1.*

1. $\text{POT}_\sigma(d_i) < \text{POT}_\sigma(x_i) \Rightarrow A_i$ *opening,*
2. $\text{POT}_\sigma(d_i) > \text{POT}_\sigma(x_i)$, A_i *consecutive, not closed* $\Rightarrow A_i$ *closing,*
3. $\text{POT}_\sigma(s) < \text{POT}_\sigma(r)$, B_i *consecutive, not closed* $\Rightarrow B_i$ *closing,* C_i *opening,*
4. $\text{POT}_\sigma(s) < \text{POT}_\sigma(r) < \text{POT}_\sigma(y_i)$, B_i *closed,* C_i *consecutive, not closed* $\Rightarrow C_i$ *closing, and*
5. $\text{POT}_\sigma(r) < \text{POT}_\sigma(y_i) < \text{POT}_\sigma(s)$, C_i *consecutive, not closed* $\Rightarrow C_i$ *closing,* B_i *opening.*

Proof: Let σ be a strategy belonging to one of the phases specified in Table 1.

1. Let $\text{POT}_\sigma(d_i) < \text{POT}_\sigma(x_i)$. We need to show that A_i is opening. We consider two cases.

   If A_i is closed, let $a_{i,j}$ be an arbitrary node on the cycle. It is easy to see that $\text{POT}_\sigma(()\text{A}_i) = \text{POT}_\sigma(d_i)$ and $\text{POT}_\sigma(()a_{i,j}) = \text{POT}_\sigma(d_i)$. It follows that $(a_{i,j}, x_i)$ is an improving edge for every j.

   If A_i is not closed, let $a_{i,j}$ be an arbitrary node on the cycle. Again, we consider two cases here.

   If $j > \overline{\gamma}_i(\sigma) + 1$ it follows that $a_{i,j}$ cannot reach the node A_i via the current strategy or via switching itself. Let $l < j$ be the largest l s.t. $\sigma(a_{i,l}) = 0$. It follows that $\text{POT}_\sigma(a_{i,j-1}) = \text{POT}_\sigma(x_i) + \varepsilon l$. Computing the difference of both choices x_i and $a_{i,j-1}$ shows that switching out of the cycle is profitable.

   $$\text{POT}_\sigma(x_i) + \varepsilon j - \text{POT}_\sigma(a_{i,j-1}) = \varepsilon(j - l) > 0$$

   If $j \leq \overline{\gamma}_i(\sigma) + 1$ it follows that $a_{i,j}$ can reach the node A_i via the current strategy or via switching itself. Assume w.l.o.g. that $j > 1$ (case $j = 1$ almost the same). Let l be the largest l s.t. $\sigma(a_{i,l}) = 0$. It follows that $\text{POT}_\sigma(a_{i,j-1}) = (1 - \varepsilon) \cdot (\text{POT}_\sigma(x_i) + \varepsilon l) + \varepsilon \text{POT}_\sigma(d_i)$. Computing the difference of both choices x_i and $a_{i,j-1}$ shows that switching out of the cycle is profitable.

   $$\text{POT}_\sigma(x_i) + \varepsilon j - \text{POT}_\sigma(a_{i,j-1}) = \varepsilon(j - (1 - \varepsilon)l + \text{POT}_\sigma(x_i) - \text{POT}_\sigma(d_i)) > 0$$

   since $\text{POT}_\sigma(x_i) - \text{POT}_\sigma(d_i) > h$.

2. Let $\text{POT}_\sigma(d_i) > \text{POT}_\sigma(x_i)$, A_i consecutive, not closed. We need to show that A_i is closing.

   Let $l = \overline{\gamma}_i(\sigma)$. It is not hard to see that the following holds for all $j \leq l$.

   $$\text{POT}_\sigma(a_{i,j}) = (1 - \varepsilon) \cdot (\text{POT}_\sigma(x_i) + \varepsilon h) + \varepsilon \text{POT}_\sigma(d_i)$$

   First, we compute the difference of both choices of $a_{i,j}$ for $j \leq l + 1$ to show that switching into the cycle is profitable. Assume w.l.o.g. that $j > 1$ (case $j = 1$ almost the same).

   $$\text{POT}_\sigma(x_i) + \varepsilon j - \text{POT}_\sigma(a_{i,j-1}) = \varepsilon(j - (1 - \varepsilon)h + \text{POT}_\sigma(x_i) - \text{POT}_\sigma(d_i)) < 0$$

   since $\text{POT}_\sigma(x_i) - \text{POT}_\sigma(d_i) < h$.

   Second, let $j > l + 1$. As before, it is easy to see that moving out of the cycle is profitable for node $a_{i,j}$.

The other statements 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 B.3 and Lemma B.4 to compute the potentials of all important nodes in the game to determine whether a successor of V_0-controlled node is improving or not.

We will use the notation $\mathcal{O}(1)$ to denote a small number ranging in $[-1; 1]$.

Lemma 4.1. Proof: Let $\mathfrak{b}$ be a global bit state, $k = k_{\mathfrak{b}}$ and σ be a strategy belonging to one of the phases with global bit state $\mathfrak{b}$. Let $\mathfrak{b}' = \mathfrak{b} + 1$. Let $\delta = \delta(\sigma, k)$, $\eta = \eta(\sigma, k)$, $S_i = \sum_{j \geq i,\, \mathfrak{b}_j = 1} (\langle d_j \rangle + \langle y_j \rangle)$, $S_i^l = \sum_{l \geq j \geq i,\, \mathfrak{b}_j = 1} (\langle d_j \rangle + \langle y_j \rangle)$, $T_i = \sum_{j \geq i,\, \mathfrak{b}'_j = 1} (\langle d_j \rangle + \langle y_j \rangle)$, and $T_i^l = \sum_{l \geq j \geq i,\, \mathfrak{b}'_j = 1} (\langle d_j \rangle + \langle y_j \rangle)$.

First, we apply Lemma B.2 and compute the potentials of all important nodes, see Table 4 for all the potentials. Second, we compute the differences between the potentials of two successors of a node to determine which edges are improving switches, see Table 5 for all the potential differences. Third, we derive from Table 5 that the improving switches w.r.t. w_i and u_i are exactly those specified in Table 1. Fourth, we apply Lemma B.4 to derive from Table 5 that the improving switches w.r.t. $b_{i,j}$, $c_{i,j}$, and $a_{i,j}$ are exactly those specified in Table 1. □

Phase	1–4	5–7
$\text{POT}_\sigma(s)$	S_1	T_1

Phase	1–4	5–7
$\text{POT}_\sigma(r)$	$S_1 + \langle r \rangle$	$[S_1; T_1 + \langle x_1 \rangle] + \langle r \rangle + \mathcal{O}(1)$

Phase	1–4	5–7
$\text{POT}_\sigma(x_i)$	$S_1 + \langle x_i \rangle$	$T_1 + \langle x_i \rangle$

Phase	1–3	4		5–7
		$i \leq \delta$	$i > \delta$	
$\text{POT}_\sigma(u_i)$	S_i		T_i	

Phase	1–4	5–6	7
		$i > k$	$i > \eta$
$\text{POT}_\sigma(w_i)$	S_i		T_i

Phase	1–4	5–6	
		$i \geq k$	$i < k$
$\text{POT}_\sigma(y_i)$	$S_{i+1} + \langle y_i \rangle$		$[S_{i+1}; T_1 + \langle x_{i+1} \rangle] + \langle y_i \rangle + \mathcal{O}(1)$

Phase	7			
	$i \geq \eta$	$i{+}1 = \eta = k$	$i{+}1 = \eta < k$	$i{+}1 < \eta$
$\text{POT}_\sigma(y_i)$	$T_{i+1} + \langle y_i \rangle$	$S_{i+1} + \langle y_i \rangle + \mathcal{O}(1)$	$T_1 + \langle x_{i+1} \rangle + \langle y_i \rangle + \mathcal{O}(1)$	$T_1 + [\langle x_\eta \rangle; \langle x_{i+1} \rangle] + \langle y_i \rangle + \mathcal{O}(1)$

Phase	1–4		5-6		7	
	$\mathfrak{b}_i{=}0$	$\mathfrak{b}_i{=}1$	$i{>}k, \mathfrak{b}_i{=}1$	$i{>}k, \mathfrak{b}_i{=}0$	$i{<}k \vee i{>}k, \mathfrak{b}_i{=}0$	$i{=}k \vee i{>}k, \mathfrak{b}_i{=}1$
$\text{POT}_\sigma(\text{A}_i)$	$S_1 + \langle x_i \rangle + \mathcal{O}(1)$	S_i	T_i	$T_1 + \langle x_i \rangle + \mathcal{O}(1)$		T_i

Phase	1–2	
	$\mathfrak{b}_i{=}0$	$\mathfrak{b}_i{=}1$
$\text{POT}_\sigma(d_i)$	$S_1 + \langle d_i \rangle + \frac{1}{2}\langle r \rangle + \mathcal{O}(1)$	S_i

Phase	3		
	$\mathfrak{b}_i{=}1$	$i{=}k$	$i{>}k, \mathfrak{b}_i{=}0$
$\text{POT}_\sigma(d_i)$	S_i	$S_1 + \langle d_i \rangle + \langle r \rangle + \mathcal{O}(1)$	$S_1 + \langle d_i \rangle + \frac{1}{2}\langle r \rangle + \mathcal{O}(1)$

Phase	4		
	$\mathfrak{b}_i{=}1$	$i{=}k$	$i{>}k, \mathfrak{b}_i{=}0$
$\text{POT}_\sigma(d_i)$	S_i	T_k	$S_1 + \frac{1}{2}\langle r \rangle + \langle d_i \rangle + \mathcal{O}(1)$

Phase	5–7		
	$i{=}k \vee i{>}k, \mathfrak{b}_i{=}1$	$i{>}k, \mathfrak{b}_i{=}0$	$i{<}k$
$\text{POT}_\sigma(d_i)$	T_i	$T_1 + \frac{1}{2}\langle r \rangle + \langle d_i \rangle + [S_1^k - \langle y_k \rangle - \langle d_k \rangle; \langle x_1 \rangle] + \mathcal{O}(1)$	$T_1 + \langle d_i \rangle + \mathcal{O}(1)$

Table 4: Potentials

Phase	1–4	5–7
$\textsc{pot}_\sigma(r) - \textsc{pot}_\sigma(s)$	$\langle r \rangle > 0$	$[S_1^k - \langle y_k \rangle - \langle d_k \rangle; \langle x_1 \rangle] + \langle r \rangle + \mathcal{O}(1) < 0$

Phase	1–4
$\textsc{pot}_\sigma(y_i) - \textsc{pot}_\sigma(r)$	$\langle y_i \rangle - \langle r \rangle - S_1^i > 0$

Phase	1–4	5–6	
		$i \geq k$	$i < k$
$\textsc{pot}_\sigma(y_i) - \textsc{pot}_\sigma(s)$	$\langle y_i \rangle - S_1^i > 0$	$\langle y_i \rangle - T_1^i > 0$	$[S_{i+1}^k - \langle y_k \rangle - \langle d_k \rangle; \langle x_{i+1} \rangle] + \langle y_i \rangle + \mathcal{O}(1) < 0$

Phase	7			
	$i \geq \eta$	$i{+}1 = \eta = k$	$i{+}1 = \eta < k$	$i{+}1 < \eta$
$\textsc{pot}_\sigma(y_i) - \textsc{pot}_\sigma(s)$	$\langle y_i \rangle - T_1^i > 0$	$\langle y_i \rangle - \langle y_k \rangle - \langle d_k \rangle + \mathcal{O}(1) < 0$	$\langle x_{i+1} \rangle + \langle y_i \rangle + \mathcal{O}(1) < 0$	$[\langle x_\eta \rangle; \langle x_{i+1} \rangle] + \langle y_i \rangle + \mathcal{O}(1) < 0$

Phase	1–4		5–7		7	
	$\mathfrak{b}_i{=}0$	$\mathfrak{b}_i{=}1$	$i{>}k, \mathfrak{b}_i{=}1$	$i{>}k, \mathfrak{b}_i{=}0$	$k{\geq}i{+}1{>}\eta$	$i{=}k$
$\textsc{pot}_\sigma(\textsc{a}_i) - \textsc{pot}_\sigma(w_{i+1})$	$S_1^i + \langle x_i \rangle + \mathcal{O}(1) < 0$	$\langle y_i \rangle + \langle d_i \rangle > 0$		$T_1^i + \langle x_i \rangle + \mathcal{O}(1) < 0$		$\langle y_k \rangle + \langle d_k \rangle > 0$

Phase	1–2		3		
	$\mathfrak{b}_i{=}0$	$\mathfrak{b}_i{=}1$	$\mathfrak{b}_i{=}1$	$i{=}k$	$i{>}k, \mathfrak{b}_i{=}0$
$\textsc{pot}_\sigma(d_i) - \textsc{pot}_\sigma(x_i)$	$\langle d_i \rangle - \langle x_i \rangle + \frac{1}{2}\langle r \rangle + \mathcal{O}(1) < 0$	$-S_1^{i-1} - \langle x_i \rangle > 0$		$\langle d_i \rangle - \langle x_i \rangle + \langle r \rangle + \mathcal{O}(1) < 0$	$\langle d_i \rangle - \langle x_i \rangle + \frac{1}{2}\langle r \rangle + \mathcal{O}(1) < 0$

Phase	4		
	$\mathfrak{b}_i{=}1$	$i{=}k$	$i{>}k, \mathfrak{b}_i{=}0$
$\textsc{pot}_\sigma(d_i) - \textsc{pot}_\sigma(x_i)$	$-S_1^{i-1} - \langle x_i \rangle > 0$	$\langle y_k \rangle + \langle d_k \rangle - \langle x_k \rangle > 0$	$\langle d_i \rangle - \langle x_i \rangle + \frac{1}{2}\langle r \rangle + \mathcal{O}(1) < 0$

Phase	5–7		
	$i{=}k \vee i{>}k, \mathfrak{b}_i{=}1$	$i{>}k, \mathfrak{b}_i{=}0$	$i{<}k$
$\textsc{pot}_\sigma(d_i) - \textsc{pot}_\sigma(x_i)$	$-T_1^{i-1} - \langle x_i \rangle > 0$	$\frac{1}{2}\langle r \rangle + \langle d_i \rangle - \langle x_i \rangle + [S_1^k - \langle y_k \rangle - \langle d_k \rangle; \langle x_1 \rangle] + \mathcal{O}(1) < 0$	$\langle d_i \rangle - \langle x_i \rangle + \mathcal{O}(1) < 0$

Phase	1–2		3		
	$\mathfrak{b}_i{=}0$	$\mathfrak{b}_i{=}1$	$\mathfrak{b}_i{=}1$	$i{=}k$	$i{>}k, \mathfrak{b}_i{=}0$
$\textsc{pot}_\sigma(d_i) - \textsc{pot}_\sigma(u_{i+1})$	$S_1^i + \langle d_i \rangle + \frac{1}{2}\langle r \rangle + \mathcal{O}(1) < 0$	$\langle y_i \rangle + \langle d_i \rangle > 0$		$S_1^i + \langle d_i \rangle + \langle r \rangle + \mathcal{O}(1) < 0$	$S_1^i + \langle d_i \rangle + \frac{1}{2}\langle r \rangle + \mathcal{O}(1) < 0$

Phase	4		
	$i{=}k \vee i < \delta \vee i{>}k, \mathfrak{b}_i{=}1$	$i{>}k, \mathfrak{b}_i{=}0$	$k{>}i{\geq}\delta$
$\textsc{pot}_\sigma(d_i) - \textsc{pot}_\sigma(u_{i+1})$	$\langle y_i \rangle + \langle d_i \rangle > 0$	$S_1^i + \langle d_i \rangle + \frac{1}{2}\langle r \rangle + \mathcal{O}(1) < 0$	$S_i^k - \langle y_k \rangle - \langle d_k \rangle < 0$

Phase	5–7		
	$i{=}k \vee i{>}k, \mathfrak{b}_i{=}1$	$i{>}k, \mathfrak{b}_i{=}0$	$i{<}k$
$\textsc{pot}_\sigma(d_i) - \textsc{pot}_\sigma(u_{i+1})$	$\langle y_i \rangle + \langle d_i \rangle > 0$	$T_1^i + \frac{1}{2}\langle r \rangle + \langle d_i \rangle + [S_1^k - \langle y_k \rangle - \langle d_k \rangle; \langle x_1 \rangle] + \mathcal{O}(1) < 0$	$T_1^i + \langle d_i \rangle + \mathcal{O}(1) < 0$

Table 5: Potential Differences

C Probabilistic lemmas

Lemma 4.2. Proof: Let $p(a, b)$ be the probability that the closing cycles acquire b new edges before the opening cycles, which currently have a edges pointing into them, open completely. We can ignore switches that do not belong to the opening cycles or the closing cycle. Thus, the probability that the next relevant edge chosen belongs to the opening cycles is $\frac{a}{a+1}$, while the probability that it belongs to the closing cycle is $\frac{1}{a+1}$. We thus get the following recurrence relation:

$$\begin{aligned} p(a,0) &= 1 \\ p(0,b) &= 0 \\ p(a,b) &= \frac{a}{a+1}p(a-1,b) + \frac{1}{a+1}p(a,b-1) \end{aligned}$$

We can now easily prove by induction that $p(a, b) \leq \frac{a}{2^b}$. For $a = 0$ or $b = 0$ the inequality clearly holds. Otherwise we have:

$$p(a,b) = \frac{a}{a+1}p(a-1,b) + \frac{1}{a+1}p(a,b-1) \leq \frac{a}{a+1}\frac{a-1}{2^b} + \frac{1}{a+1}\frac{a}{2^{b-1}} = \frac{a(a-1)+2a}{(a+1)2^b} = \frac{a}{2^b}$$

□

Lemma 4.3. Proof: As the probability of each of the two competing cycles to acquire a new edge is the same, we are essentially looking at the following 'experiment'. A fair coin is repeatedly tossed until either b heads or a tails are observed, for some $a < b$. We would like to bound the probability that b heads are observed before a heads.

The probability of getting b heads before a tails is exactly the probability of getting *less* than a tails in the first $a + b - 1$ tosses, which is at most the probability of getting *at most* a heads in the first $a + b$ tosses. The above probability can be easily bounded using the Chernoff bound. Let X be the number of heads observed in the first $a + b$ tosses. Then $\mu = E[X] = \frac{a+b}{2}$. The Chernoff bound, in the case of a fair coin (see Corollary 4.10 on page 71 of [MU05]), states that for every $0 < \delta < 1$ we have

$$\Pr[X \leq (1-\delta)\mu] \;\leq\; \mathrm{e}^{-\mu\delta^2} \;.$$

Let $\delta = \frac{b-a}{b+a}$. Then,

$$(1-\delta)\mu \;=\; a \quad , \quad \mu\delta^2 \;=\; \frac{(b-a)^2}{2(b+a)} \;,$$

and the claim of the lemma follows. □

D Parity games

In this section, we show how the random edge lower bound graphs can be turned into a parity game to provide a lower bound for random edge here 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 [FHZ11] and [Fri10].

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 1-*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 strategy for the whole game.

Theorem D.1 ([Fri10]) *Let G be a 1-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, while giving the nodes on the cycle odd priorities to make sure that moving into the cycle is only profitable by switching one edge at a time.

For a tuple $\zeta = (n, (\ell_i)_{0 \le i \le n}, h, g)$, with $n, \ell_i, h, g > 0$, we define the underlying graph $G_\zeta = (V_0, V_1, E, \Omega)$ of a parity game as shown schematically in Figure 5. More formally:

$$\begin{aligned} V_0 \;&:= \{a_{i,j} \mid i \in [n], j \in [h]\} \;\cup\; \{b_{i,j} \mid i \in [n], j \in [\ell_i]\} \;\cup\; \{c_{i,j} \mid i \in [n], j \in [g]\} \cup \\ &\quad \{d_i, y_i, x_i \mid i \in [n]\} \;\cup\; \{w_i, u_i \mid i \in [n+1]\} \;\cup\; \{t, r, s\} \\ V_1 \;&:= \{\mathrm{A}_i, \mathrm{B}_i \mid i \in [n]\} \end{aligned}$$

Table 6 defines the edge sets and the priorities of G_ζ.

Node V	Successors in E	Priority Ω
t	t	1
w_{n+1}	t	2
u_{n+1}	t	2
w_i	w_{i+1}, A_i	2
u_i	u_{i+1}, d_i	2
A_i	$d_i, a_{i,h}$	4
$a_{i,1}$	A_i, x_i	3
$a_{i,j+1}$	$a_{i,j}, x_i$	3
x_i	s	$4i+3$

Node V	Successors in E	Priority Ω
r	w_1	6
s	u_1	2
d_i	B_i	$4i+5$
y_i	w_{i+1}	$4i+6$
B_i	$y_i, b_{i,\ell_i}, c_{i,g}$	4
$b_{i,1}$	B_i, s	3
$b_{i,j+1}$	$b_{i,j}, s$	3
$c_{i,1}$	B_i, r	3
$c_{i,j+1}$	$c_{i,j}, r$	3

Table 6: Edges and Priorities of G_ζ

The first important observation to make is that the parity game is a 1-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 B.1 in the MDP world.

Lemma D.2 *Starting with an initial strategy σ s.t. $\overline{\sigma}(w_*) = \overline{\sigma}(s_*) = 0$, we have that G_ζ is a 1-sink parity game.*

All other definitions are exactly as in Section 4. Particularly, Table 1 becomes applicable again. The following lemma has the exact same formulation as Lemma 4.1 in the MDP world.

Lemma D.3 *The improving switches from strategies in the parity game that belong to the phases are exactly those specified in Table 1.*

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

All other proofs in Section 4 rely on Table 1 and Lemma 4.1, hence we transfer our main theorem to the parity game world.

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

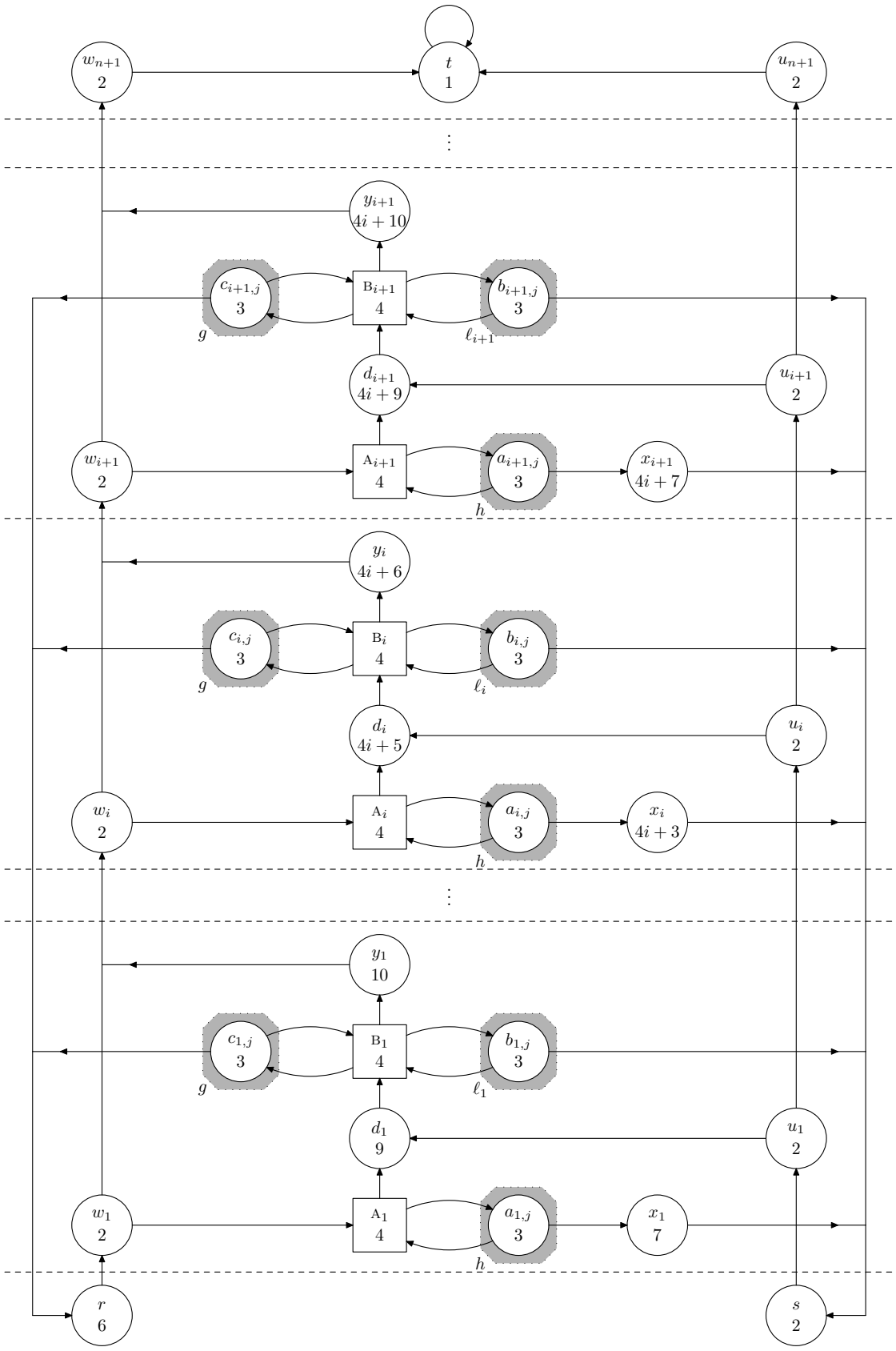

Figure 5: Lower bound parity game for Random-Edge.

```
Function RANDOM-FACET(G, σ)
  if E_0 = σ then
    return σ
  else
    Choose e ∈ E_0 \ σ uniformly at random
    σ' ← RANDOM-FACET(G \ {e}, σ)
    if e is an improving switch w.r.t. σ' then
      σ'' ← σ'[e]
      return RANDOM-FACET(G, σ'')
    else
      return σ'
```

```
Function RANDCOUNT(N)
  if N ≠ ∅ then
    Choose i ∈ N uniformly at random
    RANDCOUNT(N \ {i})
    for j ∈ N ∩ [i − 1] do  𝔟(j) ← 0
    𝔟(i) ← 1
    RANDCOUNT(N ∩ [i − 1])
```

Figure 6: The RANDOM-FACET algorithm (left), and a randomized bit-counter (right).

E Lower bound for the RANDOM-FACET algorithm for MDPs

The proof of Theorem D.4 presented here is closely tied to the corresponding proof in [FHZ11].

The RANDOM-FACET algorithm is shown in Figure 6. It works in a recursive manner, maintaining at all times a subset $F \subseteq E_0$ of the edges available to player 0, and a policy σ using only edges of F. The set of edges F defines a subgraph G_F that corresponds to a smaller MDP. We use the notation $\sigma[e]$ for the policy obtained from σ by performing the improving switch e.

The idea of the construction is to show that the RANDOM-FACET algorithm simulates a *randomized bit-counter* when run on an MDP from our family of lower bound MDPs. Unlike the construction for RANDOM-EDGE, the simulated counter is not a traditional binary counter, however. The randomized bit-counter is described in Figure 6. It counts recursively with a subset $N \subseteq [n]$ of the n bits. Intuitively, this corresponds to the behavior of the RANDOM-FACET algorithm.

Let $g(n)$ be the expected number of steps (recursive calls) performed by an n-bit randomized counter. It is easy to see that $g(0) = 1$ and that

$$g(n) = 1 + g(n-1) + \frac{1}{n}\sum_{i=0}^{n-1} g(i) \quad , \quad \text{for } n > 0.$$

In fact, $g(n) = 2^{\Omega(\sqrt{n})}$ (see, e.g., [FS09, p. 596-597]), and $g(n)$ is, thus, of the right subexponential form. We require a polylogarithmic number of vertices and edges to describe each bit, however.

E.1 Construction

For integers $n, g, h \geq 1$, we define a family of *lower bound MDPs* with underlying graphs $G_{n,g,h} = (V_0, V_R, E_0, E_R, \Omega, p)$ that the RANDOM-FACET algorithm requires many iterations to solve. n denotes the number of bits in the simulated randomized bit-counter, and g and h are parameters

later to be specified in the analysis. We use multi edges for convenience. A similar graph without multi edges can easily be defined by introducing additional vertices.

A graphical description of $G_{n,g,h}$ is given in Figure 7. Round vertices are controlled by player 0 and at square vertices the choice is made at random according to the probabilities on the outgoing edges. A shaded rectangle with label g indicates that the corresponding subgraph has been copied g times, as shown in Figure 8. Bold arrows are multi edges with multiplicity h. All rewards are described in terms of priorities, and only vertices x_i, y_i and d_i have priorities. Thus, most edges have reward zero.

Formally, $G_{n,g,h}$ is defined as follows.

$$\begin{aligned} V_0 &:= \{a_{i,j,k} \mid i \in [n], j \in [g], k \in [h]\} \cup \{b_{i,j} \mid i \in [n], j \in [gh]\} \cup \\ &\quad \{d_i, x_i, y_i \mid i \in [n]\} \cup \{u_i, w_i \mid i \in [n+1]\} \cup \{t\} \\ V_R &:= \{\mathrm{A}_{i,j} \mid i \in [n], j \in [g]\} \cup \{\mathrm{B}_i \mid i \in [n]\} \end{aligned}$$

Table 7 defines the edge set E, the priority assignment function Ω, multiplicities of edges, and the probability assignment function $p : E_R \to [0,1]$. $b_{i,*}$ is a shorthand for the set $\{b_{i,j} \mid j \in [gh]\}$. We again use $N = 3n+1$, $\varepsilon = N^{-(4n+8)}$ and $\langle v \rangle = (-N)^{\Omega(v)}$.

Node V_0	Successors in E_0	Priority Ω	Multiplicity
$a_{i,j,k}$	$\mathrm{A}_{i,j}$	-	1
	x_i		h
$b_{i,j}$	B_i	-	1
	u_{i+1}		h
d_i	B_i	$4i-1$	1
u_i	d_i	-	h
	u_{i+1}		h
u_{n+1}	t	-	1
w_i	$\mathrm{A}_{i,*}$	-	h
	w_{i+1}		h
w_{n+1}	t	-	1
x_i	u_i	$4i-3$	1
y_i	w_{i+1}	$4i$	1
t	t	-	1
Node V_R	**Successors in E_R**	**Priority Ω**	**Probability p**
$\mathrm{A}_{i,j}$	d_i	-	ε
	$a_{i,j,*}$		$\frac{1-\varepsilon}{h}$
B_i	y_i	-	ε
	$b_{i,*}$		$\frac{1-\varepsilon}{gh}$

Table 7: Priorities, edges, multiplicities, and transition probabilities of $G_{n,g,h}$.

The initial strategy σ given as input to the RANDOM-FACET algorithm is described by:

$$\begin{aligned} \forall i \in [n]\ \forall j \in [gh] &: \sigma(b_{i,j}) \neq \mathrm{B}_i \\ \forall i \in [n]\ \forall j \in [g]\ \forall k \in [h] &: \sigma(a_{i,j,k}) \neq \mathrm{A}_{i,j} \\ \forall i \in [n] &: \sigma(u_i) = u_{i+1} \\ \forall i \in [n] &: \sigma(w_i) = w_{i+1} \end{aligned}$$

Note that $G_{n,g,h}$ is a unichain. More specifically, we have the following lemma.

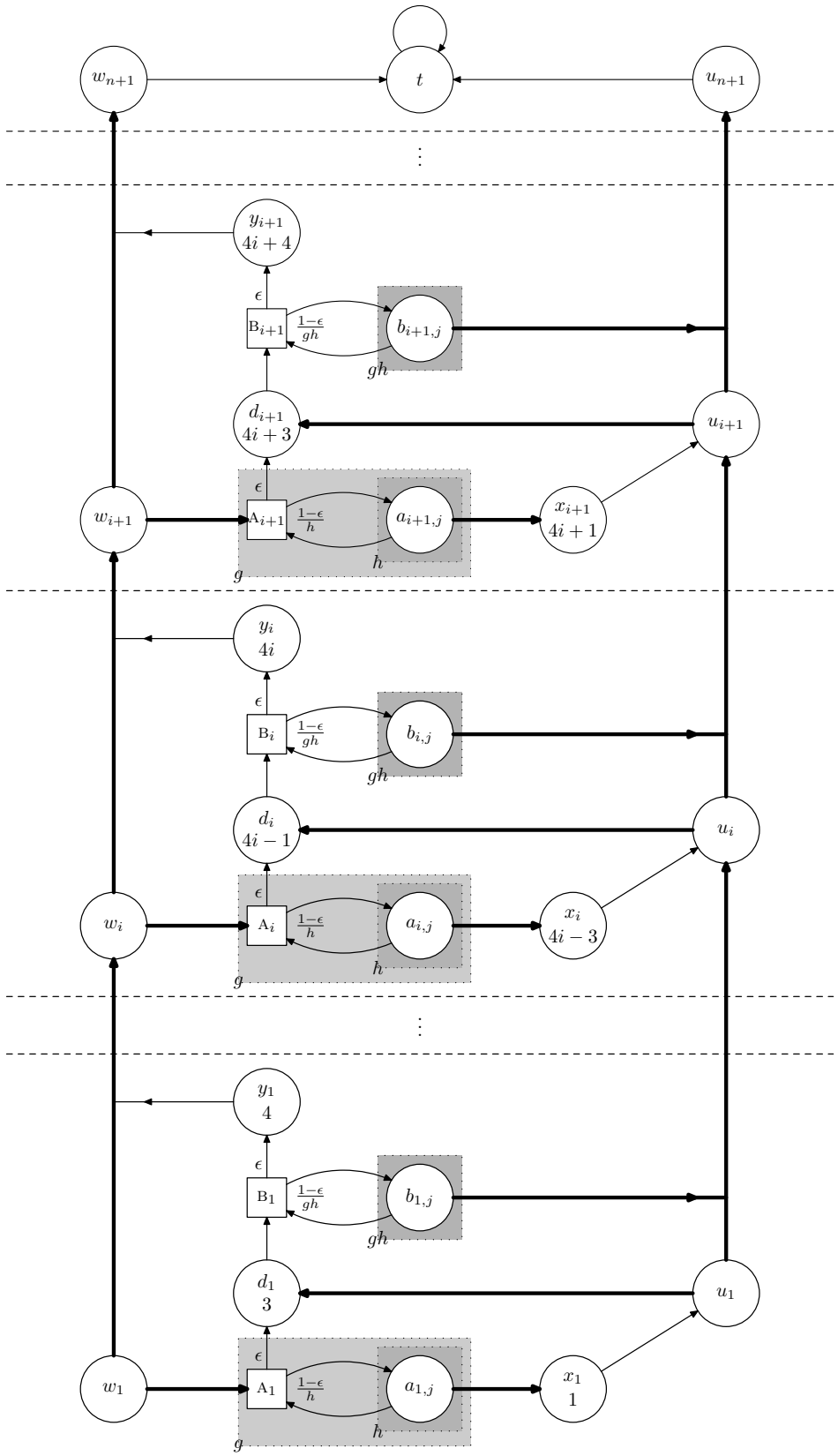

Figure 7: Lower bound MDP for the RANDOM-FACET algorithm.

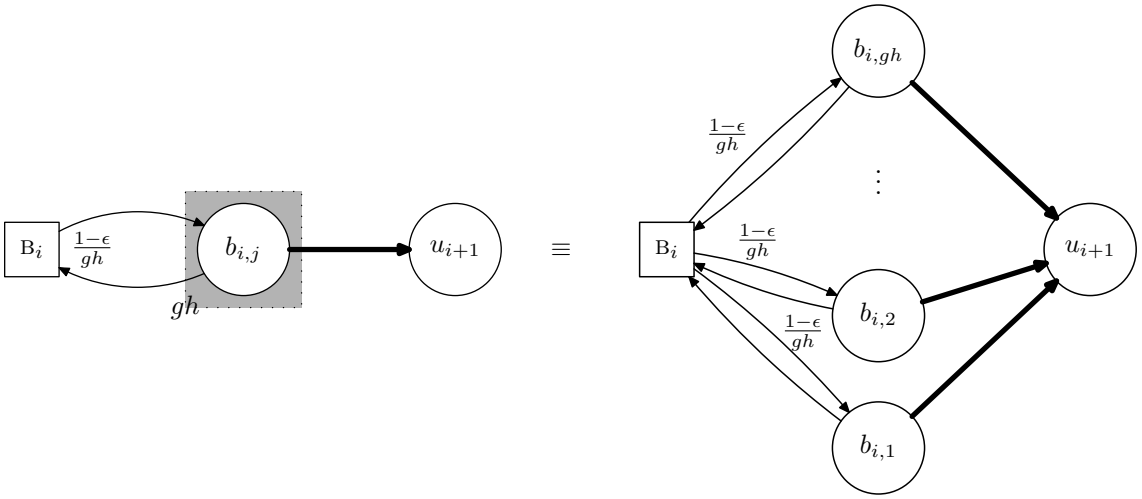

Figure 8: Duplication of a subgraph.

Lemma E.1 *For every strategy σ, the MDP with underlying graph $G_{n,g,h}$ ends in the sink t with probability* 1*.*

Proof: Let σ be a strategy. We write $v \rightsquigarrow v'$ to denote that the MDP conforming with σ starting in v reaches v' with positive probability. Note that $v \rightsquigarrow v'$ and $v' \rightsquigarrow v''$ implies $v \rightsquigarrow v''$.

We need to show that $v \rightsquigarrow t$ for every node v. Obviously, $t, w_{n+1}, u_{n+1} \rightsquigarrow t$.

First, it is easy to see by backwards induction on $i \leq n$ that $\mathrm{A}_i \rightsquigarrow d_i \rightsquigarrow \mathrm{B}_i \rightsquigarrow y_i \rightsquigarrow w_{i+1} \rightsquigarrow t$, and hence also that $x_i \rightsquigarrow u_i \rightsquigarrow t$. We then also have $w_i, a_i, b_i \rightsquigarrow t$.

Since all vertices reach t with positive probability, and t is an absorbing state, the statement of the lemma follows. □

E.2 Optimal strategies for MDPs corresponding to subgraphs

The Random-Facet algorithm operates with a subset of the edges controlled by player 0, $F \subseteq E_0$, such that the corresponding subgraph G_F is an MDP. We next introduce notation to concisely describe F. We say that F is *complete* if it contains at least one instance of every multi edge. We define the set of multi edges without multiplicities as:

$$\begin{aligned} M \;=\; & \{(a_{i,j,k}, x_i) \mid i \in [n], j \in [g], k \in [h]\} \;\cup \\ & \{(b_{i,j}, u_{i+1}) \mid i \in [n], j \in [gh]\} \;\cup \\ & \{(u_i, d_i) \mid i \in [n]\} \;\cup \\ & \{(u_i, u_{i+1}) \mid i \in [n]\} \;\cup \\ & \{(w_i, \mathrm{A}_{i,j}) \mid i \in [n], j \in [g]\} \;\cup \\ & \{(w_i, w_{i+1}) \mid i \in [n]\}. \end{aligned}$$

Furthermore, for $F \subseteq E_0$ define:

$$b_i(F) \in \begin{cases} 1 & \text{if } \forall j \in [gh] : (b_{i,j}, \mathrm{B}_i) \in F \\ 0 & \text{otherwise} \end{cases}$$

$$a_{i,j}(F) \in \begin{cases} 1 & \text{if } \forall k \in [h] : (a_{i,j,k}, \mathrm{A}_{i,j}) \in F \\ 0 & \text{otherwise} \end{cases}$$

$$a_i(F) \in \begin{cases} 1 & \text{if } \exists j \in [g] : a_{i,j}(F) = 1 \\ 0 & \text{otherwise} \end{cases}$$

That is, $b_i(F) = 1$ if and only if F contains every edge leading to B_i, and $a_{i,j}(F) = 1$ if and only if F contains every edge leading to $\text{A}_{i,j}$.

Finally, we define:

$$reset(F) = \max\left(\{0\} \cup \{i \in [n] \mid b_i(F) = 1 \wedge a_i(F) = 0\}\right)$$

to be the maximum index i for which $b_i(F) = 1$ and $a_i(F) = 0$. Intuitively, all bits with index lower than i will be reset when computing the optimal strategy in the subgraph G_F.

Let $F \subseteq E_0$ be a complete set. We say that a strategy $\sigma \subseteq F$ is *well-behaved* if for every $i \in [n]$, all copies $a_{i,j,k}$, for $j \in [\ell]$ and $k \in [r]$, and all copies $b_{i,j}$, for $j \in [\ell r]$, adopt corresponding choices, whenever possible. More formally, for every i, j_1, j_2, k_1, k_2, if $(a_{i,j_1,k_1}, \text{A}_{i,j_1}), (a_{i,j_2,k_2}, \text{A}_{i,j_2}) \in F$, then $\sigma(a_{i,j_1,k_1}) = \text{A}_{i,j_1}$ if and only if $\sigma(a_{i,j_2,k_2}) = \text{A}_{i,j_2}$, and similarly, for every i, j_1, j_2, if $(b_{i,j_1}, \text{B}_i), (b_{i,j_2}, \text{B}_i) \in F$, then $\sigma(b_{i,j_1}) = \text{B}_i$ if and only if $\sigma(b_{i,j_2}) = \text{B}_i$. If $(a_{i,j_1,k_1}, \text{A}_{i,j_1}), (a_{i,j_2,k_2}, \text{A}_{i,j_2}) \in F$, then the choices available at a_{i,j_1,k_1} are identical to the choices available at a_{i,j_2,k_2}, and the optimal choice must be the same. Hence, it follows that an optimal strategy of player 0 in G_F is well-behaved.

The essential behavior of a well-behaved policy σ is characterized by two Boolean vectors $\alpha(\sigma) = (\alpha_1, \ldots, \alpha_n)$ and $\beta(\sigma) = (\beta_1, \ldots, \beta_n)$ that are defined as follows:

$$\alpha_i(\sigma) \in \begin{cases} 1 & \text{if } \forall j \in [g]\ \forall k \in [h]: \\ & \quad (a_{i,j,k}, \text{A}_{i,j}) \in F \Rightarrow \sigma(a_{i,j,k}) = \text{A}_{i,j} \\ 0 & \text{if } \forall j \in [g]\ \forall k \in [h]: \sigma(a_{i,j,k}) \neq \text{A}_{i,j} \end{cases}$$

$$\beta_i(\sigma) \in \begin{cases} 1 & \text{if } \forall j \in [gh]: \\ & \quad (b_{i,j}, \text{B}_i) \in F \Rightarrow \sigma(b_{i,j}) = \text{B}_i \\ 0 & \text{if } \forall j \in [gh]: \sigma(b_{i,j}) \neq \text{B}_i \end{cases}$$

Similarly, given two Boolean vectors $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \ldots, \beta_n)$ we let $\sigma = \sigma(\alpha, \beta)$ be a well-behaved strategy such that $\alpha(\sigma) = \alpha$ and $\beta(\sigma) = \beta$. Note that $\sigma(\alpha, \beta)$ is not uniquely determined, as when $\alpha_i = 0$ or $\beta_i = 0$ we do not specify which copy of a multi-edge is chosen. This choice, however, is irrelevant.

The i'th bit of the randomized bit-counter is interpreted as being *set*, for some complete set F and a well-behaved strategy σ, if $a_i(F) = b_i(F) = 1$ and $\alpha_i(\sigma) = \beta_i(\sigma) = 1$.

Let σ^*_F be an optimal strategies for player 0 in the subgraph $G_F = (V_0, V_1, F \cup E_1, \Omega, p, s)$ defined by edges of F. Then σ^*_F is always well-behaved, and we let $\alpha^*(F) = \alpha(\sigma^*_F)$ and $\beta^*(F) = \beta(\sigma^*_F)$. The following lemma then describes the key parts of optimal strategies in the construction.

Again, we will use the notation $\mathcal{O}(1)$ to denote a small number ranging in $[-1; 1]$.

Lemma E.2 *Let $F \subseteq E_0$ be complete. Then σ^*_F is well-behaved and $\beta^*_i(F) = 1$ if and only if $i \geq reset(F)$, and $\alpha^*_i(F) = 1$ if and only if $b_i(F) = 1$ and $i \geq reset(F)$.*

Proof: First, recall that from Lemma E.1 we know that all vertices have the same value, namely $\text{VAL}_{\sigma_F}(v) = \text{VAL}_{\sigma_F}(t) = 0$, for all $v \in V$. Hence, we will focus only on potentials.

Note that except for cycling among vertices in the sets $\{\text{B}_i, b_{i,j} \mid j \in [gh]\}$ and $\{\text{A}_{i,j}, a_{i,j,k} \mid k \in [h]\}$, it is not possible to visit a vertex other than t twice. The idea of the proof is to describe σ^*_F using induction starting from t. To handle cycling we make use of the following two simple observations.

1. $\beta^*_i(F) = 1$ if and only if $\text{POT}_{\sigma^*_F}(y_i) > \text{POT}_{\sigma^*_F}(u_{i+1})$.
2. $\alpha^*_i(F) = 1$ if and only if $\text{POT}_{\sigma^*_F}(d_i) > \text{POT}_{\sigma^*_F}(x_i)$.

To verify the two observations note that edges involved in cycling have cost zero, and, hence, it is irrelevant how many times for instance $\textsc{B}_i$ is visited. Furthermore, it is optimal to increase the chance of ending at y_i if and only if $\textsc{pot}_{\sigma^*_F}(y_i) > \textsc{pot}_{\sigma^*_F}(u_{i+1})$.
We split the proof into four cases:

(i) $i > reset(F)$ and $b_i(F) = 1$.

(ii) $i > reset(F)$ and $b_i(F) = 0$.

(iii) $i = reset(F)$.

(iv) $i < reset(F)$

Cases (i), (ii) and (iii) are shown jointly by backward induction on i. For the induction hypothesis we assume that $\textsc{pot}_{\sigma^*_F}(w_{i+1}) = \textsc{pot}_{\sigma^*_F}(u_{i+1})$. This clearly holds true for $i = n$. It follows that

$$\textsc{pot}_{\sigma^*_F}(y_i) = \langle y_i \rangle + \textsc{pot}_{\sigma^*_F}(w_{i+1}) > \textsc{pot}_{\sigma^*_F}(u_{i+1}),$$

and, hence, by observation 1., $\beta^*_i(F) = 1$.
Case (i). Assume that $i > reset(F)$ and $b_i(F) = 1$. Then

$$\textsc{pot}_{\sigma^*_F}(d_i) = \langle d_i \rangle + \textsc{pot}_{\sigma^*_F}(\textsc{B}_i) = \langle d_i \rangle + \textsc{pot}_{\sigma^*_F}(y_i) = \langle d_i \rangle + \langle y_i \rangle + \textsc{pot}_{\sigma^*_F}(w_{i+1}) > \textsc{pot}_{\sigma^*_F}(u_{i+1})$$

Hence, the optimal choice at u_i is $\sigma^*_F(u_i) = d_i$, and furthermore

$$\textsc{pot}_{\sigma^*_F}(x_i) = \langle x_i \rangle + \textsc{pot}_{\sigma^*_F}(u_i) = \langle x_i \rangle + \textsc{pot}_{\sigma^*_F}(d_i) < \textsc{pot}_{\sigma^*_F}(d_i).$$

By observation 2., it then follows that $\alpha^*_i(F) = 1$.
Since $i > reset(F)$ we have $a_i(F) = 1$, and we get that there exists a $j \in [g]$ such that:

$$\textsc{pot}_{\sigma^*_F}(\textsc{A}_{i,j}) = \textsc{pot}_{\sigma^*_F}(d_i) = \langle d_i \rangle + \langle y_i \rangle + \textsc{pot}_{\sigma^*_F}(w_{i+1}) > \textsc{pot}_{\sigma^*_F}(w_{i+1}).$$

Hence, the optimal choice at w_i is $\sigma^*_F(w_i) = \textsc{A}_{i,j}$, and furthermore:

$$\textsc{pot}_{\sigma^*_F}(w_i) = \textsc{pot}_{\sigma^*_F}(d_i) = \textsc{pot}_{\sigma^*_F}(u_i)$$

which completes the induction step.
Case (ii). Assume that $i > reset(F)$ and $b_i(F) = 0$. Then

$$\textsc{pot}_{\sigma^*_F}(d_i) = \langle d_i \rangle + \textsc{pot}_{\sigma^*_F}(\textsc{B}_i) = \langle d_i \rangle + \textsc{pot}_{\sigma^*_F}(u_{i+1}) + \mathcal{O}(1) < \textsc{pot}_{\sigma^*_F}(u_{i+1})$$

Hence, the optimal choice at u_i is $\sigma^*_F(u_i) = u_{i+1}$, and

$$\textsc{pot}_{\sigma^*_F}(x_i) = \langle x_i \rangle + \textsc{pot}_{\sigma^*_F}(u_i) = \langle x_i \rangle + \textsc{pot}_{\sigma^*_F}(u_{i+1}) > \textsc{pot}_{\sigma^*_F}(d_i).$$

By observation 2., it then follows that $\alpha^*_i(F) = 0$, and, furthermore, for all $j \in [g]$

$$\textsc{pot}_{\sigma^*_F}(\textsc{A}_{i,j}) < \textsc{pot}_{\sigma^*_F}(x_i) < \textsc{pot}_{\sigma^*_F}(u_{i+1}) = \textsc{pot}_{\sigma^*_F}(w_{i+1}).$$

Hence, the optimal choice at w_i is $\sigma^*_F(w_i) = w_{i+1}$, and $\textsc{pot}_{\sigma^*_F}(w_i) = \textsc{pot}_{\sigma^*_F}(w_{i+1})$, which completes the induction step.
Case (iii). Assume that $i = reset(F)$. Then $b_i(F) = 1$ and $a_i(F) = 0$. The choices and potentials at vertices y_i, $\textsc{B}_i$, $b_{i,j}$, d_i, u_i, and x_i are exactly the same as in case (i), and in particular $\alpha^*_i(F) = 1$. Since $a_i(F) = 0$ we, however, get that for all $j \in [g]$:

$$\textsc{pot}_{\sigma^*_F}(\textsc{A}_{i,j}) = \textsc{pot}_{\sigma^*_F}(x_i) + \mathcal{O}(1) = \langle x_i \rangle + \langle d_i \rangle + \langle y_i \rangle + \textsc{pot}_{\sigma^*_F}(w_{i+1}) + \mathcal{O}(1) > \textsc{pot}_{\sigma^*_F}(w_{i+1})$$

Hence, the optimal choice at w_i is $\sigma_F^*(w_i) = \mathrm{A}_{i,j}$, where $j = \mathrm{argmax}_{j' \in [g]} \mathrm{POT}_{\sigma_F^*}(\mathrm{A}_{i,j'})$, and it follows that:

$$\mathrm{POT}_{\sigma_F^*}(w_i) = \langle x_i \rangle + \mathrm{POT}_{\sigma_F^*}(u_i) + \mathcal{O}(1). \tag{1}$$

Case (iv). Assume that $i < reset(F)$. Assume, furthermore, by induction that:

$$\mathrm{POT}_{\sigma_F^*}(w_{i+1}) = \langle x_{i+1} \rangle + \mathrm{POT}_{\sigma_F^*}(u_{i+1}) + \mathcal{O}(1).$$

The base case follows from equation (1) of case (iii).
We then have

$$\mathrm{POT}_{\sigma_F^*}(y_i) = \langle y_i \rangle + \mathrm{POT}_{\sigma_F^*}(w_{i+1}) < \mathrm{POT}_{\sigma_F^*}(u_{i+1}),$$

and, hence, by observation 1., $\beta_i^*(F) = 0$. Furthermore,

$$\mathrm{POT}_{\sigma_F^*}(d_i) = \langle d_i \rangle + \mathrm{POT}_{\sigma_F^*}(\mathrm{B}_i) = \langle d_i \rangle + \mathrm{POT}_{\sigma_F^*}(u_{i+1}) + \mathcal{O}(1) < \mathrm{POT}_{\sigma_F^*}(u_{i+1})$$

Thus, the optimal choice at u_i is $\sigma_F^*(u_i) = u_{i+1}$, and

$$\mathrm{POT}_{\sigma_F^*}(x_i) = \langle x_i \rangle + \mathrm{POT}_{\sigma_F^*}(u_i) = \langle x_i \rangle + \mathrm{POT}_{\sigma_F^*}(u_{i+1}) > \mathrm{POT}_{\sigma_F^*}(d_i).$$

By observation 2., it then follows that $\alpha_i^*(F) = 0$, and, furthermore, for all $j \in [g]$:

$$\mathrm{POT}_{\sigma_F^*}(\mathrm{A}_{i,j}) = \mathrm{POT}_{\sigma_F^*}(x_i) + \mathcal{O}(1) = \langle x_i \rangle + \mathrm{POT}_{\sigma_F^*}(u_i) + \mathcal{O}(1) =$$

$$\langle x_i \rangle + \mathrm{POT}_{\sigma_F^*}(u_{i+1}) + \mathcal{O}(1) > \mathrm{POT}_{\sigma_F^*}(w_{i+1}).$$

Finally, we then get that the optimal choice at w_i is $\sigma_F^*(w_i) = \mathrm{A}_{i,j}$, for some arbitrary $j \in [g]$, and it follows that:

$$\mathrm{POT}_{\sigma_F^*}(w_i) = \langle x_i \rangle + \mathrm{POT}_{\sigma_F^*}(u_i) + \mathcal{O}(1).$$

This completes the induction step and concludes the proof.

□

Lemma E.2 corresponds directly to Lemma 7.1 of [FHZ11], and the remainder of the proof is the same as in [FHZ11].