

# Lower Bounds for Ranking-Based Pivot Rules

Yann Disser  

TU Darmstadt, Germany


Georg Loho  

FU Berlin, Germany

University of Twente, Enschede, The Netherlands

Matthew Maat  

University of Twente, Enschede, The Netherlands

Nils Mosis  

TU Darmstadt, Germany

---

## Abstract

The existence of a polynomial pivot rule for the simplex method for linear programming, policy iteration for Markov decision processes, and strategy improvement for parity games each are prominent open problems in their respective fields. While numerous natural candidates for efficient rules have been eliminated, all existing lower bound constructions are tailored to individual or small sets of pivot rules. We introduce a unified framework for formalizing classes of rules according to the information about the input that they rely on. Within this framework, we show lower bounds for *ranking-based* classes of rules that base their decisions on orderings of the improving pivot steps induced by the underlying data. Our first result is a superpolynomial lower bound for strategy improvement, obtained via a family of sink parity games, which applies to memory-based generalizations of Bland’s rule that only access the input by comparing the ranks of improving edges in some global order. Our second result is a subexponential lower bound for policy iteration, obtained via a family of Markov decision processes, which applies to memoryless rules that only access the input by comparing improving actions according to their ranks in a global order, their reduced costs, and the associated improvements in objective value. Both results carry over to the simplex method for linear programming.

**2012 ACM Subject Classification** Mathematics of computing → Combinatorial algorithms

**Keywords and phrases** lower bounds, Markov decision processes, parity games, pivot rules, policy iteration, simplex method

**Digital Object Identifier** 10.4230/LIPIcs.STACS.2026.31

**Related Version** *Preprint of full version:* <https://arxiv.org/abs/2512.16684>

## 1 Introduction

The simplex method for linear programming is among the most well-studied algorithms in existence. To this day, the fundamental question of whether the method admits a (strongly) polynomial pivot rule remains one of the most important unsolved problems in mathematical optimization. A positive answer would address Smale’s 9th problem [51] and resolve the polynomial Hirsch conjecture [19, 48]. The question of the complexity of the simplex method also surfaced in other settings that are now known to reduce to linear programming: the complexity of strategy improvement for parity games [54] and of policy iteration for Markov decision processes [47].

Despite the importance of the question and despite numerous negative results for individual pivot rules [6, 7, 22, 31–33, 35, 38, 42, 46], a general lower bound *for all* rules is currently very much out of reach. Both, a polynomial bound on the diameter of polytopes and a systematic understanding of pivot rules, remain elusive. Hence, almost all results to date have been



© Yann Disser, Georg Loho, Matthew Maat, and Nils Mosis;

licensed under Creative Commons License CC-BY 4.0

43rd International Symposium on Theoretical Aspects of Computer Science (STACS 2026).

Editors: Meena Mahajan, Florin Manea, Annabelle McIver, and Nguyễn Kim Thăng

Article No. 31; pp. 31:1–31:19



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



tailored to single rules with little hope of generalization. Exceptions include a recent bound for shadow vertex rules and steepest edge rules [12], as well as a bound for combinations of three classical rules [23]. While simultaneously handling a continuum of pivot rules, these latter results are still rather interpolations between pivot rules than conceptual classes that capture a meaningful subset of all rules.

We propose a classification of pivot rules according to the information they use, and provide general lower bounds for a natural family of rules. We hope that our treatment can be the basis for a systematic study of general classes of pivot rules in the future.

**Our results.** We introduce a formal framework for defining classes of pivot rules in terms of the information on which they base their decisions. We focus on *ranking-based* classes of rules that rely only on orderings of the improving neighbors in each step according to some underlying data (e.g., objective value, global (Bland) index, rate of improvement, etc.), without otherwise using the underlying data. This is a natural way of capturing “combinatorial” pivot rules and encompasses all classical deterministic rules.

Our first result is a superpolynomial lower bound on the running time of pivot rules that only use the relative ordering of improving edges/variables according to their global indices. This generalizes Bland’s rule, which always selects the improvement of lowest global index, to rules that use the local ranks of improving edges/variables in more elaborate ways. In particular, our result extends beyond memoryless rules. For example, it applies to an adaptation of Bland’s rule that alternates between selecting the first and the second improving edge/variable.

► **Theorem 1.1 (Informal).** *For every deterministic pivot rule that uses  $o(N/\log(N))$  memory states and bases decisions solely on the ordering of improving edges/variables by global index, the strategy improvement and the simplex algorithm take  $N^{\omega(1)}$  iterations in the worst-case, where  $N$  is the input size.*

The granted memory is in some sense not far from optimal, since, for every given game, there exists a pivot rule with  $N$  memory states that takes linear time (Observation 4.1). We prove this theorem by constructing a family of sink parity games and applying a direct connection between the strategy improvement algorithm for parity games and the simplex algorithm for linear programming. This continues a fruitful line of work originating in lower bounds for policy iteration for Markov decision processes and parity games that were transferred to LPs in different ways [7, 22, 23, 26, 29, 31, 32, 44, 45]. A key idea in these constructions is to model a binary counter by a game and use gadgets to force the algorithm to increase the counter bit by bit.

For our second result, we devise an adversarial construction for the policy iteration algorithm for Markov decision processes, and again use a well-known connection to the simplex algorithm for linear programming [30]. This allows us to prove a subexponential (i.e. of the form  $2^{\Theta(n^c)}$  for some  $0 < c < 1$ ) lower bound on pivot rules that simultaneously use rankings by global index (such as Bland’s rule), by reduced cost (such as Dantzig’s original rule), and by objective improvement (such as the largest-increase rule). As before, our result applies to rules using combinations of these orderings in elaborate ways.

► **Theorem 1.2 (Informal).** *For every deterministic and memoryless pivot rule that bases decisions solely on orderings of improving actions/variables by global index, reduced cost, and objective improvement, the policy iteration and the simplex algorithm take  $\Omega(2^{\sqrt{N}})$  iterations in the worst-case, where  $N$  is the input size.*

Conceptually, this shows that the inefficiency of Bland’s rule, Dantzig’s rule, and the largest-increase rule arises not merely from their greedy nature but already from the fact that the information they rely on is too limited.

**Related work.** On the positive side, regarding the performance of the simplex method, variants of the shadow vertex pivot rule provide polynomial simplex algorithms in average-case [15] and smoothed [17, 37, 52] analysis. Further, linear programs can be solved in weakly-polynomial time via ellipsoid [41] and interior-point [40] methods. An adaptation of the latter is strongly polynomial if the constraint matrix satisfies some sparsity condition [3, 18], while barrier methods cannot be strongly polynomial [4].

On the negative side, the simplex algorithm is NP-mighty [25], and it is PSPACE-complete to predict the behavior of certain pivot rules [1, 27]. Further, it is NP-hard to approximate the length of a shortest monotone path to the optimum [16, 21]. Recently, it was proven that there is no polynomial pivot rule for the active-set method [24], which is a natural generalization of the simplex method to non-linear objectives, even if a convex quadratic is to be maximized [8]. The proof of the latter uses *deformed products*, which were introduced in [5] to unify existing lower bound constructions for classical simplex pivot rules.

Interestingly, every pivot rule can be formulated as a *normalized-weight* pivot rule if the associated normalization may depend on the input data [13]. Further, every polyhedron allows for a similar polyhedron with small diameter [39] such that a strongly polynomial simplex algorithm for polyhedra of small diameter would already resolve Smale’s problem. In the context of tropical linear programming [10], it was shown that every strongly polynomial, *combinatorial* simplex pivot rule would provide a strongly polynomial algorithm for mean payoff games [2]. Unique sink orientations are another combinatorial abstraction of linear programs [34, 50, 53].

## 2 Preliminaries

The simplex method solves linear programs (LPs) of the form

$$\begin{aligned} \max \quad & \mathbf{c}^\top \mathbf{x}, \\ \text{s.t.} \quad & A\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{LP}_{A,\mathbf{b},\mathbf{c}}$$

where  $A \in \mathbb{Q}^{m \times n}$  has full row-rank,  $\mathbf{b} \in \mathbb{Q}^m$ , and  $\mathbf{c} \in \mathbb{Q}^n$ . A *basis*  $B = \{b_1, \dots, b_m\} \subseteq [n]$ , where we write  $[k] := \{1, 2, \dots, k\}$ , for all  $k \in \mathbb{N}$ , is a set of indices of  $m$  linearly independent columns of  $A$ . We denote the (regular) matrix consisting of these columns by  $A_B$ . Given a basis  $B$ , there is a unique  $\mathbf{x} \in \mathbb{R}^n$  such that  $A\mathbf{x} = \mathbf{b}$  and  $x_i = 0$  for all  $i \in [n] \setminus B$ . If this  $\mathbf{x}$  is feasible for  $(LP_{A,\mathbf{b},\mathbf{c}})$ , i.e.,  $x_j \geq 0$  for all  $j \in B$ , we call it a *basic feasible solution* (BFS) with basis  $B$ , and  $B$  is a *feasible basis*. In this case,  $\mathbf{x}$  is a vertex of the feasible region  $\{\mathbf{x}: A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ .

In this paper, we assume (without loss of generality) that the feasible region is non-empty, that the linear program is bounded, i.e., the optimum is attained at a vertex, and that the simplex method gets an initial feasible basis  $B$  as input. Then, the algorithm selects an *entering* index  $i \in [n] \setminus B$  and a *leaving* index  $j \in B$  such that

- $\text{rc}(i) := c_i - \mathbf{c}_B^\top A_B^{-1} \mathbf{A}_{\cdot i} > 0$ ,
- $B' := B \cup \{i\} \setminus \{j\}$  is a feasible basis,

and repeats with  $B \leftarrow B'$ , until reaching the optimum. We call  $rc(i)$  the *reduced costs* of index  $i$  (or variable  $x_i$ ) and say that  $i$  (or  $x_i$ ) is an *improving index* (or *improving variable*) for  $B$  if  $rc(i) > 0$ . We write  $\Gamma^+(B) := \{i \in [n] \setminus B : rc(i) > 0\}$  to denote the set of all improving indices for  $B$ . Since the choices of  $i$  and  $j$  are in general not unique, *pivot rules* are used to determine the behavior of the algorithm, see Section 3.

In this paper, we mainly consider *non-degenerate* linear programs, where we have a one-to-one correspondence between bases and vertices of the feasible region, such that, in each iteration of the simplex method, the choice of the leaving index is uniquely determined by the entering index.

**Sink parity games** are two-player games that are played on the vertices of a directed graph. A sink parity game (SPG) is given by a tuple  $(V_0, V_1, E, \pi)$ , specifying a directed graph  $G = (V_0 \cup V_1 \cup \{\top\}, E)$  and a *priority* function  $\pi : V_0 \cup V_1 \cup \{\top\} \rightarrow \mathbb{Z} \cup \{-\infty\}$ . We will often also refer to the sink parity game itself by  $G$ . We generally assume that each vertex of  $G$  has at least one outgoing edge. Further, the vertex  $\top$ , called the *sink*, has precisely one outgoing edge, which is a self-loop, and is the only vertex with priority  $\pi(\top) = -\infty$ .

The game is played between player 0 and player 1, where player  $i$  owns the vertices of  $V_i$ . At the start of the game, a token is placed on an initial vertex  $v_0$ . At step  $k$ , the owner of the current node  $v_k$  moves the token along an outgoing edge to a successor node  $v_{k+1}$ . The game continues infinitely, so the token travels along an infinite path  $(v_0, v_1, v_2, \dots)$ . The objective of player 0 is to maximize the outcome, given by  $\sum_{i=0}^{\infty} (-t)^{\pi(v_i)}$ , where  $t \geq |V_0 \cup V_1|$  is a constant. With this objective, the strategy improvement algorithm we will define later is equivalent to the version described in [54], see also [28] for background on parity games. Therefore, player 0 wants to collect large even priorities and avoid large odd priorities. Player 1 aims to minimize the outcome.

It can be shown that the optimal strategy for both players is positional, meaning they always make the same choice in the same node. We can thus consider a player  $i$  strategy as a function  $\sigma : V_i \rightarrow V_G$ , where  $V_G := V_0 \cup V_1 \cup \{\top\}$ . So, if player  $i$  commits to a strategy  $\sigma_i$ , the token can only move along the edges of  $E_{\sigma_i} := \{(v, \sigma_i(v)) : v \in V_i\} \cup \{(v, w) : v \in V_{1-i}\}$ .

Every sink parity game is assumed to admit

- a player 0 strategy  $\sigma_0$  such that the highest priority on any cycle in  $E_{\sigma_0}$  is even,
- a player 1 strategy  $\sigma_1$  such that the highest priority on any cycle in  $E_{\sigma_1}$  is odd.

We call the strategies from these assumptions *admissible* strategies. These two assumptions guarantee that, with optimal play, the token travels along a simple path towards  $\top$ , which guarantees that the outcome is finite for every given constant  $t$ . It also turns out that there always exists a strategy that is optimal for all starting vertices simultaneously [54]. Hence, *solving* the game usually means finding such an optimal strategy.

Let  $\sigma$  be an admissible player 0 strategy. Then, the optimal counterstrategy  $\bar{\sigma}$  for player 1 can easily be found by a shortest path computation. If the players play according to  $\sigma$  and  $\bar{\sigma}$ , the token will follow a simple path  $(v_0, v_1, \dots, v_k, \top)$  that ends in the sink. This defines a *valuation* function  $\text{Val}_\sigma : V_G \rightarrow \mathbb{N}^{\mathbb{N}}$ , where  $\text{Val}_\sigma(v_0)$  is given by the multiset  $\{\pi(v_0), \pi(v_1), \dots, \pi(v_k)\}$  for every  $v_0 \in V_G$ . We can compare these multisets by  $\triangleleft$ , where  $S_1 \triangleleft S_2$  if and only if  $\sum_{s \in S_1} (-|V_0 \cup V_1|)^s < \sum_{s \in S_2} (-|V_0 \cup V_1|)^s$ ; in this summation, elements occurring multiple times are used multiple times. We call an edge  $a = (v, w) \in E_0$  with  $\text{Val}_\sigma(w) \triangleright \text{Val}_\sigma(\sigma(v))$  an *improving move* or *improving switch* for  $\sigma$ . If we *apply* an improving switch  $a \in E_0$  to a strategy  $\sigma$ , we obtain a new strategy  $\sigma^a$  by  $\sigma^a(v) = w$  and  $\sigma^a(v') = \sigma(v')$  for all  $v' \in V_0 \setminus \{v\}$ . The improving switch  $a$  increases the value of  $v$  without decreasing the value of any other state. A strategy  $\sigma$  is optimal for a sink parity game if and only if there is no improving switch for  $\sigma$ .

**Markov decision processes** provide a framework for modeling the decision-making of an agent operating in an uncertain environment. The environment is described by a finite set  $S$  of *states* and a finite set  $A$  of *actions* representing the feasible decisions. Each action  $a \in A$  is associated with a *reward*  $\text{rew}(a) \in \mathbb{R}$  and a probability distribution  $P_a$  over  $S$ . In each state  $s \in S$ , the agent may choose among a non-empty subset  $A_s \subseteq A$  of *available* actions. When an action  $a \in A_s$  is chosen, the agent receives the reward  $\text{rew}(a)$ , and the process *transitions* to a state  $s' \in S$ , which is drawn according to  $P_a$ . For each  $s' \in S$ , we denote the probability that action  $a$  leads to  $s'$  with  $P_{a,s'}$  and call it a *transition probability*.

A state  $s$  is *reachable* from another state  $s'$  if there exists a sequence of actions such that the process transitions from  $s'$  to  $s$  with positive probability. A state  $\top \in S$  is called *sink* if it is reachable from all states and if  $A_\top = \{a\} \subseteq A$  with  $\text{rew}(a) = 0$  and  $P_{a,\top} = 1$ . Thus, once the agent reaches the sink, they cannot collect any further reward.

A *policy* is a function  $\sigma: S \rightarrow A$  assigning to each state  $s \in S$  an action  $\sigma(s) \in A_s$ . Given a policy  $\sigma$ , the *value* of each  $s \in S$ , denoted by  $\text{Val}_\sigma(s)$ , is the expected total reward obtained when the process starts in  $s$  and the agent follows  $\sigma$  in every state. In other words, the *value* function  $\text{Val}_\sigma: S \rightarrow \mathbb{R}$  is determined by the following system of Bellman [9] equations

$$\text{Val}_\sigma(s) = \text{rew}(\sigma(s)) + \sum_{s' \in S} P_{\sigma(s),s'} \text{Val}_\sigma(s'), \quad \forall s \in S, \quad (1)$$

together with  $\text{Val}_\sigma(\top) = 0$  if there is a sink  $\top$ . A policy  $\sigma$  is *optimal* (with respect to the *expected total reward criterion*) if  $\text{Val}_\sigma(s) \geq \text{Val}_{\sigma'}(s)$  for all  $s \in S$  and all policies  $\sigma'$ . See [47] for a discussion on other optimality criteria that consider discounted or average rewards.

A policy  $\sigma$  is *weak unichain* if there is a sink  $\top$ , and the agent will finally reach  $\top$  with probability one when following  $\sigma$  in every state, irrespective of the starting vertex. We call the process weak unichain if it admits an optimal policy that is weak unichain. We refer to [30] for more details on the weak unichain condition.

An action  $a \in A_s$  is an *improving switch* for policy  $\sigma$  if

$$\text{rc}_\sigma(a) := \text{rew}(a) + \sum_{s' \in S} P_{a,s'} \text{Val}_\sigma(s') - \text{Val}_\sigma(s) > 0,$$

where  $\text{rc}_\sigma(a)$  is called *reduced costs* of  $a$  (with respect to  $\sigma$ ).

If we *apply* an improving switch  $a \in A_s$  to a policy  $\sigma$ , we obtain a new policy  $\sigma^a$  by  $\sigma^a(s) = a$  and  $\sigma^a(s') = \sigma(s')$  for all  $s' \in S \setminus \{s\}$ . The improving switch  $a$  increases the value of  $s$  without decreasing the value of any other state. A policy  $\sigma$  is optimal for a weak unichain Markov decision process if and only if there is no improving switch for  $\sigma$ .

**Strategy improvement** or **policy iteration** is an algorithmic idea that can be used to find optimal strategies and values for many types of games on graphs, see, e.g., [28]. To avoid confusion, we use the name STRATEGYIMPROVEMENT when it is applied to sink parity games, and the name POLICYITERATION when applied to weak unichain Markov decision processes. The idea is simple: given an initial policy/strategy, iteratively apply improving switches until reaching an optimal solution, see Algorithm 1.

In this paper, we only consider a version of this algorithm that applies a single switch in each iteration. For a more general discussion, we refer to [47, 54]. The algorithm will only visit admissible strategies/weak unichain policies, and is guaranteed to return the optimal strategy/policy after a finite number of iterations. Whenever there are multiple improving switches, the choice in the algorithm is determined by an *improvement rule*, analogously to pivot rules for the simplex method.

■ **Algorithm 1** POLICYITERATION or STRATEGYIMPROVEMENT.

---

**input:**  $\sigma$  weak unichain policy (MDP) or admissible strategy (SPG)

---

**while**  $\sigma$  admits an improving switch **do**

- $a \leftarrow$  an improving switch for  $\sigma$
- $\sigma \leftarrow \sigma^a$

**return**  $\sigma$

---

It is well-known that, for every weak unichain Markov decision process  $\mathcal{M}$ , there is a linear program  $\text{LP}_{\mathcal{M}}$  such that the application of the simplex method to  $\text{LP}_{\mathcal{M}}$  is equivalent to the application of POLICYITERATION to  $\mathcal{M}$ , in the following sense.

- **Theorem 2.1** ([30, Sec. 4.5]). *Let  $\mathcal{M}$  be a weak unichain Markov decision process. Then, there exists a non-degenerate, bounded linear program  $\text{LP}_{\mathcal{M}}$  of the form  $(\text{LP}_{A,\mathbf{b},\mathbf{c}})$  such that*
- *there is a bijection between variables of  $\text{LP}_{\mathcal{M}}$  and actions of  $\mathcal{M}$ ,*
  - *there is a bijection between basic feasible solutions of  $\text{LP}_{\mathcal{M}}$  and weak unichain policies for  $\mathcal{M}$ , where variables in the basis correspond to actions used in the policy,*
  - *given a basic feasible solution  $\mathbf{x}$  of  $\text{LP}_{\mathcal{M}}$  and the corresponding weak unichain policy  $\sigma$  for  $\mathcal{M}$ , the reduced cost of each variable equals the reduced cost of the associated action, and the objective value of  $\mathbf{x}$  equals  $\mathbf{c}^\top \mathbf{x} = \sum_{s \in S} \text{Val}_\sigma(s)$ .*

Recently, [44] provided an analogous reduction for sink parity games, based on [49].

- **Theorem 2.2** ([44, Thm. 3.4, Sec. 3.1]). *Let  $G$  be a sink parity game in which every vertex has a unique priority. Then, there exists a non-degenerate, bounded linear program  $\text{LP}_G$  of the form  $(\text{LP}_{A,\mathbf{b},\mathbf{c}})$  such that*
- *there is a bijection between the variables of  $\text{LP}_G$  and the edges in  $(V_0 \times V_G) \cup \{(\top, \top)\}$ ,*
  - *there is a bijection between the admissible strategies for  $G$  and the basic feasible solutions of  $\text{LP}_G$ , where the variables in the basis correspond to the edges used in the strategy,*
  - *given a basic feasible solution  $\mathbf{x}$  of  $\text{LP}_G$  and the associated admissible strategy  $\sigma$  for  $G$ , a variable is improving at  $\mathbf{x}$  if and only if the corresponding edge is improving for  $\sigma$ .*

Conveniently, it is very simple to transform a sink parity game into an equivalent sink parity game with unique priorities by repeating the following: whenever there are two nodes with priority  $p$ , we can keep one of them at priority  $p$  and increase the priority of all other nodes with priority at least  $p$  by 2. This has no effect on the run of strategy improvement, unless there are two strategies with the same value, in which case it may introduce new improving moves. We refer to this well-known transformation as the *standard transformation*.

### 3 Ranking-based pivot rules

Pivot rules specify, in each iteration of the simplex method, which index enters and which index leaves the basis. Since the inception of the simplex method by Dantzig in 1947 [20], many pivot rules have been proposed. Formally, a pivot rule is a function that gets the data of the linear program as well as the current basis as input, and returns a pair of an entering and a leaving index. Given an LP of the form  $(\text{LP}_{A,\mathbf{b},\mathbf{c}})$  with  $m$  equality constraints and  $n$  variables, its data can be written as elements of  $\mathcal{L}_{m,n} := \{(A, \mathbf{b}, \mathbf{c}) : A \in \mathbb{Q}^{m \times n}, \mathbf{b} \in \mathbb{Q}^m, \mathbf{c} \in \mathbb{Q}^n\}$ . The basis is an element of  $\mathcal{P}([n])$ ; here, we denote the powerset of a finite set  $S$  by  $\mathcal{P}(S) := 2^S$ . Some pivot rules allow for ties between several options of switches, that is, instead of determining a unique pair of an entering and a leaving index, a pivot rule may suggest a set of admissible index pairs. Generalizing further, a pivot rule may return a probability

distribution on the set of sets of feasible index pairs, in which case we call the rule *randomized* – otherwise, it is *deterministic*. To this end, we denote the *probability simplex* of a set  $S$  by  $\Delta(S) = \{p \in [0, 1]^S : \sum_{s \in S} p(s) = 1\}$ . Depending on whether the pivot rule has access to a finite memory of previous steps or not, we call it *memory-based* or *memoryless*, respectively. The number of available memory states may depend on the input size, so we denote it by  $H_{m,n}$ . The following definition gives a unified framework for pivot rules.

► **Definition 3.1.** A (simplex) pivot rule *with memory*  $(H_{m,n})_{n \in \mathbb{N}, m \in [n]} \in \mathbb{N}^{\mathbb{N} \times \mathbb{N}}$  is a family  $\Pi = (\Pi_{m,n})_{n \in \mathbb{N}, m \in [n]}$  of functions

$$\begin{aligned} \Pi_{m,n} : \mathcal{L}_{m,n} \times \mathcal{P}([n]) \times [H_{m,n}] &\rightarrow \Delta(\mathcal{P}([n]^2)) \times [H_{m,n}], \\ ((A, \mathbf{b}, \mathbf{c}), B, h) &\mapsto (p, h'), \end{aligned}$$

such that, if  $B$  is a feasible, non-optimal basis of the linear program  $(LP_{A,\mathbf{b},\mathbf{c}})$ , and, additionally, we have  $(i, j) \in S$  for some  $S \in \mathcal{P}([n]^2)$  with  $p(S) > 0$ , then  $i$  is an improving index for  $B$ , and  $B \cup \{i\} \setminus \{j\}$  is a feasible basis.

Note: we use  $\mathcal{P}([n])$  for ease of notation. In reality, we are only concerned with the cases where the set  $B$  is a feasible basis. We interpret Definition 3.1 as follows. Suppose we apply the simplex method with pivot rule  $\Pi$ , the algorithm is currently at basis  $B$ , and the current memory state of  $\Pi$  is  $h$ . Then  $\Pi$  updates its memory state to  $h'$ , we draw a set  $S \in \mathcal{P}([n]^2)$  according to  $p$ , and we choose one pair  $(i, j) \in S$  of entering index  $i \in \Gamma^+(B)$  and leaving index  $j \in B$  yielding the new basis  $B \cup \{i\} \setminus \{j\}$ . Note that, if  $B$  is not a feasible basis for  $(LP_{A,\mathbf{b},\mathbf{c}})$ , then we do not pose any restriction on the output of  $\Pi$  (since this will never occur in practice).

► **Example 3.2.** This definition covers all classical pivot rules. For example, Zadeh’s rule [55] selects the improving index that was chosen least-often before. Thus, if it does not cycle, Zadeh’s rule can be implemented with  $H_{m,n} = \binom{n}{m}^n$  memory states, allowing it to count the number of times each index was chosen in the past. It returns  $p \in \Delta(\mathcal{P}([n]^2))$  with  $p(S) = 1$  for  $S = \{(i, j) \in [n]^2 : i \in \Gamma^+(B) \text{ chosen least-often}, B \cup \{i\} \setminus \{j\} \text{ feasible basis}\}$ .

Given a pivot rule  $\Pi = (\Pi_{m,n})_{n \in \mathbb{N}, m \in [n]}$ , we may write each  $\Pi_{m,n}$  as the composition of an *information function*  $I_{m,n}^\Pi : \mathcal{L}_{m,n} \times \mathcal{P}([n]) \rightarrow \Omega_{m,n}^\Pi$ , where  $\Omega_{m,n}^\Pi$  can be any set, and a *decision function*  $D_{m,n}^\Pi : \Omega_{m,n}^\Pi \times [H_{m,n}] \rightarrow \Delta(\mathcal{P}([n]^2)) \times [H_{m,n}]$  such that

$$\Pi_{m,n}((A, \mathbf{b}, \mathbf{c}), B, h) = D_{m,n}^\Pi(I_{m,n}^\Pi((A, \mathbf{b}, \mathbf{c}), B), h) \tag{2}$$

for all inputs. There are many ways to decompose a given pivot rule  $\Pi$  into an information and a decision function, for example, we may choose  $\Omega_{m,n}^\Pi = \mathcal{L}_{m,n} \times \mathcal{P}([n])$ , let  $I_{m,n}^\Pi$  be the identity function, and let  $D_{m,n}^\Pi = \Pi_{m,n}$ .

The purpose of this decomposition is to give a natural notion of classes of pivot rules, based on the information they may use. We focus on classes of pivot rules that use combinatorial information, where  $\Omega_{m,n}^\Pi$  is a finite set. We can, e.g., create such a class by fixing the set  $\Omega_{m,n}$ , the information function  $I_{m,n}$ , and the memory  $H_{m,n}$ , while allowing any decision function  $D_{m,n}$ . Of course, the fewer restrictions we pose, the broader the resulting class of pivot rules.

► **Example 3.3.** Consider the class  $\mathcal{C}$  of *combinatorial* pivot rules as defined in [2]. That is, the rules in  $\mathcal{C}$  choose, at every step, the entering index based solely on the signs of all minors of the matrix  $M = \begin{pmatrix} A & \mathbf{b} \\ \mathbf{c}^\top & 0 \end{pmatrix}$ . Thus  $\mathcal{C}$  is exactly the class of pivot rules of the form (2), where  $\Omega_{m,n}$  denotes the finite set of all possible sign patterns of minors of  $M$ , and  $I_{m,n}$  is the function that computes all these signs.

In this paper, we restrict ourselves to *deterministic* pivot rules, which return deterministic distributions  $p \in \Delta(\mathcal{P}([n]^2))$ , i.e.,  $p(S) = 1$  for a unique  $S \in \mathcal{P}([n]^2)$ . Further, we only focus on *non-degenerate* linear programs, where the leaving index  $j$  is always uniquely determined by the entering index  $i$ , for our lower bounds. Therefore, we will use the simplified notation

$$\Pi_{m,n}: \mathcal{L}_{m,n} \times \mathcal{P}([n]) \times [H_{m,n}] \rightarrow \mathcal{P}([n]) \times [H_{m,n}], \quad ((A, \mathbf{b}, \mathbf{c}), B, h) \mapsto (S, h'),$$

in the following; here, if  $B$  is a feasible, non-optimal basis, then  $i \in S$  implies  $i \in \Gamma^+(B)$ . Traditionally, in the worst-case analysis of pivot rules, the *tie-breaking* – that is, choosing an element of  $S$  – is up to the adversary, whose goal is to establish lower bounds, see e.g. [22] for a discussion on this. Therefore, when proving lower bounds, we may assume that the pivot rule never outputs a tie, as we could fix a tie-breaking otherwise. Then, we have  $|S| = 1$  and simplify the notation even further by letting the codomain of  $\Pi_{m,n}$  be  $[n] \times [H_{m,n}]$ . If there is only a single memory state, i.e.  $H_{m,n} = 1$  for all indices  $m$  and  $n$ , we call the pivot rule *memoryless*, in which case we will also leave out the memory component from the notation.

Most pivot rules from the literature select the *best* improving variable(s) according to some underlying measure derived from the input data. Such a measure naturally induces a total preorder<sup>1</sup> on the set of improving variables. In our framework, this happens when the information function  $I$  has a special structure, in which case we refer to  $I$  as a *neighbor ranking*. Given a finite set  $S$ , we denote the set of preorders over  $S$  by  $\mathcal{T}(S)$ .

► **Definition 3.4.** A neighbor ranking is a family  $R = (R_{m,n})_{n \in \mathbb{N}, m \in [n]}$  of functions

$$R_{m,n}: \mathcal{L}_{m,n} \times \mathcal{P}([n]) \rightarrow \mathcal{T}([n]), \quad ((A, \mathbf{b}, \mathbf{c}), B) \mapsto \preceq,$$

such that, if  $B$  is a feasible basis for  $(LP_{A,\mathbf{b},\mathbf{c}})$ , then  $\preceq$  is total on  $\Gamma^+(B)$ , and every non-improving index  $i \in [n] \setminus \Gamma^+(B)$  is incomparable (wrt.  $\preceq$ ) to any  $j \in [n] \setminus \{i\}$ .

Every neighbor ranking allows for a *greedy* pivot rule.

► **Definition 3.5.** Given a neighbor ranking  $R = (R_{m,n})_{n \in \mathbb{N}, m \in [n]}$ , a deterministic, memoryless pivot rule  $\Pi = (\Pi_{m,n})_{n \in \mathbb{N}, m \in [n]}$  is greedy for  $R$  if each  $\Pi_{m,n}$  is of the form (2) with

- $I_{m,n}^\Pi = R_{m,n}$ ,
- $D_{m,n}^\Pi$  returns the set of maximal elements according to  $R_{m,n}((A, \mathbf{b}, \mathbf{c}), B) =: \preceq$ , i.e., it returns  $\{i \in \Gamma^+(B) : i \succeq i', \forall i' \in \Gamma^+(B)\}$ .

► **Observation 3.6.** Given a feasible basis  $B$  for some non-degenerate  $(LP_{A,\mathbf{b},\mathbf{c}})$ , let  $B[i]$  denote the unique feasible basis obtained by adding index  $i$  to  $B$  and removing another index, and let  $\mathbf{x}$  and  $\mathbf{x}[i]$  be the basic feasible solutions with bases  $B$  and  $B[i]$ , respectively. The following classical pivot rules are greedy:

- BLAND [14] orders by index of the improving variable, selecting  $\min(\Gamma^+(B))$ .
- DANTZIG [19] orders by reduced cost, selecting  $\arg \max_{i \in \Gamma^+(B)} c_i - \mathbf{c}_B^\top A_{\cdot B}^{-1} \mathbf{A}_{\cdot i}$ .
- LARGESTINCREASE [38] orders by objective value, selecting  $\arg \max_{i \in \Gamma^+(B)} \mathbf{c}_B[i]^\top \mathbf{x}[i]$ .
- SHADOWVERTEX [46] orders by the ratio of shadow objective decrease to objective increase, selecting  $\arg \max_{i \in \Gamma^+(B)} \frac{\mathbf{d}^\top (\mathbf{x}[i] - \mathbf{x})}{\mathbf{c}^\top (\mathbf{x}[i] - \mathbf{x})}$ , where  $\mathbf{d} \in \mathbb{R}^n$  is the shadow objective.<sup>2</sup>
- STEEPESTEDGE [35] orders by the ratio of objective increase to length of the connecting edge, selecting  $\arg \max_{i \in \Gamma^+(B)} \frac{\mathbf{c}^\top (\mathbf{x}[i] - \mathbf{x})}{\|\mathbf{x}[i] - \mathbf{x}\|}$ .

<sup>1</sup> Recall that a binary relation over a set  $S$  is a *preorder* if it is reflexive and transitive. A preorder over  $S$  is *total* on  $S' \subseteq S$  if any two elements of  $S'$  are comparable. Thus, a total preorder is a total order without the requirement of antisymmetry, allowing for ties between distinct elements.

<sup>2</sup> The shadow vertex rule is greedy under the assumption that  $\mathbf{d}$  is fixed beforehand, instead of depending on the initial basis (which would require some memory). This assumption has no effect on worst-case analysis.

Now we are ready to introduce the class of *ranking-based* pivot rules, which generalizes these classical rules by relaxing their greedy behavior and, additionally, allowing the pivot rules to depend on a set  $\mathcal{R} = \{R^1, \dots, R^k\}$  of neighbor rankings  $R^i$ . That is, a ranking-based rule should base its decision solely on the relative positions of the improving variables in the preorders  $R_{m,n}^i((A, \mathbf{b}, \mathbf{c}), B)$ . However, by definition, a pivot rule outputs improving indices, so it must have access to these indices as well. With this information, it could, say, treat variable  $x_5$  differently from  $x_8$ , or even specify its decision for every basis. To prevent the rule from abusing this knowledge, we add an additional symmetry condition, forcing the rule to only base its decision on the relative positions in the rankings. This is done by ensuring that whenever the rule encounters two equivalent situations with equivalent rankings, it is forced to make equivalent choices.

► **Definition 3.7.** Let  $\mathcal{R} = \{R^1, \dots, R^k\}$  be a fixed set of neighbor rankings. A deterministic pivot rule  $\Pi = (\Pi_{m,n})_{n \in \mathbb{N}, m \in [n]}$  is called  $\mathcal{R}$ -ranking-based if

- Each  $\Pi_{m,n}$  is of the form (2) with  $\Omega_{m,n}^\Pi = \mathcal{T}([n])^k$ , and for all inputs we have

$$I_{m,n}^\Pi((A, \mathbf{b}, \mathbf{c}), B) = (R_{m,n}^1((A, \mathbf{b}, \mathbf{c}), B), \dots, R_{m,n}^k((A, \mathbf{b}, \mathbf{c}), B)) =: (\preceq_B^{R_1}, \dots, \preceq_B^{R_k}),$$

- For every two linear programs  $L \in \mathcal{L}_{m,n}$ ,  $L' \in \mathcal{L}_{m',n'}$  with respective feasible, non-optimal bases  $B, B'$ , and every memory state  $h \leq \min\{H_{m,n}, H_{m',n'}\}$ , if there exists a bijection  $\varphi: \Gamma^+(B) \rightarrow \Gamma^+(B')$  preserving each ranking, i.e.,

$$i \preceq_B^{R_j} i' \iff \varphi(i) \preceq_{B'}^{R_j} \varphi(i'), \quad \forall i, i' \in \Gamma^+(B), \forall j \in \{1, 2, \dots, k\},$$

and  $\Pi_{m,n}(L, B, h) = (S, h_1)$  and  $\Pi_{m',n'}(L', B', h) = (S', h'_1)$ , then  $\varphi(S) = S'$ , and, additionally, if  $n = n'$ , then also  $h_1 = h'_1$ .

Every deterministic memoryless pivot rule can be formulated as a ranking-based rule: choose the ranking such that the first improving variables in the ranking are the ones preferred by the pivot rule. Then, choose as the decision function the function that always picks the first variables. This gives an equivalent ranking-based rule.

Recall that the information function, which specifies a tuple of neighbor rankings in the class of ranking-based rules, is always memoryless, while the decision function may be memory-based. As an example, let BLAND denote the neighbor ranking that orders by index, then a  $\{\text{BLAND}\}$ -ranking-based pivot rule  $\Pi$  may select the improving variable with the second smallest index. If  $\Pi$  is also memory-based it may, e.g., alternately select the improving variable with the smallest and the largest index. We analyze the class of memory-based  $\{\text{BLAND}\}$ -ranking-based rules, which we call *index-based* rules for short, in Section 4.

## 4 Lower bounds for index-based pivot rules

Recall that Bland's rule always picks the improving variable that has the lowest index. In this section, we consider  $\{\text{BLAND}\}$ -ranking-based rules with memory and without ties. Recalling Definition 3.7, these rules take into consideration only the order of the indices of the improving variables, and the current memory state. The purpose of this section is to show that  $\{\text{BLAND}\}$ -ranking-based rules need a superpolynomial number of steps, even when equipped with some memory. We can let the memory only depend on  $n$  without losing generality, since we always have  $m \leq n$ , so the number of memory states is always bounded by  $\max_{m \in [n]} H_{m,n}$  if  $n$  is constant. We then denote the allowed memory of a pivot rule  $\Pi$  by  $H_{m,n} = \ell(n)$ .

Suppose that for some LP with 100 variables, there is a feasible basis  $B$  from which there are 20 improving moves, and that a {BLAND}-ranking-based rule picks the improving move with the 12<sup>th</sup> lowest index for this basis when the memory state is 5. The second condition (symmetry requirement) in Definition 3.7 then forces this rule to always pick the 12<sup>th</sup>-lowest index whenever there are 100 variables, 20 improving moves, and the memory state is 5. For this reason, the choices of the pivot rule are determined by only three parameters: the number of variables and improving variables, and the memory state.

This means we can write  $\Pi$  in terms of a function  $P: \mathbb{N}^3 \rightarrow \mathbb{N}^2$ , whose inputs represent these three parameters, and its outputs represent the chosen index and the memory update. To be precise, we have  $P(k, n, h) \in \{1, 2, \dots, k\} \times \{1, 2, \dots, \ell(n)\}$ , for all  $k, n, h \in \mathbb{N}^3$ . Then  $\Pi_{m,n}((A, \mathbf{b}, \mathbf{c}), B, h)$  is given as follows: if  $P(|\Gamma^+(B)|, n, h) = (i', h')$ , and  $i$  is the  $i'$ -th smallest index in  $\Gamma^+(B)$ , then  $\Pi_{m,n}((A, \mathbf{b}, \mathbf{c}), B, h) = (i, h')$ . We refer to a function  $P$  of this form as an  $\ell$ -index-selector function, and we say  $P$  induces the  $\ell$ -index-based pivot rule  $\Pi^P$  in this manner. The function  $P$  does not need  $m$  as input: for different  $m$  the output must still be the same by the second condition of Definition 3.7. Like the so-called *normalized-weight* rules in [13, Cor. 1.4], index-based rules are universal in the following sense.

► **Observation 4.1.** *Given some  $(LP_{A,\mathbf{b},\mathbf{c}})$ , if there exists a pivot rule that reaches the optimum in  $\ell(n)$  iterations, then there also exists an  $\ell$ -index-based rule reaching the optimum in  $\ell(n)$  iterations. In particular, any parity game with  $n$  player 0 states can be solved in linear time by some  $n$ -index-based improvement rule.*

**Proof.** Let  $(B_0, B_1, \dots, B_{\ell(n)})$  be the sequence of bases of  $(LP_{A,\mathbf{b},\mathbf{c}})$  visited by the pivot rule, where  $B_{\ell(n)}$  denotes the optimal basis. Then, we can choose a memory state  $h_i$  for every  $i \in \{0, 1, \dots, \ell(n) - 1\}$  such that, if the current memory state is  $h_i$  and the current basis is  $B_i$ , the index-based rule transitions to the basis  $B_{i+1}$  and updates the memory state to  $h_{i+1}$ . For parity games, the optimum can always be reached in a linear number of improving switches, given perfect choices of the improvement rule [30, Lem. 4.2]. ◀

Recalling that the polyhedra with the largest known diameters are linear in the dimension, this illustrates the power of this class of pivot rules. This contrasts in some sense with the following theorem, where the rest of this section is dedicated to its proof.<sup>3</sup>

► **Theorem 4.2.** *Let  $\ell(n) = o\left(\frac{n}{\log(n)}\right)$ . Then, for any  $\ell$ -index-based rule  $\Pi^P$ , there exists a family  $(LP_n)_{n \in \mathbb{N}}$  of linear programs  $LP_n$  in  $n$  variables and  $m \leq n$  constraints such that the simplex method with pivot rule  $\Pi^P$  requires  $n^{\omega(1)}$  iterations to find the optimal basis.*

Because of the combinatorial nature of index-based pivot rules, they can also be interpreted as improvement rules for strategy iteration in parity games. Then, the input  $(p, m, h_1)$  of an  $\ell$ -index-selector function  $P$  consists of the number  $p$  of improving edges available at the current strategy, the number  $m$  of player 0 controlled edges, and the current memory state  $h_1$ . Note that here, and in the remainder of this section,  $m$  denotes the number of player 0 edges in a sink parity game, which corresponds by Theorem 2.2 to the number of variables in its related LP (which was denoted by  $n$  before). This allows one to infer Theorem 4.2 from the following statement for parity games.

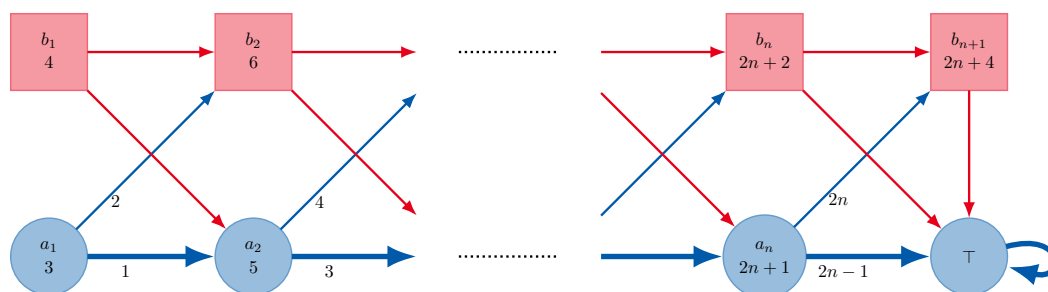
► **Theorem 4.3.** *Suppose  $\ell(m) = o\left(\frac{m}{\log(m)}\right)$ . Then, for any  $\ell$ -index-based improvement rule  $\Pi^P$ , there is a family  $(\mathcal{G}_m)_{m \in \mathbb{N}}$  of sink parity games  $\mathcal{G}_m$  with  $m$  player 0 edges, such that strategy improvement with pivot rule  $\Pi^P$  requires  $m^{\omega(1)}$  iterations to find the optimal strategy.*

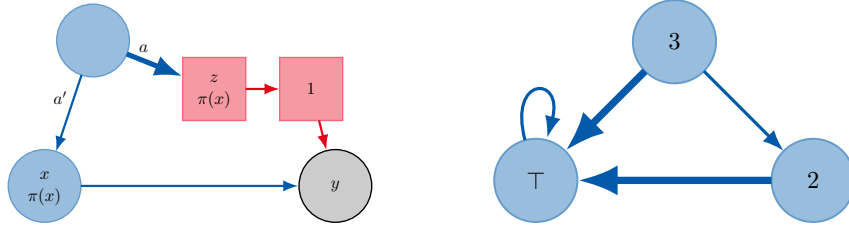
<sup>3</sup> All missing proofs are deferred to the full version.

Since the number  $N$  of non-zero input values is polynomial in  $n$  in Theorem 4.2, and polynomial in  $m$  in Theorem 4.3, the algorithms also take  $N^{\omega(1)}$  iterations, implying Theorem 1.1. We construct these parity games in several steps, starting from special cases and gradually introducing further gadgets to cover the general setting. We start with the basic case of Bland’s rule. This well-known pivot rule in LPs picks the improving variable with the lowest index to enter the basis. To use this rule in a sink parity game, we need to assign each player 0 edge a *Bland number*, indicating its index. Using the notation for index-based improvement rules, Bland’s rule is  $\Pi^P$ , where  $P$  is a selector function such that the first element of its output is always 1, independent of the input.

The games that we use to prove Theorem 4.3 all have the same base structure. For every  $n \in \mathbb{N}$ , we define a sink parity game  $G_n$ , which is shown in Figure 1.

► **Lemma 4.4.** *Strategy improvement with Bland’s rule takes  $2^n - 1$  iterations to solve the sink parity game  $G_n$  with initial strategy  $\sigma_0$ . Moreover, the order of the valuations of the nodes  $a_1$  and  $b_1$  changes in every iteration.*





■ **Figure 2** Left: the controller gadget, which ensures a constant number of improving switches.  $x, y$  and  $z$  are node labels, all other labels in the nodes denote priorities. Right: The filler gadget (the 2 and 3 denote priorities). The bold edges are always part of the initial strategy.

► **Definition 4.5.** Let  $m, \ell \in \mathbb{N}$  such that  $m \geq 4\ell$ . Let  $\mathcal{S} = (i_1, i_2, \dots, i_{\ell'}) \in [m]^{\ell'}$  be a sequence of length  $\ell' \leq \ell$ , and let  $I = \{i_1, i_2, \dots, i_{\ell'}\}$ . We say that  $\mathcal{S}$  is  $(m, \ell)$ -clustered if there exists some  $k \in \mathbb{N}$  and integers  $p_1, p_2, \dots, p_k \in [m]$  and  $q_1, \dots, q_k \in \mathbb{N}$  such that

- (i) the intervals  $[p_1, p_1 + q_1], [p_2, p_2 + q_2], \dots, [p_k, p_k + q_k]$  are pairwise disjoint and contained in the interval  $[1, m]$ , and the union of these intervals contains  $I$ ,
- (ii) for every  $c \in [k]$ , and for  $K_c := [p_c, p_c + q_c] \cap I$ , we have

$$q_c + 1 \geq \lfloor \frac{m}{2\ell} \rfloor \cdot \min(\max(K_c) - p_c + 1, p_c + q_c - \min(K_c) + 1),$$

- (iii)  $\sum_{j=1}^k (q_j + 1) + 2(\ell - \ell') \leq m$ .

We use this notion to define a class of improvement rules, by saying that the  $\ell$ -index-selector function  $P$  is  $(m_i, \ell_i)$ -clustered if the sequence  $g_1, g_2, \dots, g_{\ell'_i}$  is  $(\frac{m_i}{3}, \ell_i)$ -clustered<sup>4</sup>. Intuitively, this means the preferences of the improvement rule lie in small clusters. That allows us to state the following result.

► **Lemma 4.6.** Suppose we have an  $\ell$ -index-based improvement rule  $\Pi^P$ , suppose  $m_i$  is divisible by 3,  $\ell_i := \ell(m_i)$ , and  $m_i \geq 12\ell_i$ . If  $P$  is  $(m_i, \ell_i)$ -clustered, then there exists a parity game with  $m_i$  player 0 edges, which takes at least  $2^{\frac{m_i}{12\ell_i} - 1}$  iterations to solve for SI with improvement rule  $\Pi^P$ .

The proof idea of Lemma 4.6 is as follows: since  $P$  is  $(m_i, \ell_i)$ -clustered, this means that the outputs of  $P$  are contained in intervals  $[p_c, p_c + q_c]$ , where the outputs of  $P$  are either all close to the start or to the end of the interval. We make a copy of the binary counter from Figure 1 for each interval, and we replace each player 0 edge a fixed number of copies of itself. This artificially increases the number of improving moves, so that it suffices for the pivot rule to choose the indices *close* to the smallest index in each copy of the binary counter.

Furthermore, we need to make sure that the total number of improving moves stays constant. This is done by replacing each player 0 edge  $(x, y)$  by the controller gadget in Figure 2. Finally, to make sure we have exactly  $m_i$  player 0 edges in total, we add a number of filler gadgets from Figure 2, carefully giving them the right Bland numbers.

We also need another technical definition.

► **Definition 4.7.** Let  $m, \ell \in \mathbb{N}$  such that  $m \geq 4\ell$ . Let  $\mathcal{S} = (i_1, i_2, \dots, i_{\ell'}) \in [m]^{\ell'}$  be a sequence of length  $\ell' \leq \ell$ , and let  $I = \{i_1, i_2, \dots, i_{\ell'}\}$ . We say that  $\mathcal{S}$  is  $(m, \ell)$ -dispersed if there exists some  $k \in \mathbb{Z}_{\geq 0}$  and integers  $\psi, p_1, p_2, \dots, p_k \in [m]$  and  $\xi, q_1, \dots, q_k \in \mathbb{N}$  such that, for  $K = I \cap [\psi, \psi + \xi]$ ,

<sup>4</sup> This is well-defined since  $m_i \geq 12\ell_i \geq 12\ell'_i$  and therefore  $\frac{m_i}{3} \geq 4\ell'_i$ .

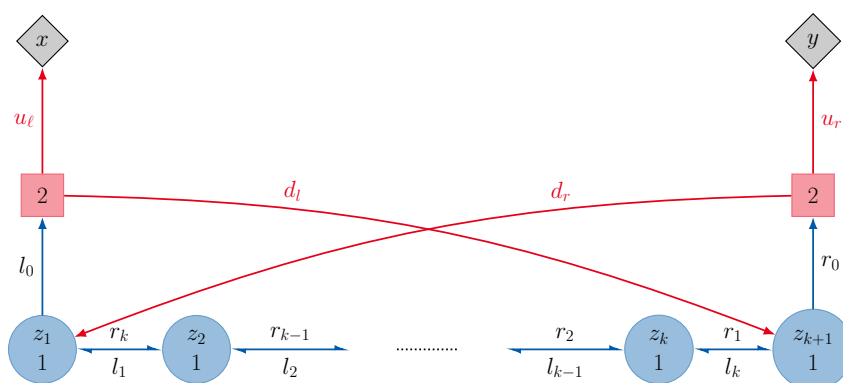


Figure 3 The delayer gadget. It makes a significant switch take  $k + 1$  times more iterations.

- (i) the intervals  $[\psi, \psi + \xi], [p_1, p_1 + q_1], [p_2, p_2 + q_2], \dots, [p_k, p_k + q_k]$  are pairwise disjoint and contained in  $[1, m]$ , and the union of these intervals contains  $I$ ,
- (ii) for  $j = 1, 2, \dots, k$ , we have  $q_j \geq 2 \cdot |\{c \in [\ell'] : i_c \in [p_j, p_j + q_j]\}| - 1$ ,
- (iii) either  $K = \{\psi\}$  or  $K = \{\psi + \xi\}$ ,
- (iv)  $\xi \geq \lfloor \frac{m}{2\ell} \rfloor |\{c : i_c \in K\}| - 1$ ,
- (v)  $(\xi + 1) + \sum_{j=1}^k (q_j + 1) + 2(\ell - \ell') \leq m$ .

We say the  $\ell$ -index-selector function  $P$  is  $(m_i, \ell_i)$ -dispersed if the sequence  $(g_1, g_2, \dots, g_{\ell'})$  is  $(\frac{m_i}{3}, \ell_i)$ -dispersed. Intuitively, this means there is one big interval  $[\psi, \psi + \xi]$  in which the pivot rule has almost no outputs. We prove the following lemma:

► **Lemma 4.8.** *Suppose we have an  $\ell$ -index-based improvement rule  $\Pi^P$ , suppose  $m_i$  is divisible by 3,  $\ell_i := \ell(m_i)$ , and  $m_i \geq 12\ell_i$ . If  $P$  is  $(m_i, \ell_i)$ -dispersed, then there exists a parity game with  $m_i$  player 0 edges, which takes at least  $2^{\frac{m_i}{12\ell_i} - 1}$  iterations to solve for SI with improvement rule  $\Pi^P$ .*

The main proof idea for Lemma 4.8 is as follows: the dispersedness tells us that there is a large interval  $[\psi, \psi + \xi] \subseteq [1, m]$ , in which the the outputs of the improvement rule are either all at the start or at the end. We take one copy of the binary counter from Figure 1, and then use the delayer gadget as seen in Figure 3: each player 0 node with two outgoing edges towards nodes  $x$  and  $y$  is replaced by this gadget. By doing this, it essentially delays a switch by  $k$  moves: it now takes  $k + 1$  iterations to get the same result as usually one iteration would. We pick the Bland numbers such that this counter will be identified with  $[\psi, \psi + \xi]$ , and we pick the parameters  $k$  from the gadgets such that each iteration of the original binary counter takes exactly as long as it takes the improvement rule to repeat its memory state. We call the time until reaching the same memory state a cycle.

In each cycle,  $P$  can give many different outputs, so we need to deal with the outputs that are outside  $[\psi, \psi + \xi]$ . By Lemma 4.4, the (replacements of)  $a_1$  and  $b_1$  alternate which of them is the best, every cycle. We attach additional copies of the delayer gadget to our game, identifying  $x$  with  $b_1$  and  $y$  with  $a_1$  in each gadget. This creates new improving moves every cycle for every output of  $P$  outside the interval. Finally, like for Lemma 4.6, we use controller and filler gadgets from Figure 2 to make sure the number of improving moves is constant and the total number of edges equals  $m_i$ .

## 4.2 Completing the proof

We combine the results of Lemmas 4.6 and 4.8 with the following lemma.

► **Lemma 4.9.** *Let  $m, \ell \in \mathbb{N}$  such that  $m \geq 4\ell$ . Let  $\mathcal{S} = (i_1, i_2, \dots, i_{\ell'}) \in [m]^{\ell'}$  be a sequence of length  $\ell' \leq \ell$ . Then  $\mathcal{S}$  is  $(m, \ell)$ -clustered or  $(m, \ell)$ -dispersed.*

Finally, we drop the assumption that  $P(\frac{m_i}{3}, m_i, h_{\ell'_i}) = (s, h_1)$  for some  $s$ , which we made at the start. In general, the memory states visited by the improvement rule do not just form one repeating sequence; if  $\ell''_i$  is such that  $P(\frac{m_i}{3}, m_i, h_{\ell''_i}) = (s, h_{\ell''_i})$  and the number of improving moves stays  $\frac{m_i}{3}$ , then the choices output by  $P$  are

$$(g_1, g_2, \dots, g_{\ell''_i}, g_{\ell''_i+1}, \dots, g_{\ell'_i}, g_{\ell'_i}, g_{\ell'_i+1}, \dots, g_{\ell'_i}, g_{\ell'_i}, g_{\ell'_i+1}, \dots)$$

In this case, we can modify the constructions from Lemmas 4.6 and 4.8, by simply adding some filler gadgets that account for the one-time switches related to  $g_1, g_2, \dots, g_{\ell''_i-1}$ .

► **Lemma 4.10.** *Suppose we have an  $\ell$ -index-based improvement rule  $\Pi^P$ , suppose  $m_i$  is divisible by 3,  $\ell_i := \ell(m_i)$ , and  $m_i \geq 12\ell_i$ . There exists a parity game with  $m_i$  player 0 edges, which takes at least  $2^{\frac{m_i}{12\ell_i}-1}$  iterations to solve for SI with improvement rule  $\Pi^P$ .*

To conclude, by definition  $\frac{m_i}{\ell_i} \geq 3i \log(m_i)$ , so  $2^{\frac{m_i}{12\ell_i}-1} \geq 2^{\frac{1}{4} \log(m_i)-1} - 1 = \frac{1}{2} m_i^{\frac{1}{4}} - 1$ . Since  $i \rightarrow \infty$ , this yields that  $\Pi^P$  takes a superpolynomial number of steps in the worst case to solve sink parity games. Hence this completes the proof of Theorem 4.3.

Finally, to complete the proof of Theorem 4.2, it suffices to show that we can assume that all priorities in the game are unique by Theorem 2.2. One can quickly check that all the constructions and the gadgets used are not significantly<sup>5</sup> affected by the standard transformation. This completes the proof of Theorem 4.2: we proved that no strongly polynomial index-based pivot rule exists when fewer than  $o(n/\log n)$  memory states are available. The term “strongly” cannot be left out here, since the LP that is implicitly constructed via Theorem 2.2 has doubly exponential coefficients. It remains open whether this lower bound extends to rules with larger memory.

► **Question 1.** *Is there a super-polynomial lower bound on the running time of index-based pivot rules when at least  $\Omega(n/\log n)$  memory states are available?*

## 5 Lower bounds for a memoryless subclass of ranking-based pivot rules

Three classical simplex pivot rules are Dantzig’s original pivot rule, Bland’s rule, and the largest-increase rule, which greedily select the improving variable with the largest reduced cost, the smallest global index, and the largest associated objective improvement, respectively; see Observation 3.6. We use the names BLAND, DANTZIG, and LARGESTINCREASE to refer both to the pivot rules and to the underlying neighbor rankings, see Definition 3.5. Conceptually, as we will see in the following, the inefficient worst-case behavior of these rules [6, 38, 42] arises not merely from their greedy nature but already from the fact that the information they rely on is too limited. More precisely, our main result in this section is that every {BLAND, DANTZIG, LARGESTINCREASE}-ranking-based simplex pivot rule (see Definition 3.7) is at least subexponential. Equivalently, every deterministic, memoryless simplex pivot rule that

<sup>5</sup> There may be slight changes in the order of non-significant switches due to edge copying in the proof of Lemma 4.6.

bases its decision solely on the number of improving variables and their relative rankings by index, reduced cost, and objective improvement requires subexponential time in the worst-case.

► **Theorem 5.1.** *Let  $\Pi$  be a  $\{\text{BLAND}, \text{DANTZIG}, \text{LARGESTINCREASE}\}$ -ranking-based pivot rule. Then, there exists a family of linear programs with  $N$  non-zero entries in the constraint matrix such that the simplex method with pivot rule  $\Pi$  takes  $\Omega(2^{\sqrt{N}})$  iterations.*

Given the close connection between `STEEPESTEDGE` and `SHADOWVERTEX` [12], it is natural to ask whether our results can be extended to include these rules as well.

► **Question 2.** *Does there exist a polynomial-time  $\{\text{BLAND}, \text{DANTZIG}, \text{LARGESTINCREASE}, \text{STEEPESTEDGE}, \text{SHADOWVERTEX}\}$ -ranking-based pivot rule?*

The remainder of this section is devoted to proving that no polynomial  $\{\text{BLAND}, \text{DANTZIG}, \text{LARGESTINCREASE}\}$ -ranking-based pivot rule exists for `POLICYITERATION`. Although ranking-based pivot rules were defined only for the simplex method, the definitions carry over directly via the correspondence between `POLICYITERATION` and the simplex method, see Theorem 2.1. Explicitly, a neighbor ranking assigns to each policy of the Markov decision process a total preorder on the set of its improving switches. Consequently, the following lower bound for `POLICYITERATION` implies Theorem 5.1 since the number of non-zero matrix entries in the induced linear program is linear in the number of transition probabilities in the process, see [47] or [36, Sec. 2.4]. Both theorems together prove Theorem 1.2.

► **Theorem 5.2.** *Let  $\Pi$  be a  $\{\text{BLAND}, \text{DANTZIG}, \text{LARGESTINCREASE}\}$ -ranking-based pivot rule. Then, there exists a family of weak unichain Markov decision processes with  $N$  non-zero transition probabilities such that `POLICYITERATION` with  $\Pi$  takes  $\Omega(2^{\sqrt{N}})$  iterations.*

We still assume, as motivated in Section 3, that  $\Pi$  does not return ties. Moreover, in our constructions in this section, the considered neighbor rankings always return total orders on the set of improving variables, that is, there are no ties in  $\preceq_B$ .

Let  $\mathcal{R} = \{R_1, \dots, R_k\}$  be a fixed set of neighbor rankings. Then, by Definition 3.7, a memoryless  $\mathcal{R}$ -ranking-based rule  $\Pi$  bases its decisions only on the neighbor rankings in  $\mathcal{R}$ . Consider the special case where the simplex method is at a basis  $B$  such that all neighbor rankings in  $\mathcal{R}$  induce the same total preorder  $\preceq_B$  on the set of improving neighbors  $\Gamma^+(B)$ , that is, we have  $\preceq_B^{R_1} = \dots = \preceq_B^{R_k} = \preceq_B$ . In this situation, the only information that  $\Pi$  has about improving switches is their relative positions in the common preorder  $\preceq_B$ . At such bases,  $\Pi$  effectively only sees a single ranking of the improving variables. Thus, since  $\Pi$  has no access to the underlying data and due to our symmetry condition in Definition 3.7, its decisions are of the form ‘whenever given 20 improving variables, choose the 12<sup>th</sup>-lowest index in  $\preceq_B$ ’.

We can use a similar idea to  $\ell$ -improving functions from Section 4. In the case where all neighbor rankings agree, we can write  $\Pi$  in terms of a function  $f_\Pi: \mathbb{N} \rightarrow \mathbb{N}$  with  $f_\Pi(k) \in \{1, 2, \dots, k\}$ , for all  $k \in \mathbb{N}$ , such that  $\Pi_{m,n}((A, \mathbf{b}, \mathbf{c}), B) = i$ , where  $i$  is the index of the  $f_\Pi(|\Gamma^+(B)|)$ -th smallest improving variable in  $\Gamma^+(B)$  according to  $\preceq_B$ . Note that  $f_\Pi$  is independent of  $m$  and  $n$ , since  $\Pi$  does not depend on  $m, n$  by Definition 3.7.

Now fix an arbitrary  $\{\text{BLAND}, \text{DANTZIG}, \text{LARGESTINCREASE}\}$ -ranking-based pivot rule  $\Pi$ . We analyze the running time of `POLICYITERATION` with pivot rule  $\Pi$  depending on the structure of the function  $f_\Pi$ , which determines the choice of  $\Pi$  at policies where all three neighbor rankings agree. More precisely, depending on the behavior of  $f_\Pi$ , we construct four different families of Markov decision processes. Then, we argue that, on each of these families,

POLICYITERATION with pivot rule  $\Pi$  visits a (sub-)exponential number of policies when started at a suitable initial policy. The common key feature of the constructed families is that the three neighbor rankings agree at each of the visited policies, such that the behavior of  $\Pi$  is fully determined by  $f_\Pi$  during the run of the algorithm.

First, we consider the case  $f_\Pi \equiv 1$ . That is, whenever the neighbor rankings BLAND, DANTZIG, and LARGESTINCREASE agree, the pivot rule  $\Pi$  chooses the least-preferred improving switch in the common ranking. We prove an exponential lower bound for such  $\Pi$ . The core structure of the constructed family of Markov decision processes is based on a lower bound construction for parity games given in [43]. As in Section 4, we divide the processes into levels, or *bits*, and prove that the algorithm simulates a *binary counter*. Since  $f_\Pi$  selects the least-preferred switch, the main feature of our processes is that improving switches in low levels are less improving than those in higher levels.

Second, we prove a subexponential lower bound for the case  $f_\Pi(k) = o(\sqrt{k})$ . The key observation for the proof is the following: if  $\mathcal{M}$  is one of our processes from the first case, where  $f_\Pi \equiv 1$ , and we *copy* each of its  $n$  actions  $n$  times, then we obtain a new process  $\mathcal{M}'$  with  $n' = n^2$  actions such that switching the least-preferred action in  $\mathcal{M}$  corresponds to switching one of its  $n = \sqrt{n'}$  copies in  $\mathcal{M}'$ . Thus, if  $n$  is large enough, the condition  $f_\Pi(k) = o(\sqrt{k})$  yields that  $\Pi$  on the new process  $\mathcal{M}'$  mimics the behavior from the first case. Since the number  $n'$  of actions in  $\mathcal{M}'$  is quadratic in  $n$ , we obtain a subexponential lower bound of  $\Omega(2^n) = \Omega(2^{\sqrt{n'}})$ .

Third, we consider the case  $f_\Pi(k) = k$ , for all  $k \in \mathbb{N}$ , and give an exponential lower bound. Our proof strengthens the exponential lower bound of [23] by ensuring that all three neighbor rankings agree at every visited policy, which is necessary for our  $f_\Pi$ -based analysis. The main approach for our construction is to take the processes from the first case, where  $f_\Pi \equiv 1$ , and replace each action by some *gadget* that was introduced in [23]. Informally, this gadget allows us to scale the reduced costs and objective improvements of improving switches at the expense of introducing additional improving switches, whose influence must be controlled in the construction. By choosing the parameters in these gadgets such that switches in lower levels have larger reduced costs and associated objective improvements than those in higher levels, we obtain the desired exponential bound.

Finally, we prove a subexponential lower bound for the case  $f_\Pi(k) \neq o(\sqrt{k})$ , by extending the Markov decision processes that were used for the third case. By assumption, there exists a monotone sequence  $(k_i)_{i \in \mathbb{N}} \in \mathbb{N}^{\mathbb{N}}$  such that  $f_\Pi(k_i) \geq c\sqrt{k_i}$ , for some constant  $c > 0$  and for all  $i \in \mathbb{N}$ . For each  $i \in \mathbb{N}$ , we take one of the processes from the third case, and equip it with gadgets such that, for a suitable initial policy and during a sufficient number of iterations, the number of improving switches is exactly  $k_i$ , and the  $f_\Pi(k_i)$ -th least-preferred improving switch in the new process corresponds to the most-preferred switch in the original process. This yields a subexponential bound since  $\Pi$  mimics the behavior from the third case on the new process, whose size is quadratic in the size of the original process.

Combining the results for  $f_\Pi(k) = o(\sqrt{k})$  and  $f_\Pi(k) \neq o(\sqrt{k})$  yields Theorem 5.2, which in turn implies Theorem 5.1; these two theorems establish our second main result, Theorem 1.2.

---

## References

- 1 Ilan Adler, Christos Papadimitriou, and Aviad Rubinfeld. On simplex pivoting rules and complexity theory. In *International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 13–24. Springer, 2014. doi:10.1007/978-3-319-07557-0\_2.

- 2 Xavier Allamigeon, Pascal Benchimol, Stéphane Gaubert, and Michael Joswig. Combinatorial simplex algorithms can solve mean payoff games. *SIAM Journal on Optimization*, 24(4):2096–2117, 2014. doi:10.1137/140953800.
- 3 Xavier Allamigeon, Daniel Dadush, Georg Loho, Bento Natura, and László A Végh. Interior point methods are not worse than simplex. *SIAM Journal on Computing*, pages FOCS22–178, 2025. doi:10.1137/23M1554588.
- 4 Xavier Allamigeon, Stéphane Gaubert, and Nicolas Vandame. No self-concordant barrier interior point method is strongly polynomial. In *Proceedings of the 54th Annual ACM Symposium on Theory of Computing (STOC)*, pages 515–528, 2022. doi:10.1145/3519935.3519997.
- 5 Nina Amenta and Günter M Ziegler. Deformed products and maximal shadows of polytopes. *Contemporary Mathematics*, 223:57–90, 1999. doi:10.1090/conm/223/03132.
- 6 David Avis and Vasek Chvátal. Notes on Bland’s pivoting rule. In *Polyhedral Combinatorics: Dedicated to the memory of D.R. Fulkerson*, Mathematical Programming Studies, pages 24–34. Springer, 1978. doi:10.1007/BFb0121192.
- 7 David Avis and Oliver Friedmann. An exponential lower bound for Cunningham’s rule. *Mathematical Programming*, 161(1):271–305, 2017. doi:10.1007/s10107-016-1008-4.
- 8 Eleon Bach, Yann Disser, Sophie Huiberts, and Nils Mosis. An unconditional lower bound for the active-set method in convex quadratic maximization, 2025. doi:10.48550/arXiv.2507.16648.
- 9 Richard Bellman. Dynamic programming. *science*, 153(3731):34–37, 1966. doi:10.1126/science.153.3731.34.
- 10 Pascal Benchimol. *Tropical aspects of linear programming*. PhD thesis, Ecole Polytechnique, 2014. URL: <https://hal-polytechnique.archives-ouvertes.fr/tel-01198482>.
- 11 Henrik Björklund and Sergei Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. *Discrete Applied Mathematics*, 155(2):210–229, 2007. doi:10.1016/j.dam.2006.04.029.
- 12 Alexander E Black. Exponential lower bounds for many pivot rules for the simplex method. In *International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 86–99. Springer, 2025. doi:10.1007/978-3-031-93112-3\_7.
- 13 Alexander E Black, Jesús A De Loera, Niklas Lütjeharms, and Raman Sanyal. The polyhedral geometry of pivot rules and monotone paths. *SIAM Journal on Applied Algebra and Geometry*, 7(3):623–650, 2023. doi:10.1137/22M1475910.
- 14 Robert G Bland. New finite pivoting rules for the simplex method. *Mathematics of Operations Research*, 2(2):103–107, 1977. doi:10.1287/moor.2.2.103.
- 15 Karl Heinz Borgwardt. The Shadow-Vertex Algorithm. In *The Simplex Method: A Probabilistic Analysis*, Algorithms and Combinatorics, pages 62–111. Springer, 1987. doi:10.1007/978-3-642-61578-8\_2.
- 16 Jean Cardinal and Raphael Steiner. Inapproximability of shortest paths on perfect matching polytopes. *Mathematical Programming*, 210(1):147–163, 2025. doi:10.1007/s10107-023-02025-4.
- 17 Daniel Dadush and Sophie Huiberts. A friendly smoothed analysis of the simplex method. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC)*, pages 390–403, 2018. doi:10.1145/3188745.3188826.
- 18 Daniel Dadush, Zhuan Khye Koh, Bento Natura, Neil Olver, and László A Végh. A strongly polynomial algorithm for linear programs with at most two nonzero entries per row or column. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1561–1572, 2024. doi:10.1145/3618260.3649764.
- 19 George Dantzig. *Linear programming and extensions*. Princeton university press, 1963. doi:10.1515/9781400884179.
- 20 George B. Dantzig. Reminiscences about the origins of linear programming. *Operations Research Letters*, 1(2):43–48, 1982. doi:10.1016/0167-6377(82)90043-8.

- 21 Jesús A De Loera, Sean Kafer, and Laura Sanità. Pivot rules for circuit-augmentation algorithms in linear optimization. *SIAM Journal on Optimization*, 32(3):2156–2179, 2022. doi:10.1137/21M1419994.
- 22 Yann Disser, Oliver Friedmann, and Alexander V. Hopp. An exponential lower bound for Zadeh’s pivot rule. *Mathematical Programming*, 199(1):865–936, 2023. doi:10.1007/s10107-022-01848-x.
- 23 Yann Disser and Nils Mosis. A unified worst case for classical simplex and policy iteration pivot rules. In *34th International Symposium on Algorithms and Computation (ISAAC)*, pages 27–1, 2023. doi:10.4230/LIPIcs.ISAAC.2023.27.
- 24 Yann Disser and Nils Mosis. An unconditional lower bound for the active-set method on the hypercube. In *Proceedings of the 23rd Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 213–227, 2025. doi:10.1007/978-3-031-93112-3\_16.
- 25 Yann Disser and Martin Skutella. The Simplex Algorithm Is NP-Mighty. *ACM Transactions on Algorithms (TALG)*, 15(1):5:1–5:19, 2018. doi:10.1145/3280847.
- 26 John Fearnley. Exponential lower bounds for policy iteration. In *Proceedings of the 37th International Colloquium Conference on Automata, Languages and Programming: Part II, ICALP’10*, pages 551–562, Berlin, Heidelberg, 2010. Springer-Verlag. doi:10.1007/978-3-642-14162-1\_46.
- 27 John Fearnley and Rahul Savani. The complexity of the simplex method. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing (STOC)*, pages 201–208, 2015. doi:10.1145/2746539.2746558.
- 28 Nathanaël Fijalkow, Nathalie Bertrand, Patricia Bouyer-Decitre, Romain Brenguier, Arnaud Carayol, John Fearnley, Hugo Gimbert, Florian Horn, Rasmus Ibsen-Jensen, Nicolas Markey, Benjamin Monmege, Petr Novotný, Mickael Randour, Ocan Sankur, Sylvain Schmitz, Olivier Serre, and Mateusz Skomra. Games on Graphs, 2023. doi:10.48550/arXiv.2305.10546.
- 29 Oliver Friedmann. An exponential lower bound for the parity game strategy improvement algorithm as we know it. In *2009 24th Annual IEEE Symposium on Logic In Computer Science*, pages 145–156, 2009. doi:10.1109/LICS.2009.27.
- 30 Oliver Friedmann. *Exponential Lower Bounds for Solving Infinitary Payoff Games and Linear Programs*. PhD thesis, LMU Munich, 2011. URL: <http://edoc.ub.uni-muenchen.de/13294/>.
- 31 Oliver Friedmann, Thomas Dueholm Hansen, and Uri Zwick. A subexponential lower bound for the random facet algorithm for parity games. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 202–216. SIAM, 2011. doi:10.1137/1.9781611973082.19.
- 32 Oliver Friedmann, Thomas Dueholm Hansen, and Uri Zwick. Subexponential lower bounds for randomized pivoting rules for the simplex algorithm. In *Proceedings of the forty-third annual ACM symposium on Theory of computing (STOC)*, pages 283–292. Association for Computing Machinery, 2011. doi:10.1145/1993636.1993675.
- 33 Oliver Friedmann, Thomas Dueholm Hansen, and Uri Zwick. Errata for: A subexponential lower bound for the Random Facet algorithm for Parity Games, 2014. doi:10.48550/arXiv.1410.7871.
- 34 Bernd Gärtner and Ingo Schurr. Linear programming and unique sink orientations. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 749–757, 2006. URL: <https://dl.acm.org/doi/abs/10.5555/1109557.1109639>.
- 35 Donald Goldfarb and William Y. Sit. Worst case behavior of the steepest edge simplex method. *Discrete Applied Mathematics*, 1(4):277–285, 1979. doi:10.1016/0166-218X(79)90004-0.
- 36 Thomas Dueholm Hansen. *Worst-case analysis of strategy iteration and the simplex method*. PhD thesis, Aarhus Universitet, 2012. URL: <https://pure.au.dk/portal/en/publications/worst-case-analysis-of-strategy-iteration-and-the-simplex-method>.
- 37 Sophie Huiberts, Yin Tat Lee, and Xinzhi Zhang. Upper and lower bounds on the smoothed complexity of the simplex method. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1904–1917, 2023. doi:10.1145/3564246.3585124.

- 38 Robert G. Jeroslow. The simplex algorithm with the pivot rule of maximizing criterion improvement. *Discrete Mathematics*, 4(4):367–377, 1973. doi:10.1016/0012-365X(73)90171-4.
- 39 Volker Kaibel and Kirill Kukharenko. Rock extensions with linear diameters. *SIAM Journal on Discrete Mathematics*, 38(4):2982–3003, 2024. doi:10.1137/23M1585878.
- 40 Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing (STOC)*, pages 302–311, 1984. doi:10.1145/800057.808695.
- 41 Leonid G. Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980. doi:10.1016/0041-5553(80)90061-0.
- 42 Victor Klee and George J. Minty. How good is the simplex algorithm? *Inequalities*, 1972. URL: <https://archive.org/details/inequalities0000oved/page/n5/mode/2up>.
- 43 Matthew Maat. Instances with exponential running time for strategy iteration. Master’s thesis, University of Twente, 2022. URL: <https://purl.utwente.nl/essays/92502>.
- 44 Matthew Maat. Strategy improvement, the simplex algorithm and lopsidedness, 2025. doi:10.48550/arXiv.2509.16075.
- 45 Mary Melekopoglou and Anne Condon. On the complexity of the policy improvement algorithm for markov decision processes. *ORSA Journal on Computing*, 6(2):188–192, 1994. doi:10.1287/ijoc.6.2.188.
- 46 Katta G Murty. Computational complexity of parametric linear programming. *Mathematical programming*, 19(1):213–219, 1980. doi:10.1007/BF01581642.
- 47 Martin L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 1994. doi:10.1016/S0927-0507(05)80172-0.
- 48 Francisco Santos. A counterexample to the Hirsch conjecture. *Ann. Math. (2)*, 176(1):383–412, 2012. doi:10.4007/annals.2012.176.1.7.
- 49 Sven Schewe. From Parity and Payoff Games to Linear Programming. In *Mathematical Foundations of Computer Science 2009*, Lecture Notes in Computer Science, pages 675–686, Berlin, Heidelberg, 2009. Springer. doi:10.1007/978-3-642-03816-7\_57.
- 50 Ingo Schurr and Tibor Szabó. Finding the sink takes some time: An almost quadratic lower bound for finding the sink of unique sink oriented cubes. *Discrete & Computational Geometry*, 31(4):627–642, 2004. doi:10.1007/s00454-003-0813-8.
- 51 Steve Smale. Mathematical problems for the next century. *The mathematical intelligencer*, 20(2):7–15, 1998. doi:10.1007/BF03025291.
- 52 Daniel Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing (STOC)*, pages 296–305, 2001. doi:10.1145/380752.380813.
- 53 Tibor Szabó and Emo Welzl. Unique sink orientations of cubes. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 547–555, 2001. doi:10.1109/SFCS.2001.959931.
- 54 Jens Vöge and Marcin Jurdziński. A discrete strategy improvement algorithm for solving parity games. In *International conference on computer aided verification (CAV)*, pages 202–215. Springer, 2000. doi:10.1007/10722167\_18.
- 55 Norman Zadeh. What is the worst case behavior of the simplex algorithm? Technical report, Stanford University, 1980. URL: <https://apps.dtic.mil/sti/html/tr/ADA089486/>.