# Exponential Lower Bounds for Solving Infinitary Payoff Games and Linear Programs

Oliver Friedmann

**Abstract**

*Parity games* form an intriguing family of infinitary payoff games whose solution is equivalent to the solution of important problems in automatic verification and automata theory. They also form a very natural subclass of *mean and discounted payoff games*, which in turn are very natural subclasses of turn-based *stochastic payoff games*. From a theoretical point of view, solving these games is one of the few problems that belong to the complexity class $\texttt{NP} \cap \texttt{coNP}$, and even more interestingly, solving has been shown to belong to $\texttt{UP} \cap \texttt{coUP}$, and also to PLS. It is a major open problem whether these game families can be solved in deterministic polynomial time.

*Policy iteration* is one of the most important algorithmic schemes for solving infinitary payoff games. It is parameterized by an *improvement rule* that determines how to proceed in the iteration from one policy to the next. It is a major open problem whether there is an improvement rule that results in a polynomial time algorithm for solving one of the considered game classes.

*Linear programming* is one of the most important computational problems studied by researchers in computer science, mathematics and operations research. Perhaps more articles and books are written about linear programming than on all other computational problems combined.

The *simplex* and the *dual-simplex* algorithms are among the most widely used algorithms for solving *linear programs* in practice. Simplex algorithms for solving linear programs are closely related to policy iteration algorithms. Like policy iteration, the simplex algorithm is parameterized by a *pivoting rule* that describes how to proceed from one basic feasible solution in the linear program to the next. It is a major open problem whether there is a pivoting rule that results in a (strongly) polynomial time algorithm for solving linear programs.

We contribute to both the policy iteration and the simplex algorithm by proving exponential lower bounds for several improvement resp. pivoting rules. For every considered improvement rule, we start by building 2-player *parity games* on which the respective policy iteration algorithm performs an exponential number of iterations. We then transform these 2-player games into 1-player *Markov decision processes* which correspond almost immediately to concrete linear programs on which the respective simplex algorithm requires the same number of iterations. Additionally,

we show how to transfer the lower bound results to more expressive game classes like payoff and turn-based stochastic games.

Particularly, we prove exponential lower bounds for the deterministic *switch all* and *switch best* improvement rules for solving games, for which no non-trivial lower bounds have been known since the introduction of Howard's policy iteration algorithm in 1960. Moreover, we prove exponential lower bounds for the two most natural and most studied randomized pivoting rules suggested to date, namely the *random facet* and *random edge* rules for solving games and linear programs, for which no non-trivial lower bounds have been known for several decades. Furthermore, we prove an exponential lower bound for the *switch half* randomized improvement rule for solving games, which is considered to be the most important multi-switching randomized rule. Finally, we prove an exponential lower bound for the most natural and famous history-based pivoting rule due to Zadeh for solving games and linear programs, which has been an open problem for thirty years.

Last but not least, we prove exponential lower bounds for two other classes of algorithms that solve parity games, namely for the *model checking algorithm* due to Stevens and Stirling and for the *recursive algorithm* by Zielonka.

# Synopsis

This thesis provides an overview of our results, presenting new lower bounds for algorithms that solve infinitary payoff games as well as new lower bounds for the simplex algorithm for solving linear programs. In particular, it summarizes the main results of the following papers:

(1) O. Friedmann. Recursive Algorithm for Parity Games requires Exponential Time. In *Theoretical Informatics and Applications, Cambridge Journals, 2011.*

(2) O. Friedmann. An Exponential Lower Bound for the latest Deterministic Strategy Iteration Algorithms. In *Logical Methods in Computer Science, Selected Papers of the Conference LICS 2009.*

(3) O. Friedmann, T. Hansen and U. Zwick. Subexponential Lower Bounds for Randomized Pivoting Rules for Solving Linear Programs. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC'11*, San Jose, CA, USA, 2011. Winner of the *Best Paper Award.*

(4) O. Friedmann. A Subexponential Lower Bound for Zadeh's Pivoting Rule for Solving Linear Programs and Games. In *Proceedings of the 15th Conference on Integer Programming and Combinatorial Optimization, IPCO'11*, New York, NY, USA, 2011. Awarded with *Zadeh's Prize.*

(5) O. Friedmann, T. Hansen and U. Zwick. A Subexponential Lower Bound for the Random Facet Algorithm for Parity Games. In *Proceedings of the*

*Symposium on Discrete Algorithms, SODA'11*, San Francisco, CA, USA, 2011.

(6) O. Friedmann. The Stevens-Stirling-Algorithm for Solving Parity Games Locally Requires Exponential Time. In *International Journal of Foundations of Computer Science, Volume 21, Issue 3*, 2010.

(7) O. Friedmann. An Exponential Lower Bound for the Parity Game Strategy Improvement Algorithm as we know it. In *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science, LICS'09*, Los Angeles, CA, USA, 2009. Winner of the *Kleene Award 2009*.

# Extended Abstract

In this thesis, we consider *infinitary payoff games*, which are important subjects of algorithmic game theory on the one hand, and have some of its applications in the domain of modal logic and automata theory on the other hand. Additionally, we consider *linear programming*, which is probably one of the most important fields in convex optimization.

We mainly investigate the most important algorithm that solves infinitary payoff games, namely the *policy iteration* method under a complexity theoretical point of view, analyzing its worst-case runtime. Similarly, we investigate the worst-case runtime of the *simplex method* for linear programs.

**Infinitary Payoff Games** We consider a variety of closely related classes of games in this thesis, which we like to call *infinitary payoff games*. These are zero-sum, perfect information games played by one or two players, and sometimes by an additional randomized player controlled by nature. The board is a directed, total graph, and each vertex of the graph belongs to one of the players.

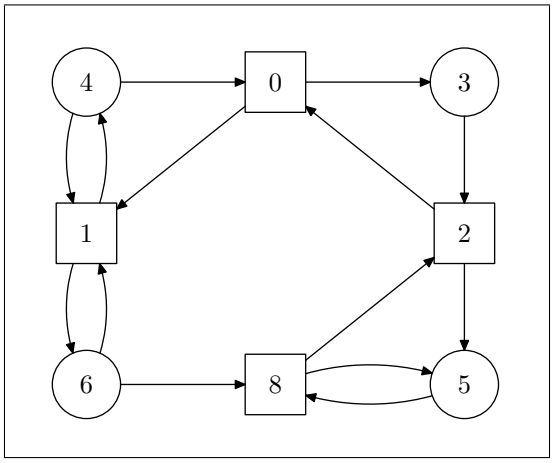
The game is played by putting a single token on one of the vertices (for instance, on a designated starting node), and moving it along an outgoing edge to a successor node. The player, to which the current node belongs to, decides, which outgoing edge to take. If the current node is owned by the randomized player, then one outgoing edge is picked arbitrarily at random. This process continues ad infinitum, yielding an infinite sequence of nodes. It now depends on the specific class of games to determine, which payoff the players receive or who wins the infinite play.

It is the objective of each player to maximize his or her payoff, or to win against the other player. A player's *strategy* in a game is a plan of action for whatever situation might arise when playing against any opponent. A strategy specifies for each node owned by the player, which respective successor node is to take, and in general, this can depend on the whole *history* of the play up to that stage. If a strategy does not depend on the history, we say that the strategy is *positional*.

All games that we consider are *positionally determined*, meaning that positional strategies suffice to answer the decision problems associated with the

games. This is convenient for many reasons, for instance as the number of positional strategies is finite if the game has finite size.

*Parity games* are infinitary payoff two-player games played on directed graphs with integer priorities assigned to their vertices. The two players, called *even* and *odd*, construct an infinite path in the game graph. Even wins, if the largest priority that appears an infinite number of times on the path is even. Odd wins otherwise. A parity game might look as follows (circle nodes are owned by the even player):




The problem of *solving* a parity game, i.e., determining which of the two players has a *winning strategy*, is known to be equivalent to the problem of $\mu$-calculus model checking [EJ91, EJS93, Sti95, GTW02]. It is also at the core of various problems in computer-aided verification, namely validity checking for branching-time logics [FL10, FLL10] and controller synthesis [VAW03].

Parity games form a very special subclass of *mean payoff games* [Pur95, EM79, GKK88, ZP96], which itself form a subclass of *discounted payoff games*, which in turn form a very special subclass of turn-based *stochastic games* [Con92, AM09]. More general *stochastic games* were previously considered by Shapley [Sha53].

Another extremely important class of infinitary payoff "games" are *Markov decision processes*, named after Andrey Markov, providing a mathematical model for sequential decision making under uncertainty. The study of Markov decision processes started with the seminal work of Bellman [Bel57]. It can be seen as a special subclass of turn-based stochastic games in which only one player is really used. Markov decision processes have many applications in practice, for instance in robotics, automated control, economics, operations research and artificial intelligence.

Parity and related, more expressive game classes like payoff and stochastic games, are a very interesting subject on their own from a complexity theoretical point of view. While it is known that the decision problems corresponding to these games belong to $\texttt{NP} \cap \texttt{coNP}$ [EJS93, Pur95], and even to $\texttt{UP} \cap \texttt{coUP}$ [Jur98,

ZP96], as well as to PLS [BM08], it is a major open problem whether any of these game families can be solved in polynomial time. Markov decision processes, on the other hand, *can* be solved in polynomial time by special techniques obtained from the domain of linear programming.

See Figure 1 for a summary of the most important relations. It is easy to see that having two players in a game raises the difficulty in obtaining polynomial-time decision procedures quite a lot.

A variety of algorithms for solving parity games has been invented so far. The most prominent deterministic ones are the recursive algorithm by Zielonka [Zie98], the local $\mu$-calculus model checker by Stevens and Stirling [SS98], Jurdziński's small progress measures algorithm [Jur00], the subexponential algorithm by Jurdziński, Paterson and Zwick [JPZ06] with a so-called big-step variant by Schewe [Sch07], as well as several variations of the policy iteration technique (see below), which is the only method that also applies to the other game classes.

This variety is owed to the theoretical challenge of answering the question whether parity (or any of these) games can be solved in polynomial time, rather than practical motivations. The currently best known upper bound on the deterministic solution of parity games is $\mathcal{O}(e \cdot n^{\frac{1}{3}p})$ due to Schewe's big-step algorithm [Sch07], where $e$ is the number of edges, $n$ is the number of nodes and $p$ is the number of different priorities in the game.

**Policy Iteration** The *strategy improvement*, *strategy iteration* or *policy iteration* technique is the most general approach that can be applied as a solving procedure for infinitary payoff games and related problems. It was introduced by Howard [How60] in 1960 for solving problems on Markov decision processes, and has been adapted by Hoffman and Karp in 1966 for solving nonterminating stochastic games [HK66]. Later, Condon adapted the algorithm for solving turn-based stochastic games [Con92], and Puri, Zwick and Paterson used the method to solve discounted and mean payoff games [Pur95, ZP96]. Finally, Jurdziński and Vöge formulated a *discrete* variant of the policy iteration algorithm for solving parity games [VJ00].

The beauty of the policy iteration technique lies in its simplicity. It is based on a (fixpoint-)iteration over a special finite subclass of strategies of the first player. In each iteration, the current strategy is mapped to a *valuation*. The valuation of a strategy allows us to decide *whether* the strategy is *optimal* for the first player, and if not, *how* we can improve the strategy to obtain a *better* one. An appealing feature is that we can compute valuations efficiently. In order to find an optimal strategy, which allows us to derive a solution for the game, we apply the following scheme, starting with an arbitrary strategy $\sigma$:

**Algorithm 1** Policy Iteration

```
1: while σ is not optimal do
2:     σ ← Improve(σ)
3: end while
```

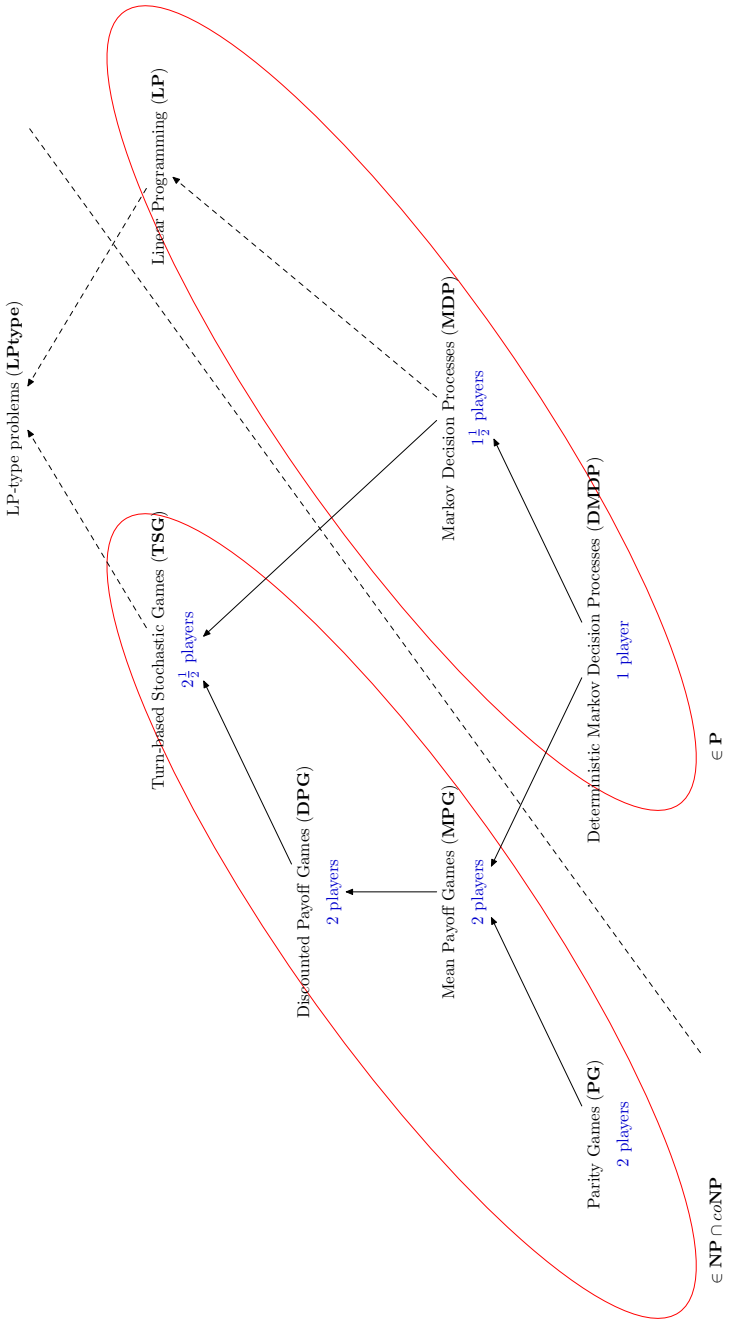



Figure 1: Reductions and Complexity

Policy iteration in fact describes a whole class of algorithms, as in general, there is more than one candidate strategy to proceed with in each iteration. The method of choosing successor policies is called *improvement rule*. Under the assumption that we only consider efficient improvement rules, it follows that the computational complexity of policy iteration essentially only depends on the number of iterations, since a single iteration is carried out in deterministic polynomial time.

This leaves us with the question whether every improvement rule leads to a small number of iterations. Not very surprisingly, this is not the case. An example has been known for some time for which a sufficiently poor choice of a deterministic improvement rule causes an exponential number of iterations [BV07]. Then, we could ask whether it is even theoretically possible to obtain an improvement rule that results in a small number of iterations. Here, the answer is yes, and the easy proof is folklore. However, the proof does not reveal any insights on how to formulate such an improvement rule. Or putting it differently, the improvement rule that we could get from the proof is not efficiently computable itself.

A variety of improvement rules has been invented so far, which is probably owed to the theoretical challenge of finding an efficient policy iteration algorithm. Generally, there are *deterministic*, *randomized* and *memorizing* improvement rules. The most important ones, that are mentioned in the literature, are the deterministic Switch-All [VJ00] and Switch-Best [Sch08], the randomized Switch-Half [MS99], Random-Facet [Kal92, Kal97, MSW96] and Random-Edge (folklore), and the memorizing Least-Entered [Zad80] rules. No non-trivial lower bounds on the worst-case complexity of all of these rules have been known until now.

As we will see, policy iteration is moreover closely related to an algorithm called *simplex method* for solving linear programs.

**Linear Programming** Linear programming is one of the most important fields of optimization theory, and is very actively researched. Many economical and practical tasks can be expressed as a linear programming instance, and several subgoals of other optimization problems in computer science require linear programs to be solved.

The linear programming problem is to maximize (or minimize) a given *linear objective function* while satisfying some additional linear equalities and inequalities. Formally, a linear programming problem is to *maximize* an objective function

$$c_1 x_1 + \ldots + c_n x_n$$

*subject to* a number of linear (in)equalities, called *constraints*,

$$\begin{aligned} a_{1,1}x_1 + \ldots + a_{1,n}x_n &= b_1 \\ a_{2,1}x_1 + \ldots + a_{2,n}x_n &= b_2 \\ &\vdots \\ a_{m,1}x_1 + \ldots + a_{m,n}x_n &= b_m \end{aligned}$$

where all *coefficients* $c_i$, all $b_i$, as well as all coefficients $a_{i,j}$ are real numbers.

We will see that Markov decision processes can be formulated as linear programs, which, in fact, is also the reason why we can solve this class of games in polynomial time.

There are, essentially, three kinds of algorithms that solve linear programs. First, there is the *simplex algorithm*, which has been proposed by Dantzig [Dan63] in 1963. The exact complexity of this algorithm is unknown, and it is a major open problem to answer this question adequately. The other two algorithms for solving linear programs, namely the *ellipsoid method* by Khachiyan [Kha79] and the *interior-point method* by N. Karmarkar [Kar84], handle linear programs in polynomial time, however not in *strongly* polynomial time.

The difference between *strongly* and "normal" or *weakly* polynomial time is subtle, and it depends on the method that we apply for *measuring* the size of a given linear program. Clearly, the number of variables and the number of constraints should contribute *linearly* to the size. But how do we measure the coefficients? Classically, their magnitude and precession would also contribute to the size, as we would need to find an *encoding* of the involved coefficients. If the runtime of an algorithm can be polynomially bounded in terms of this measure, we would speak of a weakly polynomial-time algorithm. On the other hand, if the runtime can be polynomially bounded by a measure that *only* depends on the number of variables and constraints, we would speak of strongly polynomial-time algorithm.

Although it is not even clear, whether (a variant of) the simplex method is a polynomial-time algorithm, it has the potential to be even a strongly polynomial-time algorithm, which is the reason why many people still consider this method.

**Simplex Algorithm** The *simplex algorithm* is based on the observation that the space of points satisfying all constraints essentially is a convex polytope (disregarding some special cases), and the objective function has an optimal value on one of its vertices. Hence, the idea is to start with an arbitrary vertex, to check whether the objective function is optimal on that vertex, and if not, to improve to some adjacent vertex.

Similar to the policy iteration algorithm, the simplex method is parameterized by a *pivoting rule* that selects one of the eligible neighboring vertices. Again, the complexity of the simplex algorithm essentially only depends on the number of *pivoting steps*, i.e. on the number of visited vertices, since all other necessary operations can be performed in (strongly) polynomial time.

The question whether every pivoting rule results in a small number of pivoting steps has been refuted by Klee and Minty [KM72], shortly after Dantzig presented the simplex algorithm. But unlike policy iteration, it is a major open problem, whether it is even theoretically possible to have a small number of pivoting steps. This is known as the *Hirsch conjecture* (see e.g. [Dan63], pp. 160,168).

Many of the improvement rules for policy iteration can be phrased as pivoting rules for the simplex algorithm (and vice versa), as most of them are formulated abstractly enough. As for policy iteration, no non-trivial lower bounds on the worst-case complexity of many of these rules in the context of the simplex algorithm have been known until now.

This leaves us with the question whether there is a deeper connection between the policy iteration algorithm for infinitary payoff games and the simplex algorithm for linear programs. Indeed, we will see that Markov decision processes are the "missing link" that relates policy iteration and the simplex method in some meaningful way.

**Contribution of Thesis** We present *exponential* (i.e. of the form $2^{\Omega(n)}$) and *subexponential* (i.e. of the form $2^{\Omega(n^c)}$ for some $0<c<1$) lower bounds for essentially all policy iteration improvement rules and all simplex method pivoting rules with unresolved complexity status.

First, we consider parity game policy iteration, and construct explicit families of parity games on which the different improvement rules require exponential resp. subexponential time.

We give exponential lower bounds for the deterministic Switch-All and Switch-Best rules for solving infinitary payoff games, subexponential lower bounds for the randomized Random-Edge and Random-Facet rules for solving games and linear programs, a subexponential lower bound for the randomized Switch-Half rule for solving games, and finally a subexponential lower bound for Zadeh's Least-Entered rule for solving games and linear programs.

All these problems have been open for several decades. It was hoped until now, that any of those would solve parity games in polynomial time.

Second, we show that all lower bound results obtained for parity game policy iteration can be transferred to more expressive game classes like mean and discounted payoff games, as well as turn-based stochastic games.

Third, we describe how our parity game constructions can be reshaped as Markov decision processes with the same implications for the policy iteration algorithm. This is not known to be possible in general, but we are able to at least translate our constructions.

Fourth, we formalize the relation between policy iteration for Markov decision processes and the simplex algorithm for induced linear programs, which allows us to transfer all applicable lower bound results to the domain of linear programming, solving problems that have been open for several decades. Note, however, that these results do not have any implications on the Hirsch conjecture.

Fifth, we show exponential lower bounds for the model checking algorithm due to Stevens and Stirling and for the recursive algorithm by Zielonka for solving parity games.

**Used Techniques** All lower bound constructions for policy iteration and the simplex algorithm are based on the following steps:

1. We construct a family of parity games that provides a lower bound for the respective improvement rule. We restrict ourselves in the construction to a very special form of parity games, called *sink parity games*.

   We think of parity game strategy iteration as a deterministic (due to the deterministic nature of both players) computational model in which we can implement different functional structures. All lower bound constructions are based on the implementation of a variant of a binary counter.

   The main gadget that is used to enforce exponentially many iterations in every lower bound construction, exploits intermediate cycles, and is called the *simple cycle gadget*, see Figure 2.

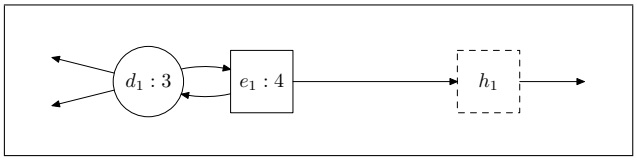



Figure 2: Simple Cycle

   Proving the constructions correct is then equal to showing that the sequence of strategies simulates the binary counter, or at least counts "good enough" with high probability when applying randomized pivoting rules.

   See Figure 4 for the full construction of 3-bit counter implemented as a parity game, that enforces exponentially many iteration when solved using the default Switch-All improvement rule.

2. We transfer the lower bound result for parity games to more expressive game classes like mean payoff games, discounted payoff games, turn-based stochastic games and the like.

   We show in general that policy iteration on sink parity games behaves exactly like policy iteration on the more expressive game classes, when the sink parity game is reduced to them by the standard reductions. We prove this correspondence independently of the applied improvement rule.

3. We transfer the lower bound results to Markov decision processes. Unfortunately, there is no standard reduction from parity games to MDPs. Therefore, we show that the translation from parity games to Markov decision processes operates as desired for every construction once again.

The only structure that we need to translate, when moving from parity games to Markov decision processes, are player 1 controlled nodes. In our lower bound games, the only role that player 1 has, is to hide the effect of some escape node.

This effect can be simulated by a randomization node that moves to the escape node with extremely (that is, inversely exponentially) low probability $\varepsilon$. See Figure 3 for a more complicated cycle setting and the correspondence between player 1 controlled cycles in parity games and randomization controlled cycles in MDPs.

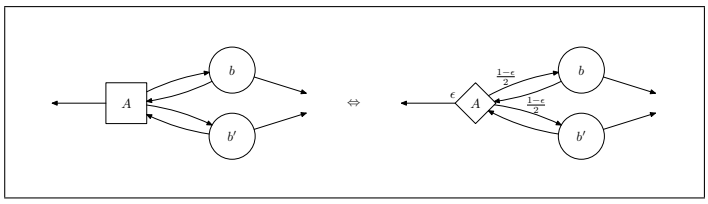



Figure 3: From Parity Games to MDPs

This technique was inspired by Fearnley's work [Fea10] and is joint work with Thomas Dueholm Hansen and Uri Zwick.

4. We transfer our lower bounds to the simplex algorithm for solving linear programs. Here, we show in general that the the simplex algorithm on linear programs, that are induced by our lower bound Markov decision processes, behaves exactly the same as policy iteration on the original games.

   The conditions for optimal values (and potentials) in a Markov decision process can be formulated as a linear program in which variables correspond to values and constraints to edges, and vice versa by duality.

   In order to transfer the lower bounds for Markov decision processes to the simplex algorithm for solving linear programs, we simply need to show that (1) basic feasible solutions in the induced linear program correspond to strategies in the original game, and that (2) adjacent basic feasible solutions with improved cost correspond to strategies that have been improved by a single improving switch. This allows us to transfer lower bounds for Markov decision processes immediately to the simplex algorithm, assuming that the respective improvement rule can be formulated accordingly as a pivoting rule.

   This technique is joint work with Thomas Dueholm Hansen and Uri Zwick.

**Future Directions** The most important question that obviously remains is whether parity games (or any of the other infinitary payoff two-player games) are solvable in polynomial time. Policy iteration still seems to be the most

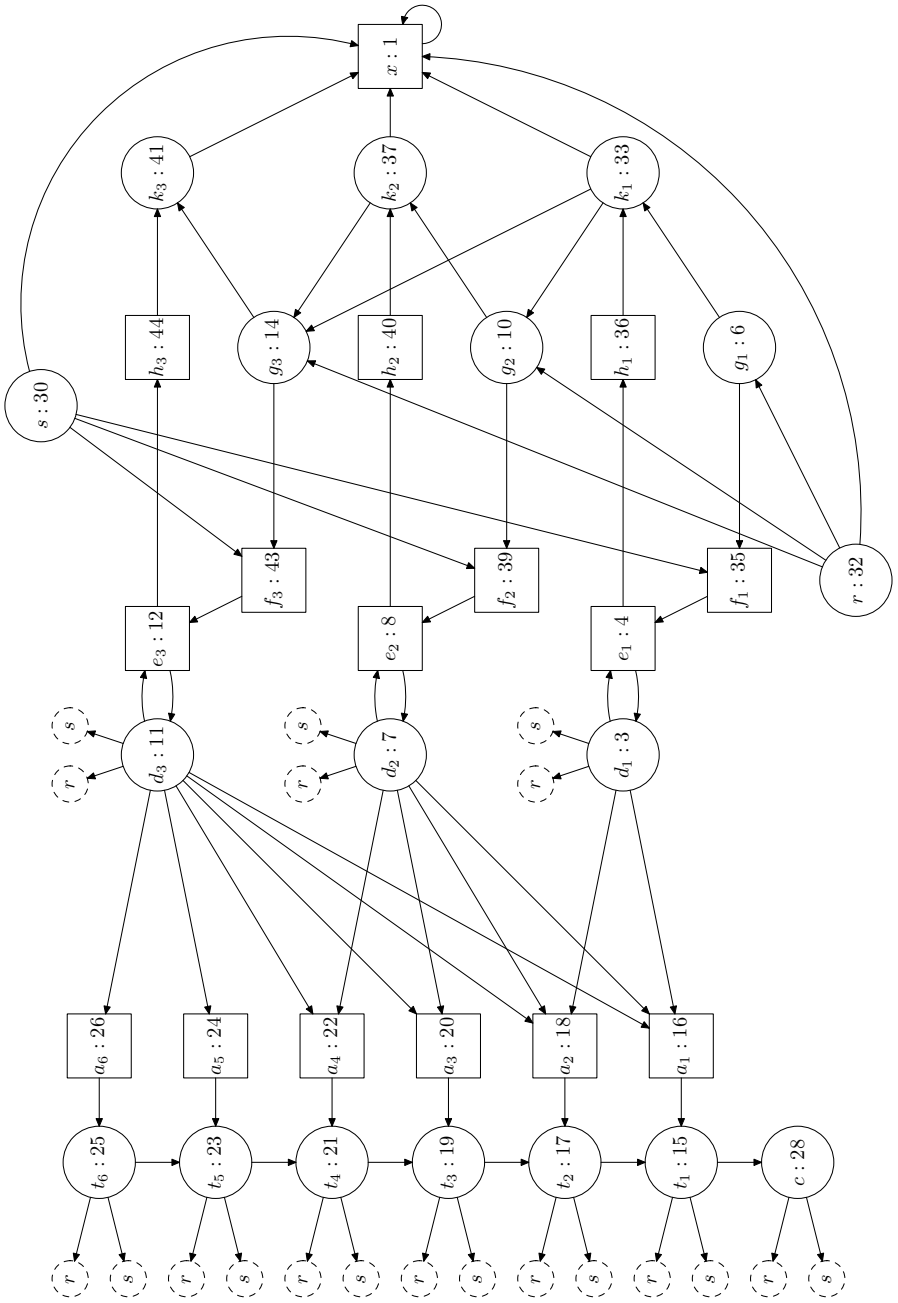



Figure 4: SWITCH-ALL Lower Bound Game $G_3$

promising candidate for giving rise to a polynomial-time procedure. However, it might be necessary to investigate non-standard improvement rules that go far beyond of what we use today.

Since parity games are the simplest class in this hierarchy of games, we think that this class should be the easiest to find a polynomial-time algorithm for. In fact, it is not too hard to show that many pivoting rules, including Switch-All, solve parity games in polynomial time, if player 1 does not appear in structures similar to the player 0 dominated cycles that we use in this thesis. We think that a rigorous analysis of these structures could help to design a pivoting rule that solves parity games efficiently.

As for the computational complexity of infinitary two-player games, it would be interesting to see whether there are deeper connections between solving the games and other NP-problems, that are neither known to be in P, nor known to be NP-complete, like the graph isomorphism problem or the factorization problem of natural numbers.

In the domain of linear programming, the most important open problems are, perhaps, whether linear programs can be solved in strongly polynomial time, whether any of the many variants of the polynomial Hirsch conjecture holds, and whether there is a polynomial-time admitting pivoting rule.

We think that it would be promising to analyze whether it is possible to extend Markov decision processes slightly in such a way that we can still reduce them to linear programming, but without being able to show that their diameter is small. This would allow us to construct a counter example to the Hirsch conjecture in the domain of games, which has been proven to be a very helpful abstraction when constructing concrete linear programs.

# References


[AM09] D. Andersson and P. B. Miltersen. The complexity of solving stochastic games on graphs. In *ISAAC '09: Proceedings of the 20th International Symposium on Algorithms and Computation*, pages 112–121, Berlin, Heidelberg, 2009. Springer.

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

[BM08] A. Beckmann and F. Moller. On the complexity of parity games. In *Proceedings of the BCS Conference Visions of Computer Science*, 2008.

[BV07] H. Björklund and S. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. *Discrete Applied Mathematics*, 155(2):210–229, 2007.

[Con92] A. Condon. The complexity of stochastic games. *Information and Computation*, 96:203–224, 1992.

[Dan63] G. B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, 1963.

[EJ91] E. Emerson and C. Jutla. Tree automata, $\mu$-calculus and determinacy. In *Proceedings of the 32nd Symposium on Foundations of Computer Science*, pages 368–377, San Juan, 1991. IEEE.

[EJS93] E. Emerson, C. Jutla, and A. Sistla. On model-checking for fragments of $\mu$-calculus. In *Proceedings of the 5th Conference on CAV, CAV'93*, volume 697 of *LNCS*, pages 385–396. Springer, 1993.

[EM79] A. Ehrenfeucht and J. Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8:109–113, 1979.

[Fea10] J. Fearnley. Exponential lower bounds for policy iteration. *CoRR*, abs/1003.3418, 2010.

[FL10] O. Friedmann and M. Lange. A solver for modal fixpoint logics. *Electronic Notes Theoretical Computer Science*, 262:99–111, May 2010.

[FLL10] O. Friedmann, M. Latte, and M. Lange. A decision procedure for CTL$^*$ based on tableaux and automata. In *IJCAR*, pages 331–345, 2010.

[GKK88] V. A. Gurvich, A. V. Karzanov, and L. G. Khachiyan. Cyclic games and an algorithm to find minimax cycle means in directed graphs. *USSR Computational Mathematics and Mathematical Physics*, 28:85–91, 1988.

[GTW02] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games*, LNCS. Springer, 2002.

[HK66] A. J. Hofmann and R. M. Karp. On nonterminating stochastic games. *Management Science*, 12(5):359–370, 1966.

[How60] R. Howard. *Dynamic Programming and Markov Processes.* The M.I.T. Press, 1960.

[JPZ06] M. Jurdziński, M. Paterson, and U. Zwick. A deterministic subexponential algorithm for solving parity games. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithm, SODA'06*, pages 117–123. ACM, 2006.

[Jur98] M. Jurdziński. Deciding the winner in parity games is in $\mathtt{UP} \cap \mathtt{coUP}$. *Information Processing Letters*, 68(3):119–124, 1998.

[Jur00] M. Jurdziński. Small progress measures for solving parity games. In H. Reichel and S. Tison, editors, *Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science, STACS'00*, volume 1770 of *LNCS*, pages 290–301. Springer, 2000.

[Kal92] G. Kalai. A subexponential randomized simplex algorithm (extended abstract). In *Proceedings of the 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. In *STOC '84: Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311, New York, NY, USA, 1984. ACM.

[Kha79] L. Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[KM72] V. Klee and G. L. Minty. How good is the simplex algorithm? *Inequalities*, III:159–179, 1972.

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

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

[Pur95] A. Puri. *Theory of Hybrid Systems and Discrete Event Systems*. PhD thesis, University of California, Berkeley, 1995.

[Sch07] S. Schewe. Solving parity games in big steps. In *Proceedings of the 27th International Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS'07*, volume 4855 of *LNCS*, pages 449–460. Springer, 2007.

[Sch08] S. Schewe. An optimal strategy improvement algorithm for solving parity and payoff games. In *Proceedings of the 17th Annual Conference on Computer Science Logic, CSL'08*, volume 5213 of *LNCS*, pages 369–384. Springer, 2008.

[Sha53] L. S. Shapley. Stochastic games. *Proceedings of National Academy of Sciences USA*, 39:1095–1100, 1953.

[SS98] P. Stevens and C. Stirling. Practical model-checking using games. In B. Steffen, editor, *Proceedings of the 4th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'98*, volume 1384 of *LNCS*, pages 85–101. Springer, 1998.

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

[VAW03] A. Vincent, A. Arnold, and I. Walukiewicz. Games for synthesis of controllers with partial observations. *Theoretical Computer Science*, 303(1):7–34, 2003.

[VJ00] J. Vöge and M. Jurdzinski. A discrete strategy improvement algorithm for solving parity games. In *Proceedings of the 12th International Conference on Computer Aided Verification, CAV'00*, volume 1855 of *LNCS*, pages 202–215. Springer, 2000.

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

[Zie98] W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *TCS*, 200(1–2):135–183, 1998.

[ZP96] U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1-2):343–359, 1996.