UPPSALA
UNIVERSITET

Digital Comprehensive Summaries of Uppsala Dissertations
from the Faculty of Science and Technology 3

Combinatorial Optimization for Infinite Games on Graphs

BY
HENRIK BJÖRKLUND

ACTA
UNIVERSITATIS
UPSALIENSIS
UPPSALA
2005

ISSN 1651-6214
ISBN 91-554-6129-8
urn:nbn:se:uu:diva-4751

Dissertation at Uppsala University to be publicly examined in Ångstömlaboratoriet, room 10132, Friday, February 18, 2005, at 14:15 for the Degree of Doctor of Philosophy. The examination will be conducted in English.

Abstract

Björklund, H. 2005. Combinatorial Optimization for Infinite Games on Graphs. Acta Universitatis Upsaliensis. *Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology* 3. vi, 51 pp. Uppsala. ISBN 91-554-6129-8

Games on graphs have become an indispensable tool in modern computer science. They provide powerful and expressive models for numerous phenomena and are extensively used in computer-aided verification, automata theory, logic, complexity theory, computational biology, etc.

The infinite games on finite graphs we study in this thesis have their primary applications in verification, but are also of fundamental importance from the complexity-theoretic point of view. They include parity, mean payoff, and simple stochastic games.

We focus on solving graph games by using iterative strategy improvement and methods from linear programming and combinatorial optimization. To this end we consider old strategy evaluation functions, construct new ones, and show how all of them, due to their structural similarities, fit into a unifying combinatorial framework. This allows us to employ randomized optimization methods from combinatorial linear programming to solve the games in expected subexponential time.

We introduce and study the concept of a controlled optimization problem, capturing the essential features of many graph games, and provide sufficient conditions for solvability of such problems in expected subexponential time.

The discrete strategy evaluation function for mean payoff games we derive from the new controlled longest-shortest path problem, leads to improvement algorithms that are considerably more efficient than the previously known ones, and also improves the efficiency of algorithms for parity games.

We also define the controlled linear programming problem, and show how the games are translated into this setting. Subclasses of the problem, more general than the games considered, are shown to belong to NP∩coNP, or even to be solvable by subexponential algorithms.

Finally, we take the first steps in investigating the fixed-parameter complexity of parity, Rabin, Streett, and Muller games.

Keywords: infinite games, combinatorial optimization, randomized algorithms, model checking, strategy evaluation functions, linear programming, iterative improvement, local search.

Henrik Björklund, Department of Information Technology. Uppsala University. Polacksbacken (Lägerhyddsvägen 2), Box 337, SE-751 05 Uppsala, Sweden

ISBN 91-554-6129-8
ISSN 1651-6214
urn:nbn:se:uu:diva-4751 (http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-4751)

To my parents

Composition of the Thesis

This thesis is composed of five published papers, preceded by an introductory survey. The first three chapters of the survey gives an informal description of the problems we investigate and our results, while the remainder summarizes the contributions of the appended and supporting papers, and puts them into the context of other research in the field. The survey is not intended to be comprehensive, but rather to serve as a guide to the appended papers. A brief summary in Swedish concludes the survey.

List of Papers

This thesis includes the following papers, which are referred to in the text by their Roman numerals.

I Björklund, H., Sandberg, S., and Vorobyov, S., A discrete subexponential algorithm for parity games. In H. Alt and M. Habib, editors, *20th International Symposium on Theoretical Aspects of Computer Science, STACS 2003*, Lecture Notes in Computer Science 2607, pages 663-674, Springer-Verlag, 2003. © Springer. Full preliminary version available as Technical Report 2002-026, Information Technology, Uppsala University.

II Björklund, H., Sandberg, S., and Vorobyov, S., Complexity of model checking by iterative improvement: the pseudo-Boolean framework. In A. Zamulin, editor, *Andrei Ershov Fifth International Conference: "Perspectives of System Informatics"*, LNCS 2890, pp. 381-394, Springer-Verlag 2003. © Springer. Full preliminary version in item (9) in the list of supporting papers.

III Björklund, H., Sandberg, S., Vorobyov, S., Memoryless determinacy of parity and mean payoff games: a simple proof. In *Theoretical Computer Science*, Vol. 310, No. 1-3, pp 365-378, 2004. © Elsevier.

IV Björklund, H., Sandberg, S., Vorobyov, S., A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. In J. Fiala, et. al., editors, *29th International Symposium on Mathematical Foundations of Computer Science, MFCS 2004*, LNCS 3153, pp. 673-685, Springer-Verlag 2004. © Springer. Extended version. To appear in *Discrete Applied Mathematics*. Preliminary version available as Technical Report DIMACS-2004-05.

V Björklund, H., Nilsson, O., Svensson, O., and Vorobyov, S., The controlled linear programming problem. Technical Report DIMACS-2004-41, DIMACS: Center for Discrete Mathematics and Theoretical Computer Science. Rutgers University, NJ, USA. September 2004.

Supporting Papers

The thesis also relies on results presented in the following papers, not reprinted here. Most technical reports can be found on the report pages of the Department of Information Technology, Uppsala University, and DIMACS, Rutgers University, NJ, USA.

1. Björklund, H., Nilsson, O., Svensson, O., and Vorobyov, S., Controlled Linear Programming: Duality and Boundedness, Technical Report 2004-56, DIMACS: Center for Discrete Mathematics and Theoretical Computer Science. Rutgers University, NJ, USA. December 2004.
2. Björklund, H., Sandberg, S., and Vorobyov, S., Randomized subexponential algorithms for infinite games. Technical Report 2004-09, DIMACS: Center for Discrete Mathematics and Theoretical Computer Science. Rutgers University, NJ, USA. April 2004.
3. Björklund, H. and Sandberg, S., Algorithms for combinatorial optimization and games adapted from linear programming. In B. ten Cate, editor, *Proceedings of the Eighth European Summer School on Logic, Language, and Information (ESSLLI) Student Session*, pp. 13-24, 2003.
4. Björklund, H., Sandberg, S., and Vorobyov, S. On fixed-parameter complexity of infinite games. Abstract in Nordic Workshop on Programming Theory 2003. Åbo Akademi, Dept. Computer Science, pp. 62-62, 2003. Full version in Technical Report 2003-038, Information Technology, Uppsala University, August 2003.
5. Björklund, H., Sandberg, S., and Vorobyov, S., Randomized subexponential algorithms for parity games. Technical Report 2003-019, Information Technology, Uppsala University, April 2003.
6. Björklund, H., Sandberg, S., and Vorobyov, S., An improved subexponential algorithm for parity games. Technical Report 2003-017, Information Technology, Uppsala University, March 2003.
7. Björklund, H., Sandberg, S., and Vorobyov, S., On combinatorial structure and algorithms for parity games. Technical Report 2003-002, Information Technology, Uppsala University, January 2003.
8. Björklund, H., Sandberg, S., and Vorobyov, S., An experimental study of algorithms for completely unimodal optimization. Technical Report 2002-030, Department of Information Technology, Uppsala University. October 2002.
9. Björklund, H., Sandberg, S., and Vorobyov, S., Optimization on completely unimodal hypercubes. Technical Report 2002-018, Information Technology, Uppsala University, May 2002.
10. Björklund, H. and Vorobyov, S., Two adversary lower bounds for parity games. Technical Report 2002-008, Department of Information Technology, Uppsala University. February 2002.

11. Björklund, H., Petersson, V., and Vorobyov, S., Experiments with iterative improvement algorithms on completely unimodal hypercubes. Research Report MPI-I-2001-2-003, Max-Planck-Institut für Informatik, Saarbrücken, Germany. June 2001.

Other Refereed Publications

1. Björklund, H., State Verification. In M. Broy et al, editors, *Model Based Testing of Reactive Systems*, Lecture Notes in Computer Science, to appear, 2004.

Contents

1 Motivation

Game theory, in its broadest sense, is almost as old as civilization itself. During the European antiquity and middle ages, mathematics were often taught in the form of entertaining games. Much later, the German philosopher and mathematician Gottfried Wilhelm Leibniz (1646-1716) realized the need to create a mathematical theory of games, a project that was to be continued by other mathematicians, such as John von Neumann (1903-1957) and John Nash (1928-). During the 20th century, game theory grew into a large and rich field of study, spanning over diverse academic disciplines such as philosophy, economics, sociology, operations research, biology, and mathematics.

In computer science, game theory has found a large number of applications, and many great results have been achieved. Games are used as models for computer systems, logics, automata, and complexity classes.

One of the greatest challenges for computer science today is the growing need for verification of large hardware and software systems. Our society relies more and more heavily on the correct functioning of computers and their programs. At the same time, the systems become ever bigger and more complex. Today, it is all but impossible for programmers to manually verify the correctness of their code, and time will only make it more difficult. This is why the need for automated verification methods increase dramatically. The basic idea is that given a system, we want to be able to check that it has a certain property. For a software driver controlling a printer, such a property may be that it never deadlocks, and always returns to a state where it is ready to handle a new request. In a larger system, involving many different components, the properties to be checked can be considerably more complicated.

In principle, we already know how to verify the correctness of most systems. The problem is how to do it *efficiently*. As systems grow, the issue of combinatorial explosion becomes troublesome. For most modern systems, it is completely infeasible to explicitly check each and every possible system behavior. Overcoming this obstacle is the main focus of a vast part of today's computer science research.

One possible approach is to describe the verification problem as a game. Infinite two-person adversary full information games provide a well established framework for modeling interaction between a system and its environment. A correct system can be interpreted as a player who has a winning strategy

against any strategy of the malicious environment. In the same way, a verification process can be considered as a proof that a system does possess such a strategy. If the system loses, a winning strategy for the environment can hint at necessary system improvements. During the last decades, substantial progress has been made both on fitting diverse approaches to computer-aided verification into the game-theoretic paradigm and, simultaneously, on developing efficient algorithms for solving games, i.e., determining the winner and its strategy; see [21, 10, 27, 34, 52] and Paper I. In a program paper [44], Papadimitriou stated that the complexity of finding Nash equilibria in a general version of such games is, together with factoring, the most important open problem on the boundary of P.

Casting the problems in a game-theoretic setting has the benefit of simplification. Everything is reduced to a two-player game with simple rules, which we can study. The theory gives a useful characterization of the optimal behaviors of rational players, the so-called *Nash equilibria*. The remaining question is, for each game type, whether it is computationally feasible to find Nash equilibria, and specifically, if they can be found in polynomial time. If we conclude that this is not the case, then the original problem is also too hard, and we need to look at other approaches. If, on the other hand, the game problem is efficiently solvable, we can try to extend the algorithms to work for the original problem, with all involved details.

In model checking, we are given a model of a system and a formula in some logic, and the question is whether the formula is true in the model. One of the games we study, parity games, is polynomial time equivalent to model checking for the modal μ-calculus. This logic is very expressive, subsuming most of the commonly used temporal logics, such as LTL, CTL, CTL*, etc. This means that an efficient algorithm for solving parity games would also allow efficient model checking for a great number of properties. Unfortunately, the computational complexity of solving parity games remains unknown. We only know that the problem belongs to the class NP$\cap$co-NP, but not whether it is actually in P.

The same is true for other problems we study as well, including mean payoff and simple stochastic games, which can be used to model long-term average benefits and systems with random choices, respectively. The NP$\cap$co-NP membership gives hope that they may be efficiently solvable, and since the problems are very useful, it is important to improve the known algorithms.

For other games, such as Rabin and Streett games, it is already known that they are complete for presumably intractable complexity classes. However, we might still want to solve special instances of these problems. This makes it interesting to look at the complexity from other angles, to try to determine which instances are actually solvable. One such possibility is to investigate the so-called fixed parameter complexity.

An approach to solving games on graphs that has not been thoroughly investigated previously is using combinatorial optimization. It is a large and well-studied topic in computer science, and provides a rich toolbox of efficient randomized algorithms and analytic tools that have successfully resolved a wide spectrum of challenging problems. It seems natural to try to fit graph games into the frameworks of combinatorial optimization, and apply known techniques to attack and solve game-theoretic problems arising in verification, but the full potential of this approach has yet to be determined. This thesis presents a series of novel results that show how techniques from combinatorial linear programming can be applied successfully to creating better algorithms for infinite games on graphs.

Many algorithms stated explicitly for a specific game become easier to analyze when stated in straightforward combinatorial terms, avoiding details that are not essential. Formulating game-theoretic verification problems as general combinatorial problems helps us understand their basic structure, and makes it easier to design new algorithms for them.

2 Our Contributions

The contributions of this thesis can be seen as a small part of the large effort within computer science towards automated verification, outlined in Chapter 1. It also has relevance to automata and complexity theory.

Our main theme is strategy improvement algorithms for infinite duration games played on graphs. The basic idea behind this approach is to assign values to the strategies of one of the players, and then search the strategy space guided by these values. The objective is to find the strategy that has been assigned the largest value. Such a scheme for assigning values is called a *strategy evaluation function*, and is mainly applicable to games with *memoryless determinacy* (see Chapter 6 and Paper III), when the strategy space we need to consider is finite. They include parity, mean payoff, discounted payoff, and simple stochastic games.

Apart from a strategy evaluation function, a strategy improvement algorithm consists of a search policy, telling the algorithm how to proceed from strategy to strategy until the one with the best value is found. Efficiency depends crucially on this policy.

In this work we are concerned with both parts of strategy improvement algorithms. We refine and invent new strategy evaluation functions and improvement policies in order to speed up the calculations performed in each iteration and also get better overall complexity analysis. By analyzing the combinatorial structure of the functions, we are able to show how randomization methods from combinatorial optimization can be used to provide new, more efficient algorithms for solving games.

Our work on developing strategy evaluation functions has two parts. For parity games, Vöge and Jurdziński invented the first discrete evaluation function. We refine their method, thereby limiting the maximal number of improvement steps for games with fewer colors than vertices. (For definitions of the games; see Chapter 4.) The modification is described in detail in Paper I. For mean payoff games, we develop the first discrete strategy evaluation function. It is based on the longest-shortest paths problem, a new, controlled version of shortest paths. This allows improvement algorithms to avoid costly high-precision computations with rational numbers in each iteration. In a combinatorial model of computation, this makes the complexity bounds independent of the edge weights in the game graph. Also, it greatly improves

practical efficiency. The function is described in Paper IV.

In analyzing the combinatorial structure of strategy evaluation functions, we characterize all considered functions as being *recursively local-global*, and in the case of parity and simple stochastic games even *completely local-global*. For definitions of these classes, see Chapter 8 and Paper II. They have a number of beneficial features, closely related to the well studied *completely unimodal* functions [30, 54, 51, 8]. The characterization allows for an improved analysis of some algorithms (Paper II). It also provides an abstract framework for future investigations of improvement algorithms. Furthermore, we can show that all the studied evaluation functions fit into the framework of *LP-type problems* [48]. This implies that any algorithm for solving this general class of combinatorial problems can be reused for games. Considerations of this kind allowed us to develop the first randomized subexponential algorithms for parity games (Paper I), mean payoff games (Paper IV), and simple stochastic games with arbitrary outdegree [6, 7]. All of this is discussed in Chapter 8.

In Chapter 5 and Paper V we develop another kind of unifying framework, the *controlled linear programming* (CLP) problem. This is a version of linear programming in which a controller is allowed to select and discard constraints according to simple rules. It provides a simple, unified view of parity, mean payoff, discounted payoff, and simple stochastic games, which can all be modeled as particular, restricted instances of the CLP problem. We show that many interesting subclasses of controlled linear programming belong to NP$\cap$coNP, and give algorithms for solving them, based on combinatorial optimization and strategy improvement. We also give characterizations of subexponentially solvable subclasses in terms of linear algebra.

In classical complexity, the border for feasibility is considered to coincide with that for P. Since it remains unknown on which side of this border the games we study belong, it makes sense to consider other aspects of their complexity, in order to get a better understanding. In Chapter 9 we consider the *fixed-parameter complexity* of parity games. By combining known reductions, we come to the interesting conclusion that under the most natural parameterizations, they belong to the same complexity class as Rabin, Streett, and Muller games. This is a collision with classical complexity, since Rabin and Streett games are complete for NP respectively coNP, and raises a number of question regarding the common features of the games and their relations to complexity classes. The exact fixed-parameter complexity of the games remains unknown, but we show that if both players in a Streett game are restricted to using only positional strategies, the problem becomes complete for the presumably intractable class W[1].

In Chapter 6 and Paper III we give a proof of the fact that parity and mean payoff games are determined in positional strategies. This is by no means a

new result. On the contrary, it is well known and can be proved in a number of ways [19, 22, 42, 41, 55, 26]. Our motivation was to investigate if a completely constructive proof could be given, without referring to any nonelementary methods, fixed-point theorems, or limit arguments, while at the same time working for both games in a uniform way. We also hope that it contributes, together with the other proofs, to a better understanding of the inner workings of the problems.

3 Games and Combinatorial Optimization

Combinatorial optimization is all about finding a needle in a haystack. In other words, we are given a finite, but usually very large, collection of objects, and want to find an element that is in some sense optimal. In most cases, there is a function from the collection to an ordered set, and we want to find an object that maximizes the function value. Typically, the collection has a succinct representation, and the actual number of objects is exponential in the representation size. Therefore, checking all objects one by one and simply picking the best is not feasible. This chapter describes what combinatorial optimization has to do with solving games.

The games we study are not the kind you would pick out of a drawer at night to play with friends. Rather, they were invented as models of other phenomena, and the players are abstract thought-constructs. What we study is how the games would end if we assume that the players are perfect, and always use optimal strategies. Algorithms that answer this question are said to *solve* the games.

Many of the games we will discuss can actually be viewed as being played by only one player. There is some goal that she wants to achieve, and the question is whether there is a strategy that allows her to do this. Let us consider an example. Suppose we have a directed acyclic graph G with weighted edges, distinguished source s and target t, and a pebble placed on s. Now we can imagine a player, who is allowed to move the pebble along edges of the graph, with the goal of reaching t while keeping the total weight of traversed edges smaller than some number k.

The problem the player faces is clearly nothing else than the shortest path problem for acyclic graphs, a well known optimization problem solvable in polynomial time. This gives us a simple way of determining whether the player has a strategy for getting to t with cost smaller than k, even though there may be exponentially many paths from s to t. Simply apply a known shortest-path algorithm. We will repeatedly encounter this kind of one-player game, equivalent to some polynomial time solvable optimization problem. They are primarily used as a help for solving more complicated games, where an opponent is involved.

To continue our example, suppose we add another player. We call the original player MIN and the new one MAX. We also divide the vertices of the

graph into two sets, V_{MAX} and V_{MIN}. The goal of MIN is still to reach t with cost smaller than k, but now, as soon as the pebble reaches a vertex in V_{MAX}, MAX selects the next edge to follow, and he tries to spoil the game for MIN. Given k, MAX wins if the sum of weights of edges that the pebble is moved along before reaching t is at least k.

Since the graph is acyclic, the pebble will never get to a vertex twice. Once the pebble reaches a vertex, MAX tries to maximize the value of the remaining path to t, regardless of what has happened earlier. Therefore, it is enough for MAX to select one outgoing edge from each vertex in V_{MAX} in advance, deciding to only play along these edges. Such a selection σ is called a *strategy* for MAX. Now we can construct the subgraph of G corresponding to a game where MAX has decided to play according to σ. It is obtained by removing all edges leaving vertices in V_{MAX} except those chosen by σ, and is called G_σ. Now, by solving the shortest path problem on G_σ, we can answer the following question: Assuming that MAX uses σ, what is the smallest cost MIN can achieve for reaching t? In this way, a specific number is associated with σ, corresponding to the outcome of the game when MAX uses σ. For every other strategy of MAX, a cost can also be computed in the same way. This give us a function from the strategies of MAX to an ordered set. The function reflects the relative quality of strategies, and is therefore called a *strategy evaluation function*. We now have a scheme for computing the outcome of the game, assuming that both players play optimally. For every strategy σ of MAX, compute the shortest path from s to t in G_σ, and return the maximum cost over all strategies.

Unfortunately, the number of possible strategies of MAX is huge, exponential in the size of V_{MAX}, so computing the value for each one is infeasible for large graphs. In our current case, this is not a big concern, since the outcome of the game is easily computable by a bottom-up dynamic programming algorithm, after topologically sorting the vertices. Our simple game gives us an example of how a two-player game, where a player can fix a strategy in advance and the resulting one-player game is easy to solve, can be interpreted as a combinatorial optimization problem. Compared to the standard shortest paths problem, control has been given to MAX in some vertices, and we therefore call it the *controlled shortest paths* problem. For acyclic graphs, it is easy to solve, but as soon as cycles are possible, we get a much harder problem, for which the exact complexity is unknown. It is closely related to the so-called mean payoff games, and we study it in detail in Chapter 7 and Paper IV, deriving a way of assigning values to mean payoff game strategies.

Generally, when describing the games we study in combinatorial optimization terms, the collection of objects we optimize over is the set of strategies of MAX. This set can be exponentially large in the size of the graph used to represent the game, and thus cannot be searched exhaustively. The function

we optimize is a strategy evaluation function. In the game of our example, this function is computed for a strategy σ by constructing G_σ and solving the corresponding shortest-path problem. The strategy with the best function value should also be an optimal strategy. This is clear in the above example, but must be proved for more complicated games. More about this in Chapter 7.

4 Game Definitions

The games we study are played on finite, directed graphs. Detailed information about definitions, algorithms, and other interesting results can be found in, e.g., [19, 29, 22, 42, 12, 46, 56, 50, 55, 34, 52, 27], although this list is far from comprehensive. With the exception of simple stochastic games, the game graphs are leafless, and the games have infinite duration. There are two players, Player 0 and Player 1. (In games with quantitative objectives, we will often call them MAX and MIN instead.) In a game graph $G = (V,E)$, the vertices are partitioned into two sets, V_0 and V_1, corresponding to the two players. A pebble is placed on a start vertex v_0, and is moved by the players along edges of G. If the pebble is on a vertex in V_0, Player 0 selects the next edge to follow; otherwise Player 1 does. Again, simple stochastic games is the only exception. In this case the graph has two sinks where the game ends, and there are vertices belonging to neither player, where random choices are made instead of player decisions.

By moving the pebble, the players construct a sequence of vertices (or, equivalently, edges), called a *play*.

Definition 4.0.1 *A* strategy *for Player 0 is a function $\sigma : V^* \cdot V_0 \to V$, such that if $\sigma(v_0,\ldots,v_k) = u$, then $(v_k,u) \in E$. A play $v_0,v_1,\ldots$ is* consistent *with σ if $\sigma(v_0,\ldots,v_k) = v_{k+1}$ for all k such that $v_k \in V_0$. Strategies for Player 1 are defined symmetrically.*

Given two strategies, one for each player, there is a unique play consistent with both strategies, again with the exception of simple stochastic games. What differentiates the games are the objectives of the players. Here we give the necessary definitions for the games we will encounter most frequently in the sequel.

Definition 4.0.2 *In a* parity game *(PG), we are given a game graph $G = (V,E)$ and a* coloring $c : V \to \mathbb{N}$. *A play π is* winning *for Player 0 if the largest color of a vertex appearing infinitely often in π is* even. *Otherwise, π is winning for Player 1.*

Notice that parity games have *qualitative* objectives. Each player either wins or loses a play, there is no notion of winning more or less. A strategy σ of Player 0 is *winning* if all plays consistent with σ are winning for Player 0.

Definition 4.0.3 *In a* mean payoff game *(MPG), we are given a game graph* $G = (V,E)$ *and a* cost function $w : E \to \mathbb{Z}$. *The players are called* MAX *and* MIN. *If* $e_1e_2\ldots$ *is the sequence of edges in a play, then* MIN *pays* MAX *the amount* $\liminf_{k\to\infty} 1/k \cdot \sum_{i=1}^{k} w(e_i)$ *(if the value is negative,* MAX *actually pays* MIN*).*

Thus mean payoff games have *quantitative* objectives. The players can win more or less.

Definition 4.0.4 *A* discounted payoff game *(DPG) is a game graph together with a* cost function $w : E \to \mathbb{Z}$ *and a* discounting factor $\lambda \in (0,1)$. *If* $e_1e_2\ldots$ *is the sequence of edges in a play, then* MIN *pays* MAX $(1-\lambda)\cdot\sum_{i=1}^{\infty}\lambda^i \cdot w(e_i)$.

DPGs are similar to MPGs, but for the latter, any prefix of a play can be disregarded without affecting the outcome, while in a DPG, the first steps are the least discounted, and thus have the largest influence.

Simple stochastic games, as mentioned, differ in that the game graph has two sinks and probabilistic vertices belonging to neither player.

Definition 4.0.5 *In a* simple stochastic game *(SSG) the vertex set* V *is partitioned into the sets* $V_{\text{MAX}}, V_{\text{MIN}}, V_{\text{AVG}}, \{s_0\}$, *and* $\{s_1\}$, *where* s_0 *is the* 0-sink, s_1 *is the* 1-sink, *and* V_{AVG} *is the set of* average vertices. *For each* $v \in V_{\text{AVG}}$ *there is a probability distribution on all outgoing edges from* v. *Every time the pebble reaches* $v \in V_{\text{AVG}}$, *the next edge to follow is selected randomly, according to this distribution. The goal of* MAX *is to maximize the probability of reaching the 1-sink, while* MIN *tries to minimize this probability.*

All the games have associated decision problems. For parity games it is the question whether Player 0 has a winning strategy. For the quantitative games, the question is if MAX has a strategy that ensures payoff at least k, or in the case of simple stochastic games, that the probability of reaching the 1-sink is at least k, for some number k.

All these decision problems belong to the complexity class NP∩coNP; see, e.g., [22, 43, 29, 56, 11]. None of them is known to belong to P. This status is interesting. It implies that the problems are highly unlikely to be NP-complete, and it is not at all impossible that there are efficient algorithms for solving them. Very few natural problems have been shown to be in NP∩coNP without at the same time being in P. Up until recently, the PRIMES problem shared this status, but it has subsequently been shown to have a polynomial time algorithm [1]. There is a known chain of polynomial time reductions between the games: PG $\leq_p$ MPG $\leq_p$ DPG $\leq_p$ SSG; see, e.g., [56, 46]. It is not known whether any of the reductions can be reversed.

5 A General Game-Like Problem

As we saw in Chapter 3, some games can be described as combinatorial optimization problems, and vice versa. In this chapter we describe a game-like combinatorial optimization problem, more general than the games described in Chapter 4. In fact, it is general enough to model all of them. This problem was first defined and studied in Paper V. We describe it as a game, played by MAX and MIN.

The game consists of a system S of linear inequalities over the variables $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$, owned by MAX, and $\mathbf{y} = \{y_1, y_2, \ldots, y_m\}$, possessed by MIN. Every constraint has the form

$$\begin{aligned} x_i &\leq p_i^k(\mathbf{y}) + w_i^k, \text{ or} \\ y_j &\leq q_j^l(\mathbf{x}) + {w'}_j^l, \end{aligned}$$

where p_i^k and q_j^l are linear homogeneous polynomials with *nonnegative* coefficients, and $w_i^k, {w'}_j^l \in \mathbb{R}$, for $i \in \{1, \ldots, n\}$, $k \in \{1, \ldots, n_i\}$, $j \in \{1, \ldots, m\}$, $l \in \{1, \ldots, m_j\}$, $n, n_i, m, m_j \in \mathbb{N}^+$. For each variable $v \in \mathbf{x} \cup \mathbf{y}$, there is at least one constraint with v in the left-hand side.

The game is played as follows. First, MAX selects a set σ of n constraints, such that for each $x_i \in \mathbf{x}$, there is exactly one constraint in σ with x_i in the left-hand side. Then, MIN makes a similar selection τ of one constraint per variable in $\mathbf{y}$. This results in the linear system $S(\sigma, \tau)$, consisting only of the constraints in σ and τ. Next, an arbiter solves the linear program

$$\begin{aligned} &\textit{maximize } \textstyle\sum v, \ \ v \in \mathbf{x} \cup \mathbf{y} \\ &\textit{subject to } S(\sigma, \tau). \end{aligned}$$

If the result is a finite number c, then MIN pays the amount c to MAX (if c is negative, MAX will have to pay MIN). If the system is unbounded, MIN pays an infinite amount, while MAX loses infinitely much if the system is infeasible.

Once the players have selected their strategies, the computation of the outcome only involves solving a linear program, which Khachiyan proved can be done in polynomial time. Thus the interesting question is how difficult it is to compute the best strategies for the two players.

To investigate this, it is helpful to describe the game as a combinatorial

problem. This is achieved by noting that once MAX has decided on a strategy, the smallest value MIN can achieve by any strategy can be computed by considering the system $S(\sigma)$ consisting of all constraints with variables from $\mathbf{y}$ on the left-hand side, but only those from σ for variables from $\mathbf{x}$ [2]. We simply solve the linear program

$$\begin{aligned} &\textit{maximize } \sum v,\ v \in \mathbf{x} \cup \mathbf{y} \\ &\textit{subject to } S(\sigma). \end{aligned}$$

This means that as soon as MAX has selected a strategy, the outcome of the game can be computed in polynomial time. Thus we have a suitable strategy evaluation function, and are left with a typical combinatorial problem: given exponentially many possible strategies, each associated with a real number or $\pm\infty$, find the one with the largest value. We call this the *controlled linear programming* (CLP) problem, and study it in Paper V and [2].

If we allow coefficients in the polynomials defining constraints to be negative, the problem becomes NP-complete; see Paper V. However, with non-negative coefficients, things are much more interesting. We first invented the controlled linear programming problem as a generalization of the longest-shortest paths problem defined in Paper IV. Thus it can be used to model both parity and mean payoff games. In Paper V, we show that if the coefficients are restricted to be integral, the problem still belongs to NP∩coNP, even though this class appears to be considerably more general than mean payoff games. The NP∩coNP membership is shared by a number of other subclasses, including generalizations of discounted payoff and simple stochastic games, that can also be solved in randomized subexponential time, using methods from combinatorial linear programming.

We are interested in the CLP problem for a number of reasons. First, it gives us another unified view of strategy improvement, in addition to the concept of recursively local-global functions discussed in Chapter 8. Second, through the CLP problem, we can give uniform and easy proofs for some of the most important properties of strategy evaluation functions; see Chapter 7. Third, as the CLP problem is described in terms of linear algebra, the multitude of results from this field can be used for investigating game problems, a work we begin in Paper V and [2]. Fourth, the CLP problem leads to the interesting question of how much the games from Chapter 4 can be generalized while staying in NP∩coNP. This question is also given partial answers in Paper V and [2].

Given a CLP instance S, with rational coefficients and constants, and pair (σ, τ) of strategies, we get a system $S(\sigma, \tau)$ with exactly one constraint per variable. We can rewrite this system in matrix form as $Ax \leq b$ where A is a square matrix and b is a vector. The value of the strategy pair is $\max\{1^T x | Ax \leq$

$b\}$. In [2] we show that this is a finite number if and only if the system $Ax = 0$ has the 0 vector as its *unique* solution. This means that we can reformulate the problem of solving parity and mean payoff games in the following way. We are given a CLP problem. The question is whether there is a strategy σ of MAX such that for every strategy τ of MIN, the square matrix A obtained by writing $S(\sigma,\tau)$ in matrix form has $\{0\}$ as its kernel.

A CLP instance S is said to be *strongly bounded* if for every pair (σ,τ), the linear program

$$\begin{aligned} &\textit{maximize } \textstyle\sum v,\ v \in \mathbf{x} \cup \mathbf{y} \\ &\textit{subject to } S(\sigma,\tau). \end{aligned}$$

has a finite optimal solution. As can be seen from Paper V, the systems achieved by reduction from discounted payoff or simple stochastic games are strongly bounded. We show in [2] that all systems in this more general class can be optimized by subexponential algorithms and the corresponding decision problem belongs to NP∩coNP.

6 Memoryless Determinacy

What makes the methods for solving graph games discussed in Chapter 7 and 8 work, is the fact that parity, mean payoff, discounted payoff, and simple stochastic games all have the property known as *memoryless determinacy*. This means that for each vertex a player owns, he can decide before the game even starts what he will do if the play ever reaches the vertex, without jeopardizing the payoff. Memoryless determinacy allows us to focus only on *positional strategies*.

Definition 6.0.6 *A* positional strategy *for Player 0 is a strategy that depends only on the last vertex of the play so far, not on the whole history. In other words, it is a function $\sigma : V_0 \to V$, such that if $\sigma(u) = v$, then $(u,v) \in E$. Positional strategies for Player 1 are defined symmetrically.*

A qualitative graph game has memoryless determinacy if, for every instance, one of the players has a winning positional strategy. If a game has quantitative objectives and memoryless determinacy, every instance has an optimal value each player can achieve, and positional strategies guaranteeing these values.

Since parity games have memoryless determinacy, the vertices of any instance can be divided into two sets, W_0 and W_1, such that whenever the game starts from a vertex in W_0, Player 0 has a winning positional strategy, and otherwise Player 1 does. These sets are called the *winning sets* of the players. Furthermore, the players have *uniform* positional winning strategies from their whole winning sets. This means that whenever play starts in W_0, Player 0 can use the *same* positional strategy, regardless of which vertex in W_0 is the first.

It has been known for some time that the games we investigate have memoryless determinacy. Ehrenfeucht and Mycielski proved it for mean payoff games as early as in 1973 [18, 19]. Memoryless determinacy for parity games can be proved as a corollary of this result. The proof utilizes a sophisticated cyclic interplay between *infinite* duration games and their *finite* counterparts. Properties of the infinite games are shown by considering the finite games and vice versa. Ehrenfeucht and Mycielski raised the question whether it is possible to give a direct proof, avoiding this cyclic dependence. Paper III answer the question affirmatively. Later, Gurvich, Karzanov, and Khachiyan gave a

constructive proof [29]. It is rather involved, using estimates of norms of solutions to systems of linear equations, convergence to the limit, and precision arguments.

Emerson [20], unaware of Ehrenfeucht's and Mycielski's proof, sketched the first memoryless determinacy proof for parity games in 1985. His proof is based on a fairly complicated simplification by Hossley and Rackoff [32] of Rabin's original decidability proof [47] for Rabin automata. It relies on König's lemma. A later, more self-contained, determinacy proof by Emerson and Jutla [22] relies heavily on the μ-calculus, and is non-constructive. For example, the definition of a strategy in [22] uses properties of all paths in a binary tree, a set of continuum cardinality. Mostowski [42] independently proved the same result in 1991.

As mentioned, memoryless determinacy for parity games can also be proved as a simple corollary to the result for mean payoff games. A parity game with n vertices can be reduced to a mean payoff game on the same graph. If a vertex in the parity game has color k, all its outgoing edges are assigned weight $(-1)^k \cdot n^k$. It is easy to verify that MAX can get a nonnegative payoff in the mean payoff game if and only if Player 0 wins the parity game; see, e.g., [46].

Later McNaughton [41] proved memoryless determinacy for a subclass of Muller games, defined by the structure of the winning condition, and including parity games. The proof is constructive, and allowed McNaughton to give an exponential time algorithm for finding optimal strategies. In 1998 Zielonka gave two elegant proofs for parity games on possibly infinite graphs [55]. One of them is constructive; see also the survey by Küsters in [27]. Recently, Zielonka and Gimbert were able to identify a set of conditions on the payoff function which are sufficient for memoryless determinacy, and which are satisfied by both parity and mean payoff games [26].

The proof given in Paper III is simple, direct, and works uniformly for a number of games, including parity and mean payoff. The condition is that the game has an equivalent finite version, played until the first vertex repetition, and that the winner is determined by the sequence of vertices on the resulting loop, modulo cyclic permutations. It proceeds by elementary induction on the edges of the game graph, completely avoiding powerful external methods. Like the proof in [19], it utilizes finite duration versions of the games, but there is no cyclic dependence between properties of finite and infinite games. The proof is constructive, even though the algorithm it straightforwardly suggests is not very efficient.

7 Strategy Evaluation and Iterative Improvement

When solving games with memoryless determinacy, we can restrict our attention to the *finite* set of positional strategies for each player. One of the most important methods for finding the best such strategies is *iterative strategy improvement*. Originally developed for Markov decision processes [33, 13], this approach has been extensively used to solve games; see, e.g., [31, 12, 36, 46, 52] and Papers I, II, IV and V. The idea is to assign values to the positional strategies of one of the players. An initial strategy is selected and then iteratively improved by local changes, guided by the values.

The way values are assigned to strategies is crucial for strategy improvement algorithms to work. When seen as functions defined on the space of positional strategies, all strategy evaluations we consider, except the one for the CLP problem in its most general form, satisfy the property that local optima are global. Furthermore, every global optimum corresponds to a strategy that is "sufficiently good". In parity games, this means that it is winning from all vertices where this is possible. For all evaluation functions, the value of a strategy is a vector of values for the individual vertices of the game. This allows us to use the concepts of *attractive* single switches, and *stable* strategies. Roughly speaking, a switch is *attractive* if it selects a successor with a better value *with respect to the current strategy*. A strategy is *stable* if it has no attractive switches. The strategy evaluation functions we consider have the following two properties.

Profitability of attractive switches. Let σ be a strategy and v a vertex. If the value under σ of the successor $\sigma(v)$ of v is worse than the value of one of v's other successors w (the switch to this successor is *attractive*), then changing σ in v to w results in a better strategy (the switch is *profitable*). This property (attractive implies profitable) ensures 1) monotonicity and termination, and 2) that the algorithms can move to better strategies without actually evaluating the neighbors of the current one. In games such as mean payoff games, where the edges have weights, the impact of the edges used must also be considered; see Paper IV.

Optimality of stable strategies. If a strategy is *stable*, i.e., has no attractive switches, then it is globally optimal. Consequently, the algorithms can terminate, reporting a global optimum, as soon as a stable strategy is found, without

evaluating its neighbors.

Together, these two properties imply that we can let the iterative improvement be guided by attractiveness of switches. As long as there are attractive switches, we can make them, knowing that they are profitable. And if there are no attractive switches, we know that the current strategy is globally optimal. Relying on attractiveness, rather than investigating neighboring strategies explicitly, does not change the behavior of algorithms, only makes them more efficient.

In what follows, we will consider strategy evaluation functions for simple stochastic, discounted payoff, mean payoff, and parity games. The one for simple stochastic games is the oldest and best known. It is used by, e.g., Condon [12], who attributes it to Hoffman and Karp. Simple stochastic games are also the most general of the games we consider. The drawback of this measure is that each function evaluation involves solving a linear program with high precision. The measures for mean payoff and parity games avoid this. The mean payoff game function is the newest, recently discovered in Paper IV. It is discrete, simple, and efficient to compute. For parity games, Vöge and Jurdziński developed the first discrete strategy evaluation function in 2000 [52]. Papers I and V present modifications that allow for better complexity analysis.

The theory of controlled linear programming also involves strategy improvement, and provides a unified view of the strategy evaluation functions presented here.

One of the major contributions of this thesis is to show that all the strategy evaluation functions we investigate, except for general CLP, can be used together with randomized improvement policies, without reductions, to yield expected subexponential running time. This was previously only known for the case of binary simple stochastic games [36]. We discuss this further in Chapter 8.

7.1 Strategy Evaluation for Simple Stochastic Games

There is a well-known strategy evaluation function for simple stochastic games [12, 36]. Given a strategy σ of MAX, the value of a vertex is the probability of reaching the 1-sink when MAX uses σ and MIN uses an optimal counterstrategy against σ. The value of σ can be taken to be either a vector containing all vertex values, or simply their sum. Keeping track of the individual vertex values allows algorithms to make use of attractive switches. The counterstrategy of MIN and the vertex values can be found by solving a linear program; see, e.g., [12] and Paper V.

7.2 Strategy Evaluation for Discounted Payoff Games

For discounted payoff games, there is a similar strategy evaluation function, discussed by Puri [46]. Again, given a strategy σ of MAX, the value of a vertex v is the same as its value in the one player game G_σ, where MAX fixes σ, and only MIN has choices. This value can be computed in polynomial time using a fixed-point iteration method, or by solving a linear program.

7.3 Strategy Evaluation for Parity Games

Vöge and Jurdziński developed a discrete strategy evaluation function for parity games [52]. It makes the game quantitative, rather than qualitative, by stipulating that the players should try to win with the largest possible color, and should also try to optimize the colors seen on the path to the optimal loop.

Strategy improvement could already be used for parity games by reducing to discounted or simple stochastic games. The benefit of Vöge's and Jurdziński's function is that it is more efficient to compute. It avoids solving linear programs with high precision, instead using simple graph algorithms.

The strategy evaluation function from [52] assumes that all vertices have different colors, and the total number of different values that can be assigned to a vertex is $2^{\Omega(n)}$, where n is the number of vertices, regardless of the number k of colors in the original game. This is also the best known bound for any deterministic strategy improvement algorithm using the function.

In Paper I, we modify the function, avoiding the reduction to games with unique vertex colors. Basically, rather than keeping track of individual vertices that are visited on the path to the optimal loop, we record the *number* of vertices of each color. This allows us to improve the analysis for many algorithms to $O(poly(n) \cdot (n/k+1)^k)$.

7.4 Strategy Evaluation for Mean Payoff Games

In Paper IV we develop the first discrete strategy evaluation function for mean payoff games. The main idea behind this function is to look at a controlled version of the shortest paths problem. As we saw in Chapter 3, this problem is easy to solve on acyclic graphs, but for general directed graphs, it is considerably more complicated. We call the problem *longest-shortest paths* (LSP), since the controller (MAX) tries to make the shortest path from each vertex as long as possible. Given a graph G with sink t and strategy σ of MAX in the controlled vertices, we assign values to vertices in the following way. If a negative weight cycle is reachable from v in G_σ, then v gets value $-\infty$. If only positive value loops are reachable from v, and t is not, then v gets value

$+\infty$. Otherwise, we assign v the value of the shortest path from v to t in G_σ.[1] When we reduce mean payoff games to the longest-shortest paths problem, MAX can secure a value larger than 0 in the game from exactly the vertices where he can get value $+\infty$ in the LSP.

The LSP problem should not be confused with the NP-hard *longest path* problem. The difference is that in LSP, cycles are considered as infinitely long or infinitely short paths, while the longest path problem does not consider them as paths at all. Under reasonable assumptions, the decision version of the LSP problem belongs to NP∩coNP; see Paper IV.

The strategy evaluation function is easy to compute, using standard graph algorithms. The only arithmetic involved is additions and comparisons of numbers in the same order of magnitude as the edge weights. This makes algorithms considerably easier to implement, and much more efficient, than those that reduce to discounted or simple stochastic games and use the known evaluation functions for those games. These two factors also makes it attractive for solving parity games, giving a better complexity compared to Paper I.

Reduction to simple stochastic games can be uses to solve mean payoff games in randomized subexponential time; see, e.g., [7]. The new strategy evaluation function allowed us to show that in a combinatorial model of computation, the subexponential running time can be made completely independent of the edge weights.

7.5 Strategy Evaluation for Controlled Linear Programming

Recall that in the controlled linear programming problem (with nonnegative coefficients) a strategy is a selection of exactly one constraint per controlled variable. For an instance S and a strategy σ, the value of each variable under σ can be computed by solving the linear program

$$\begin{aligned} &\text{maximize } \textstyle\sum v, \;\; v \in \mathbf{x} \cup \mathbf{y} \\ &\text{subject to } S(\sigma). \end{aligned}$$

Paper V shows that for this evaluation function, attractive switches are profitable. Since all the games above can be rewritten as CLP instances, this gives a unified proof that attractivity implies profitability for all the strategy evaluation functions in this chapter.

In general, stability does not imply optimality in the CLP problem. However, it does for several broad subclasses, including those needed to cover the

[1] Loops with weight 0 constitute a special case, which complicates matters. Such loops can be avoided; see Paper IV for details.

games, as shown in Paper V.

7.6 Mixed Strategies and Interior Point Methods

The strategy evaluation functions for mean payoff, discounted payoff, and simple stochastic games, as well as the controlled linear programming problem, can also be extended to cover *mixed positional strategies*, where MAX assigns a probability distribution to the set of edges leaving each of his vertices. When the pebble reaches a vertex, he decides which edge to follow next randomly, according to this distribution. Iterative improvement can still be guided by the attractiveness of local changes. This corresponds to going through interior points of a polytope in a geometrical setting. Paper V describes this approach. An application to discounted payoff games can also be found in [45].

8 Combinatorial Optimization

One of the main contributions of this thesis is a unified characterization of strategy evaluation functions for games. For this purpose, we introduce the classes of completely local-global and recursively local-global functions, combinatorial counterparts of the strategy evaluation functions, and analyze their properties. The strategy evaluation functions we have seen are specific for each game, and the many details involved make it difficult to determine the structure of the functions. In order to understand and exploit their similarities and abstract properties, it is useful to study them in a unified way. This chapter outlines such a view.

The framework we propose is that of combinatorial optimization, which provides a rich toolbox of efficient deterministic and randomized algorithms and analytic tools that have successfully resolved many challenging problems. It seems natural to apply combinatorial and randomized algorithms to attack and solve game-theoretic problems arising in verification. Many algorithms become easier to analyze and reason about when stated in combinatorial terms, rather than game-theoretic. The reformulation also helps understanding and appreciating the problems better. Moreover, realizing the common structures underlying different games allows easy reuse of any future results for specific games in a more general setting. Our contribution towards these goals is twofold.

First, we show that some infinite games, including parity, mean payoff, discounted payoff, and simple stochastic, being recast in combinatorial terms turn out to be very easy-to-understand optimization problems, with clear and transparent structure. The simplicity of this structure allows one to abstract away from the complicated technical details of the games, and concentrate on essential properties, applying the full potential of combinatorial optimization and algorithmics. The set of all positional strategies of Player 0 in a graph game is isomorphic to a *hyperstructure*, a Cartesian product of finite sets, or simplices. We identify classes of functions with simple combinatorial definitions, that correspond to strategy evaluation functions. Thus, solving parity, mean payoff, and simple stochastic games amounts to finding global maxima of functions in these classes. The corresponding problem is easy to explain and can be appreciated by every mathematician and computer scientist.

Second, to optimize the function classes identified here, we suggest the

reuse of randomized subexponential algorithms developed for combinatorial linear programming. This establishes a direct relation between combinatorial optimization and game theory.

These considerations led to the first known expected subexponential algorithms for parity games, presented in Paper I, mean payoff games in Paper IV, and simple stochastic games of arbitrary outdegree [4, 3].

The theory of completely local-global (CLG) functions, presented in this chapter and Paper II, unifies several directions of research: connections to completely unimodal (CU) and local-global (LG) pseudo-Boolean optimization [30, 54, 51, 8], relation to linear programming, and analysis of single and multiple switching algorithms. This theory keeps much of the structure from the games, while making similarities and parallels between the games and other areas transparent.

Our starting point was an approach of Ludwig [36] who adapted a subexponential linear programming algorithm for the subclass of binary simple stochastic games. The possibilities of extending it for the general case and applying it directly to parity games remained undiscovered until we started investigating the combinatorial structure of the strategy evaluation functions.

The basic observation underlying our work is that iterative strategy improvement is closely connected to combinatorial optimization. In both cases, the objective is to find the element with the best value in a prohibitively large set. Any efficient algorithms for such a problem will have to make use of the structure of the value assignment. We make the connection explicit by examining how the evaluation functions for parity, mean payoff, discounted payoff, and simple stochastic games, as well as controlled linear programming, are structured. We believe that abstracting away from the specifics of each game, and concentrating on the common properties of these functions, will help in resolving the complexity of the problems. We demonstrate that the functions arising from the games we consider are very similar to local-global and completely unimodal functions that have previously been studied by the combinatorial optimization [54, 30, 53, 49], and linear programming communities [25, 24, 37]. We also show how this realization can be used to solve games, applying methods from linear programming.

8.1 Strategy Spaces and Hyperstructures

To represent the strategy spaces for memoryless determined games, as well as the controlled linear programming problem, in a unified way, we use the concept of *hyperstructures*, or products of simplices.

Definition 8.1.1 *Let $\mathscr{P}_1, \mathscr{P}_2, \ldots, \mathscr{P}_d$ be nonempty, pairwise disjoint sets. Then the product $\mathscr{P} = \mathscr{P}_1 \times \cdots \times \mathscr{P}_d$ is called a* hyperstructure *of dimension d.*

We use the term "hyperstructure" in analogy with hypercubes. When $|\mathscr{P}_i| = 2$ for all $1 \leq i \leq d$, the structure is isomorphic to the d-dimensional Boolean hypercube. Elements of the structure are called vertices. Two vertices of $\mathscr{P}$ are *neighbors* if they differ in exactly one coordinate.

A *substructure* of $\mathscr{P}$ is a product $\mathscr{P}' = \prod_{j=1}^{d} \mathscr{P}'_j$, where $\emptyset \neq \mathscr{P}'_j \subseteq \mathscr{P}_j$ for all j. A *facet* of $\mathscr{P}$ is a substructure obtained by fixing the choice in exactly one coordinate. Thus $\mathscr{P}'$ is a facet of $\mathscr{P}$ if there is a $j \in \{1, \ldots, d\}$ such that $|\mathscr{P}'_j| = 1$ and $\mathscr{P}'_k = \mathscr{P}_k$ for all $k \neq j$. Whenever $|\mathscr{P}'_j| = 1$ for some j, this coordinate can be disregarded, since it is fixed, and we can consider such structures to have a smaller dimension.

In a graph game where Player 0 controls d vertices, numbered 1 through d, let E_i be the set of edges leaving vertex i. Then the d-dimensional hyperstructure $S = E_1 \times \cdots \times E_d$ corresponds to the set of positional strategies for Player 0. Two vertices of S are neighbors exactly when we can get one of the corresponding strategies from the other by a single switch.[1] When Player 0 has only binary choices, S is a hypercube. In a controlled linear programming instance with d controlled variables, E_i is the set of constraints for variable x_i. A substructure of a hyperstructure corresponding to the strategies in a game is the set of strategies in a subgame, where some edges leaving vertices in V_0 have been deleted.

More about properties and terminology for hyperstructures can be found in Paper II and in [3, 7].

8.2 Functions on Hyperstructures

Functions on hyperstructures have been extensively studied in the combinatorial optimization community. Special attention has been given to functions on hypercubes; see, e.g., [8, 51]. In general, the problem of finding the maximum of an arbitrary function on a hyperstructure is of course exponential in the dimension; there is no better method than to evaluate the function for every vertex of the structure. For some classes of functions, however, better bounds are known. For others, the exact complexity is unknown. One of these is the class of *completely unimodal* (CU) functions [30, 54, 51, 8].[2] They are usually defined as having the real or natural numbers as codomain, but any partially ordered set can be used without losing any of the essential properties, as we show in [4].

In Paper II and [4, 7] we define the related class of completely local-global (CLG) functions. It is more general than the CU functions, but in fact, every CLG function can be reduced to a CU function such that the unique maxi-

[1] Recall that a single switch changes a positional strategy in exactly one vertex.

[2] Also known as abstract optimization functions [25].

mum of the latter is one of the maxima of the former. This has the benefit that any new results for CU functions immediately apply for optimizing CLG functions, even though working directly on the CLG function gives a better running time for most algorithms.

In [7] we show that the strategy evaluation functions for parity and simple stochastic games are completely local global. This means that any efficient method for optimizing CU functions can be applied to solving games. In particular, Williamson Hoke conjectured that any CU function on a hypercube can be optimized in randomized polynomial time [54]. If this is true, the same result immediately follows for parity, mean payoff, discounted payoff, and simple stochastic games.

Notation. We will consistently use $\mathscr{P}$ and D to denote hyperstructures and partially ordered sets, respectively. If $f : \mathscr{P} \to D$ is a function, and $\mathscr{P}'$ is a substructure of $\mathscr{P}$, we use $f_{|\mathscr{P}'}$ to denote the restriction of f to $\mathscr{P}'$.

When studying combinatorial function optimization two of the most important concepts are local and global optima. We remind of their definitions:

Definition 8.2.1 *Let $f : \mathscr{P} \to D$ be a partial function and let D be partially ordered by $\preceq$.*

1. *p is a* global maximum *of f if $f(p)$ is defined and $f(p) \succeq f(q)$ for every $q \in \mathscr{P}$ such that $f(q)$ is defined.*
2. *p is a* local maximum *of f if $f(p)$ is defined and $f(p) \succeq f(q)$ for every* neighbor *q of p such that $f(q)$ is defined.*

Global and local minima *are defined symmetrically.*

We now define the the most general class of functions that we know allow for subexponential optimization.

Definition 8.2.2 ([4, 3, 7]) *A partial function $f : \mathscr{P} \to D$ is called* recursively local-global *(RLG) if*

1. *for all neighbors p and q on $\mathscr{P}$ such that $f(p)$ and $f(q)$ are defined, the two function values are comparable,*
2. *for every substructure $\mathscr{P}'$ of $\mathscr{P}$, every local maximum of the restriction $f_{|\mathscr{P}'}$ of f to $\mathscr{P}'$ is also global.*

All the strategy evaluation functions mentioned in Chapter 7, except the one for the CLG problem in its full generality, are RLG. We will show that this is enough to ensure that they can be optimized in expected subexponential time in the number of dimensions (controlled vertices of Player 0). The RLG property does not, however, account for all the nice features of strategy evaluation functions. For the total functions, corresponding to parity, discounted payoff, and simple stochastic games, the subclass of completely local-global functions improve this situation.

Definition 8.2.3 ([4, 7]) *A total function $f : \mathscr{P} \to D$ such that all neighbors have comparable function values is* completely local-global *(CLG) if the following holds for every substructure $\mathscr{P}'$ of $\mathscr{P}$:*

1. *every local maximum of $f_{|\mathscr{P}'}$ is also global,*
2. *every local minimum of $f_{|\mathscr{P}'}$ is also global,*
3. *every pair of local maxima of $f_{|\mathscr{P}'}$ is connected by a path of local maxima,*
4. *every pair of local minima of $f_{|\mathscr{P}'}$ is connected by a path of local minima.*

The definition is rather technical, but actually captures properties of strategy evaluation functions in a very nice way. As discussed in Chapter 7 and Papers I and II their maxima can be found by *multiple switching algorithms*. Such algorithms make several attractive switches at once. For the algorithms to be correct, multiple switches must be profitable. We showed in [4, 7] that this is indeed the case for all CLG functions.

Theorem 8.2.4 *Let $f : \mathscr{P} \to D$ be a CLG function, $x \in \mathscr{P}$ and $y \in \mathscr{P}$ be vectors such that $y_i \neq x_i$ implies $f(x) \prec f(x$ with coordinate j changed to $y_j)$. Then $f(x) \prec f(y)$.*

CLG functions also have a number of other interesting properties [4, 7]. Paper II and [6] shows that through reduction to CU functions, any binary CLG function can be optimized with a randomized multiple switching algorithm, augmented with random sampling, in expected time $O(2^{0.453n})$.

8.3 The Structure of Strategy Evaluation functions

As mentioned, the strategy evaluation functions discussed in Chapter 7, again with the general CLP problem as only exception, are all RLG. For parity and simple stochastic games, the functions are even CLG.

Theorem 8.3.1 ([4, 7]) *The strategy evaluation functions for parity and simple stochastic games are CLG.*

This gives an alternative way of showing that multiple switching algorithms can be used with these function, and immediately implies that they possess a number of other interesting properties [4, 7].

The strategy evaluation function for mean payoff games from Paper IV is only defined for strategies such that the subgraph induced by the strategy has no negative weight cycles. Thus it cannot be CLG. It is, however, RLG, which we demonstrate here, since the proof has not been published elsewhere.

Theorem 8.3.2 *The strategy evaluation function for mean payoff games is RLG.*

Proof. Let (G,w) be an instance of longest-shortest paths with n vertices, obtained by reduction from a mean payoff game. The codomain of the strategy evaluation function is $\mathbb{Z}^n$. The function is defined exactly for the set of admissible strategies. For two tuples $\mathbf{x} = (x_1,\ldots,x_n)$ and $\mathbf{y} = (y_1,\ldots,y_n)$, we say that $\mathbf{x} < \mathbf{y}$ if $x_i \leq y_i$ for all $i \in \{1,\ldots,n\}$ and the inequality is strict in at least one coordinate. Any substructure of the hyperstructure corresponding to all strategies of MAX represents the strategies in a subgraph with the same unique sink as G. From the fact that any stable strategy is globally optimal (see Paper IV), we can infer that on any substructure, any local optimum of the function is also global.

It remains to show that the values of any two neighbors for which the function is defined are comparable. This follows by considering the subgraph where only these two strategies exist (MAX has a choice only in one vertex). If the values were incomparable, both of them would be locally maximal, contradicting the first part of the proof.

8.4 RLG Functions and LP-Type Problems

LP-type problems were first defined by Sharir and Welzl [48, 38]. They are intended as an abstract framework, capturing the structure of combinatorial linear programming and related problems, such as MINIBALL and POLYDIST. The randomized subexponential algorithm for linear programming they developed together with Matoušek [38], actually solves any LP-type problem.

This section shows how optimizing RLG-functions can be reduced to solving LP-type problems. Together with the results of Section 8.3, this provides a somewhat surprising link between game theory and computational geometry, since many problems from the latter domain can also be expressed in the LP-type framework. It also allows us to transfer any possible new algorithms for LP-type problem to RLG-optimization and games.

In order to pursue the analogy between hyperstructure optimization and LP-type problems, we need the following definition.

Definition 8.4.1 *Let $\mathscr{P} = \Pi_{i=1}^{d} \mathscr{P}_i$ be a hyperstructure, and E a subset of $\bigcup_{i=1}^{d} \mathscr{P}_i$. Then struct$(E)$ is the substructure $\Pi_{i=1}^{d}(\mathscr{P}_i \cap E)$ of $\mathscr{P}$. Thus, if E does not have elements from each $\mathscr{P}_j$, then struct$(E) = \emptyset$.*

We now define the abstract framework; see [48, 38] for details.

Definition 8.4.2 *Let H be a finite set, $\mathscr{W}$ a linearly ordered set, and $g : 2^H \to \mathscr{W}$ a function. The pair (H,g) is called an* LP-type problem *if it satisfies the*

following properties.

$$F \subseteq G \subseteq H \implies g(F) \leq g(G) \tag{8.1}$$

$$\left.\begin{array}{c} F \subseteq G \subseteq H \\ g(F) = g(G) \\ h \in H \end{array}\right\} \implies \left\{\begin{array}{c} g(F \cup h) > g(F) \\ \iff \\ g(G \cup h) > g(G) \end{array}\right. \tag{8.2}$$

Intuitively, each element of H corresponds to an element of some $\mathscr{P}_i$, and the value of a subset of H will be the maximal function value on the corresponding substructure.

A *basis* is a subset B of H with $g(B') < g(B)$ for all $B' \subset B$. The bases we will be considering correspond to vertices in hyperstructures. A *basis for* $G \subseteq H$ is a basis $B \subseteq G$ with $g(B) = g(G)$, in our case corresponding to a vertex of $struct(G)$ that maximizes the function value. To "solve" an LP-type problem means to find a basis for H. Note that "basis" is not the same as "basis for H".

Definition 8.4.3 *Let $f : \mathscr{P} \to D$ be an RLG function. For every substructure $\mathscr{P}'$ of $\mathscr{P}$, define $w_f(\mathscr{P}')$ to be the function value of the global maximum of $f_{|\mathscr{P}'}$, if f is defined for some vertex of $\mathscr{P}'$. Otherwise, $w_f(\mathscr{P}')$ is undefined.*

Now we are ready to prove the main theorem of this section. It generalizes a result we showed in [4] for total RLG functions.

Theorem 8.4.4 *An RLG-function $f : \mathscr{P} \to D$, where D is linearly ordered, can be expressed as an LP-type problem (H, g) such that a basis for H defines a global maximum of f.*

Proof. Recall that $\mathscr{P} = \prod_{j=1}^{d} \mathscr{P}_j$ and let H be the disjoint union of all $\mathscr{P}_j$. Let $m = |H|$. We assume, without loss of generality, that D is disjoint from the natural numbers. The value set of our LP-type problem will be $\mathscr{W} = \{0, 1, \ldots, 2m\} \cup D$, with the usual orders on $\{0, \ldots, 2m\}$ and D, and $n < d$ for all $n \in \mathbb{N}$ and $d \in D$. Now define $g : 2^H \to \mathscr{W}$.

$$g(G) = \begin{cases} |G| \in \mathbb{N} & \text{if } struct(G) = \emptyset; \\ m + |G| \in \mathbb{N} & \text{if } struct(G) \neq \emptyset \text{ and } w_f(struct(G)) \text{ is undefined;} \\ w_f(struct(G)) & \text{otherwise.} \end{cases}$$

Property (8.1) of Definition 8.4.2 is immediate: if $F \subseteq G$ and $g(F) \in D$ then $struct(F)$ is a substructure of $struct(G)$, so the maximum of f on $struct(F)$ is no bigger than the maximum on $struct(G)$. If $g(F) \in \mathbb{N}$ then $g(G)$ is either a bigger natural number or an element of D. We proceed to proving (8.2).

Let $F \subseteq G \subseteq H$ be such that $g(F) = g(G)$ and take $h \in H$. We need to prove that $g(F \cup h) > g(F) \iff g(G \cup h) > g(G)$. If $g(F) \in \mathbb{N}$, then $F = G$, since otherwise $g(G)$ would be greater than $g(F)$, and the equivalence follows. Assume $g(F) \in D$. The left-to-right implication is clear: if $g(F) \in D$ then $\mathit{struct}(F \cup h)$ is a substructure of $\mathit{struct}(G \cup h)$. Therefore $g(F \cup h) \leq g(G \cup h)$, and we get $g(G \cup h) \geq g(F \cup h) > g(F) = g(G)$.

Now suppose $g(G \cup h) > g(G)$. It remains to show that $g(F \cup h) > g(F)$. Assume the contrary and let $x \in \mathit{struct}(F)$ be a local maximum of f on $\mathit{struct}(F)$. Note that $g(G) = g(F) = g(F \cup h)$ by assumptions ($g(F)$ cannot be greater than or incomparable to $g(F \cup h)$ by Property (8.1) of Definition 8.4.2, which holds by the first part of this proof). Since any neighbor of x on $\mathit{struct}(G \cup h)$ is an element of either $\mathit{struct}(F \cup h)$ or $\mathit{struct}(G)$, they all have smaller, equal, or undefined values, so x is a local maximum on $\mathit{struct}(G \cup h)$. This contradicts the assumption that $\mathit{struct}(G \cup h)$ has a greater local maximum, so (H, g) is an LP-type problem.

We now show that that an element $x \in \mathscr{P}$ is a local maximum of f if and only if it is also a basis for H. If x is a maximum of f, then $g(x) = g(H)$. Also, since x has exactly one component for every $\mathscr{P}_j$, any subset of x has a value in $\mathbb{N}$. Thus x is a basis for H.

If, on the other hand x is a basis for H then it must have exactly one component from each $\mathscr{P}_j$: if $x \cap \mathscr{P}_j = \emptyset$ for some j, then x would not have the same value as H. Otherwise, if $|x \cap \mathscr{P}_j| > 1$ for some j, then there would be a subset of x with the same value. Thus x corresponds to an element of $\mathscr{P}$. Since $g(x) = g(H)$ it must be a global maximum of f on $\mathscr{P}$.

Although the theorem requires a linear order, which is not guaranteed by the RLG definition, this is not a major drawback. First, many algorithms for solving LP-type problems do not rely on the order being total. Second, the orders on strategy values for parity, mean payoff, discounted payoff, and simple stochastic games, as well as controlled linear programs, can easily be made total. Rather than comparing values of strategies componentwise in the tuple of vertex values, comparing them lexicographically extends the order to a total one.

8.5 Subexponential Optimization

Even though Khachiyan showed that the linear programming problem can be solved in polynomial time in 1979, research on algorithms has continued. For instance, Karmarkar developed a practically more efficient polynomial algorithm based on an interior point approach. One of the main reasons for the continuing interest in the problem is that it has not been determined whether there is a *strongly* polynomial algorithm for linear programming. Such an

algorithm would be polynomial *in the number of constraints and variables* in a combinatorial model of computation, while the running times Khachiyan's and Karmarkar's algorithms also depend on the actual numbers involved.

This led to the study of combinatorial linear programming. In 1992 a breakthrough was achieved when Kalai showed that a randomized algorithm could solve the problem in expected strongly subexponential time [35]. The same result was independently proved by Matoušek, Sharir, and Welzl [38, 48, 39].

In 1995, Ludwig used the same techniques to develop a subexponential randomized algorithm for binary simple stochastic games; games with vertex outdegree at most two [36]. It has a $2^{O(\sqrt{n})}$ upper bound on the number of strategy evaluations, where n is the number of vertices.

This result was a great improvement over earlier deterministic algorithms, but it has a drawback. Any simple stochastic game can be reduced to one with outdegree at most two, but if the original game has unbounded outdegree, the reduction increases the number of vertices to $\Omega(n^2)$. When applying Ludwig's algorithm, the number of iterations becomes $2^{O(\sqrt{n^2})} = 2^{O(n)}$.

Also, for solving parity and mean payoff games, reducing to simple stochastic games and applying the algorithm has another drawback. Every strategy evaluation requires solving a linear program with high precision.

Since the strategy evaluation function developed by Vöge and Jurdziński [52] had overcome the latter problem, we investigated how both the limit on outdegree, and the need for solving linear programs could be dealt with at the same time. This resulted in Paper I in which we modify both the evaluation function and Kalai's randomized algorithm. This shows that parity games can be solved in expected subexponential time without the need for costly high-precision arithmetic in each iteration.

The strategy evaluation function presented in Paper IV achieves the same for mean payoff games. Earlier algorithms for these games were pseudo-polynomial; $O(poly(n) \cdot W)$, where W is the largest edge weight [29, 56]. This can turn out badly when W is exponential in n, as is the case when reducing from parity games. The subexponential bound we achieve is *independent* of W in a combinatorial model of computation. Through a reduction, it also provides a simpler and more efficient method for solving parity games; see Paper IV.

To summarize, all strategy evaluation functions discussed in Chapter 7 can be optimized using at most $2^{O(\sqrt{n \log n})}$ function evaluations, where n is the number of controlled vertices. To do this, we can use either a version of Kalai's algorithm, or the version of the algorithm by Matoušek, Sharir, and Welzl shown as Algorithm 1; see also Paper V for applications to the controlled linear programming problem.

The algorithm starts with a vertex of the hyperstructure (a strategy) and randomly picks a facet F, not containing the vertex, to be temporarily thrown

Algoritm 1: MSW Optimization Algorithm

MSW(Hyperstructure $\mathscr{P}$, RLG-function f, initial vertex v_0)

(1) **if** $\dim(\mathscr{P}) = 0$
(2) **return** v_0
(3) choose a random facet F of $\mathscr{P}$, not containing v_0
(4) $v^* \leftarrow$ MSW($\mathscr{P} \setminus F, f, v_0$)
(5) $u \leftarrow$ the neighbor of v^* on F
(6) **if** $f(u) > f(v^*)$
(7) **return** MSW(F, f, u)
(8) **else**
(9) **return** v^*

away. It then recursively optimizes the function on the remaining structure, finding vertex v^*. If v^* is at least as good as its neighbor u on F, the algorithm terminates, since v^* is a local, and thus global, maximum. Otherwise, it jumps to u, and recursively optimizes the function on F.

The analysis from [38, 39] needs to be modified somewhat, since the domains of some RLG-functions are not linearly ordered. This is done in [7]. Note that the RLG property is essential for Algorithm 1 to work correctly. If the function to be optimized is not RLG, the algorithm can fail to find even a local minimum.

9 Fixed-Parameter Complexity

Although the material in this chapter does not conform exactly to the main theme of the thesis, it describes the starting point for an important complimentary field of research. As we have seen, the exact computational complexity for many graph games is unknown; the upper and lower bounds do not match. Other games are hard for presumably intractable classes. The goal of the research started and outlined here is to give a more subtle and refined classification of their complexity than classical theory is able to do.

To solve games with high or unknown complexity in practice, we can consider other aspects of their complexity, such as whether they are *fixed-parameter tractable* [16]. Quite often a problem instance (game) of size n contains a "natural" parameter k (number of colors or pairs), and the question is whether it can be solved in time $f(k) \cdot n^{O(1)}$ for an arbitrary function f. A positive answer indicates that large problem instances can be solved, as long as the value of the parameter is small. Since PTIME-membership proofs for, e.g., parity games appear elusive, looking at the fixed-parameter complexity is a natural first step. Many important combinatorial problems have been classified as fixed-parameter tractable or intractable [16, 9, 14, 23].

Here we consider the parameterized complexity of parity, Streett, Rabin, and Muller games. Typically, such games have decision algorithms of complexity $n^{O(k)}$. Settling their fixed-parameter tractability would give us a better understanding of which instances are practically solvable. On the other hand, proving fixed-parameter intractability saves any future work on finding polynomial time algorithms for parity games. Consequently, investigation of the fixed-parameter complexity is a promising and challenging research direction.

9.1 Fixed-Parameter Tractability

The theory of fixed-parameter complexity is two-dimensional [16]. A *parameterized language* L over a finite alphabet Σ is a subset of $\Sigma^* \times \mathbb{N}$. Call L *fixed-parameter tractable* (FPT) if there is a recursive function f and an algorithm that determines if $(x,k) \in L$ in time $f(k) \cdot n^c$ for $n = |x|$ and some constant c. The class XP is defined to contain all problems solvable in time $n^{f(k)}$ for some recursive function f, and it is known that FPT$\subsetneq$XP [16]. The basic working hypothesis of fixed-parameter complexity is that the intermediate class $W[1]$,

with complete problems PARAMETERIZED CLIQUE and INDEPENDENT SET, is also different from FPT, and there is strong evidence in support of this conjecture [16].[1] Recently, Downey *et al.* [15] introduced the new class $M[1]$, lying between FPT and $W[1]$. It is also believed to be different from FPT (otherwise there would be a subexponential algorithm for SAT) and should be useful for proving hardness results, since it is also presumably weaker than $W[1]$. The main parametric classes form the following "weirdness" hierarchy [16, 9]:

$$FPT \subseteq M[1] \subseteq W[1] \subseteq \cdots \subseteq W[P] \subseteq AW[*] \subseteq XP.$$

To establish hardness or membership in these classes, one uses FPT-reductions [16]. For our purposes, problem A is FPT-reducible to B ($A \leq_{fp} B$) if there is a Turing machine M_B with oracle for B deciding A, which on instance $\langle x,k\rangle$ terminates in time $f(k)\cdot|x|^c$ and is only allowed to form queries $\langle x',k'\rangle \in B$ with $k' \leq f(k)$ (here c is some constant and f is an arbitrary function.) The formal definitions of $W[1]$, ..., $W[P]$, and $AW[*]$ can be found in the monograph by Downey and Fellows [16] and the definition of $M[1]$ in [23]. Every problem that is hard for a class higher than FPT is considered fixed-parameter intractable. For recent surveys and articles on parameterized complexity we refer to [14, 15, 23].

9.2 Fixed-Parameter Complexity of Graph Games

For parity games, the most natural parameter is the number of colors. The relevance of this parameter is confirmed by the fact that when reducing from μ-calculus model checking, the number of colors correspond to the nestling depth of the fixpoint operators; see, e.g., [27]. In Rabin and Streett games, the number of pairs in the winning condition is a suitable parameter. For Muller games, we can take the number of equivalence classes of vertices, with respect to the winning condition, as parameter.

In [5] we show that, somewhat surprisingly, from the fixed-parameter point of view, parity games belong to the *same* complexity class as Streett, Rabin, and colored Muller games, with the above parameterizations. The standard reductions from parity to Rabin or Streett games, and from these to Muller games, are actually FPT reductions. To get the result we modified the known reduction from Muller to parity games; see, e.g., [40, 28, 50]; making it FPT. Most of the proof was already implicit in other work on the games involved [17], but the result had not been stated explicitly, and its implications, therefore, never considered.

We thus know that the four games belong to the same fixed-parameter class,

[1] This hypothesis is stronger than P $\neq$ NP.

but which one that is remains unknown. The only known upper bound is that they belong to *XP*, whose complete problems are provably intractable. Determining whether the games are fixed-parameter tractable or else complete for some class higher in the hierarchy shown in Section 9.1 would be of great interest. In [5] we provide a number of partial results in this direction. In particular, we show that if the players in Streett games are restricted to using only positional strategies, the problem is hard for $W[1]$, and thus presumably not fixed-parameter tractable. We also consider a number of other parameterizations and show results for subclasses of colored Muller games, some of which were implicit in [17].

10 Acknowledgements

First and foremost, I would like to thank my adviser, Sergei Vorobyov. Without his constant support, encouragement, interest, and never-ending flow of new, useful ideas, this work would never have been finished.

I am also indebted to the colleagues with whom I have worked closely over the years. Particularly Sven Sandberg, who participated in most of the work presented in this thesis, but also Olle Nilsson, Ola Svensson, and Viktor Petersson, who made very significant contributions.

I would also like to extend my thanks to everyone at the Information Technology Department for supplying a creative and enjoyable working environment. There are too many people to mention, but you all have my sincere appreciation.

Leonid Khachiyan, Vladimir Gurvich, and Endre Boros invited us to stay for a while at DIMACS, Rutgers University, New Jersey, at several occasions. The visits have been very productive and interesting, and I am deeply grateful.

The work presented here has been financed mainly by Vetenskapsrådet (the Swedish Research Council), which I would hereby like to thank. STINT (the Swedish Foundation for International Cooperation in Research and Higher Education) financed our visits to DIMACS, for which I am also very grateful.

Finally, I would like to thank my family and all my friends for their constant encouragement and support. I don't mention anyone in particular, for fear of forgetting someone, but I hope you all know how much you mean to me.

11 Summary in Swedish

Spelteorin har under det senaste seklet utvecklats av flera generationer framstående matematiker, till exempel John von Neumann, Leonid Kantorovich och John Nash. Den är nu en mogen disciplin med kapacitet att modellera och finna optimala beteenden i komplicerade scenarier, där många aktörer agerar i enlighet med sina intressen.

Också inom datavetenskapen har spelteorin funnit ett stort antal tillämpningar, och forskningen har lett fram till många användbara resultat. Spel används som modeller för datorsystem, logiker, automater och komplexitetsklasser.

En av de stora utmaningarna för dagens datavetenskap är det växande behovet av att verifiera stora mjuk- och hårdvarusystem, det vill säga kontrollera att de fungerar som avsett. Idag är det i princip omöjligt för programmerare att manuellt kontrollera att deras kod är korrekt, och i takt med att datorsystemen växer kommer det att bli ännu svårare. Därför behövs datorstödda verifieringsmetoder. Grundidén är att ta ett system och testa en viss egenskap. Ofta är dessa egenskaper ganska enkla, till exempel att systemet inte ska hänga sig, men för större system kan de också bli avsevärt mer komplicerade.

I princip vet vi redan hur vi kan kontrollera systemen maskinellt. Problemet är att göra det effektivt. När systemen växer blir många av dagens metoder helt enkelt för långsamma. Att råda bot på detta är en av de stora utmaningarna för dagens datavetenskap.

Verifieringsproblemen kan i många fall ses som spel. Oändliga tvåpersonspel med full information utgör ett väletablerat sätt att modellera interaktionen mellan ett system och dess omgivning. Om omgivningen kan vinna, genom att spela så att systemet tvingas bete sig på ett felaktigt sätt, är systemet inkorrekt. Givet en sådan spelmodell återstår att avgöra om det finns ett effektivt sätt att räkna ut vem som vinner, under antagandet att båda spelarna använder sina bästa strategier.

Ett av de spel vi undersöker i denna avhandling, paritetsspel, kan ses som en modell för att testa om ett system tillfredsställer krav definierade i logiken μ-kalkyl. Övriga spel som behandlas har andra, liknande tillämpningar.

Avhandlingen fokuserar på strategiförbättringsalgoritmer för oändliga spel på grafer. Enkelt uttryckt gäller det först att hitta ett sätt att utvärdera strategier så att man givet en förlorande strategi kan hitta en bättre. Utvärderingen ger

strategin ett värde, och kallas strategievalueringsfunktion. Meningen är att de bästa strategierna ska ha det högsta funktionsvärdet. Denna metod går i första hand att tillämpa när det finns ett ändligt antal strategier, vilket är fallet för paritetsspelen och deras släktingar.

Förutom evalueringsfunktionen behövs en metod för att givet en strategi hitta en annan, med bättre funktionsvärde. Hur denna metod väljs har en avgörande inverkan på algoritmens effektivitet.

I detta arbete behandlar vi båda delarna av strategiförbättringsalgoritmer. Vi uppfinner nya strategievalueringsfunktioner och förfinar gamla. Genom att undersöka dessa funktioners kombinatoriska struktur visar vi också hur slumpbaserade metoder från kombinatorisk optimering kan användas för att hitta strategier med större och större värden. Detta leder till nya, mer effektiva algoritmer för att lösa de spel vi undersöker.

Vårt arbete med att utveckla strategievalueringsfunktioner har tre delar. För paritetsspel förfinar vi den funktion som Vöge och Jurdziński utvecklat, vilket leder till bättre komplexitetsanalys för vissa algoritmer. Vi presenterar den första diskreta evalueringsfunktionen för så kallade medelavkastningsspel. Den är baserad på vad vi kallar längsta-kortaste vägen, en ny, kontrollerad och spellik, version av problemet att hitta den kortaste vägen mellan två noder i en riktad graf. Fördelen med denna funktion är att algoritmer kan undvika att göra kostsamma högprecisionsberäkningar varje gång en strategi utvärderas. I en kombinatorisk beräkningsmodell blir tidsåtgången för sådana algoritmer oberoende av vikterna i spelgrafen. Till sist visar vi att strategievaluering också kan användas på ett förtjänstfullt sätt för att lösa ett nytt problem, kontrollerad linjärprogrammering. Detta problem definieras i avhandlingen, och generaliserar de spel vi undersöker.

Vi analyserar den kombinatoriska strukturen hos de strategievalueringsfunktioner vi behandlar. Det visar sig att samtliga har likartad struktur, vilket gör det möjligt att förbättra analysen för vissa algoritmer. Vidare visar vi att problemet med att hitta den strategi som har störst värde kan reduceras till att lösa problem av så kallad LP-typ, definierade av Matoušek, Sharir och Welzl. Detta innebär att alla algoritmer för att lösa de senare kan användas också för paritetsspelen och deras släktingar.

I synnerhet visar vi att alla de spel vi undersöker, samt många restriktioner av kontrollerad linjärprogrammering, kan lösas med de subexponentiella algoritmer för kombinatorisk linjärprogrammering som uppfanns av Kalai respektive Matoušek, Sharir och Welzl.

Det är fortfarande oklart om paritetsspel kan lösas i polynomisk tid. Samtidigt är det mycket osannolikt att det skulle gå att visa några icketriviala undre gränser inom ramen för klassisk komplexitetsteori. Vi påbörjar därför arbetet med att klassificera deras så kallade fixparameterkomplexitet. Det visar sig att paritetsspel från denna synvinkel är ekvivalenta med rabin-, streett- och

mullerspel, vilket kontrasterar mot klassisk komplexitetsteori. Den exakta fixparameterkomplexiteten är fortfarande okänd, men vi visar att en variant av streettspel, där bara positionella strategier är tillåtna, har hög komplexitet.

Avhandlingen innehåller också ett bevis för att paritets- och medelavkastningsspel är bestämda i positionella strategier. Detta är på intet vis ett nytt resultat. Många andra bevis har redan publicerats. Motivet är att undersöka om ett helt konstruktivt bevis är möjligt, utan att referera till ickeelementära metoder eller fixpunktsteorem, samtidigt som det fungerar likformigt för båda typerna av spel. Vi hoppas att det, tillsammans med andra bevis, kan bidra till att öka förståelsen för spelens inre mekanik.

References

[1] M. Agrawal, N. Kayal, and N. Saxena. Primes is in P. Unpublished manuscript, `http://www.cse.iitk.ac.in/users/manindra/`, August 2002.

[2] H. Björklund, O. Nilsson, O. Svensson, and S. Vorobyov. Controlled linear programming: Duality and boundedness. Technical Report 2004-56, DIMACS (Center for Discrete Mathematics and Theoretical Computer Science) Rutgers University, NJ, December 2004. `http://dimacs.rutgers.edu/TechnicalReports/`.

[3] H. Björklund and S. Sandberg. Algorithms for combinatorial optimization and games adapted from linear programming. In B. ten Cate, editor, *Proceedings of the Eighth European Summer School on Logic Language, and Information (ESSLLI'2003) Student Session*, 2003.

[4] H. Björklund, S. Sandberg, and S. Vorobyov. On combinatorial structure and algorithms for parity games. Technical Report 2003-002, Department of Information Technology, Uppsala University, 2003. `http://www.it.uu.se/research/reports/`.

[5] H. Björklund, S. Sandberg, and S. Vorobyov. On fixed-parameter complexity of infinite games. In K. Sere and M. Waldén, editors, *Nordic Workshop on Programming Theory*, pages 62–64. Åbo Akademi, Dept. Computer Science, 2003. Full version in Technical Report 2003-038, Department of Information Technology, Uppsala University.

[6] H. Björklund, S. Sandberg, and S. Vorobyov. Randomized subexponential algorithms for parity games. Technical Report 2003-019, Department of Information Technology, Uppsala University, 2003. `http://www.it.uu.se/research/reports/`.

[7] H. Björklund, S. Sandberg, and S. Vorobyov. Randomized subexponential algorithms for infinite games. Technical Report DIMACS-2004-09, DIMACS: Center for Discrete Mathematics and Theoretical Computer Science, Rutgers University, NJ, April 2004. `http://dimacs.rutgers.edu/TechnicalReports/`.

[8] E. Boros and P. L. Hammer. Pseudo-boolean optimization. Technical Report RRR 48-2001, RUTCOR Rutger Center for Operations Research, 2001.

[9] M. Cesati. Compendium of parameterized problems. `http://bravo.ce.uniroma2.it/home/cesati/research`, 2001.

[10] E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2000.

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

[12] A. Condon. On algorithms for simple stochastic games. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 13:51–71, 1993.

[13] C. Derman. *Finite-State Markovian Decision Processes*. Academic Press, 1970.

[14] R. G. Downey. Parameterized complexity for the skeptic. In *IEEE Computational Complexity Conference (CCC'03)*, 2003.

[15] R. G. Downey, V. Estivill-Castro, M. R. Fellows, E. Prieto-Rodriguez, and F. A. Rosamond. Cutting up is hard to do: the parameterized complexity of k-cut and related problems. *Electronic Notes in Theoretical Computer Science*, 78:205–218, 2003.

[16] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer-Verlag, 1999.

[17] S. Dziembowski, M. Jurdzinski, and I. Walukiewitz. How much memory is needed to win infinite games? In G. Winskel, editor, *Logic in Computer Science, LICS'97*, pages 99–110. IEEE, 1997.

[18] A. Ehrenfeucht and J. Mycielski. Positional games over a graph. *Notices Amer. Math Soc.*, 20:A–334, 1973.

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

[20] E. A. Emerson. Automata, tableaux, and temporal logics. In R Parikh, editor, *Proc. Workshop on Logic of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 79–87. Springer-Verlag, 1985.

[21] E. A. Emerson. Model checking and the Mu-calculus. In N. Immerman and Ph. G. Kolaitis, editors, *DIMACS Series in Discrete Mathematics*, volume 31, pages 185–214, 1997.

[22] E. A. Emerson and C. S. Jutla. Tree automata, μ-calculus and determinacy. In *IEEE Symposium on Foundations of Computer Science*, pages 368–377, 1991.

[23] M. R. Fellows. New directions and new challenges in algorithm design and complexity, parameterized. In *Workshop on Algorithms And Data Structures*, volume 2748 of *Lecture Notes in Computer Science*, pages 505–520. Springer-Verlag, 2003.

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

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

[26] H. Gimbert and W. Zielonka. When can you play positionally? In J. Fiala, V. Koubek, and J Kratochvíl, editors, *Proceedings of the 29th International Symposium on Mathematical Foundations of Computer Science, MFCS 2004*, volume 3153 of *Lecture Notes in Computer Science*, pages 686–697. Springer, 2004.

[27] Erich Grädel, Wolfgang Thomas, and Thomas Wilke. *Automata, Logics and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.

[28] Y. Gurevich and L. Harrington. Trees, automata, and games. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 60–65, May 1982.

[29] V. A. Gurvich, A. V. Karzanov, and L. G. Khachiyan. Cyclic games and an algorithm to find minimax cycle means in directed graphs. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 28(5):85–91, 1988.

[30] P. L. Hammer, B. Simeone, Th. M. Liebling, and D. De Werra. From linear separability to unimodality: a hierarchy of pseudo-boolean functions. *SIAM J. Disc. Math.*, 1(2):174–184, 1988.

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

[32] R. Hossley and C. Rackoff. The emptiness problem for automata on infinite trees. In *13th Symp. on Automata and Switching Theory*, pages 121–124, 1972.

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

[34] M. Jurdziński. Small progress measures for solving parity games. In H. Reichel and S. Tison, editors, *17th STACS*, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301. Springer-Verlag, 2000.

[35] G. Kalai. A subexponential randomized simplex algorithm. In *24th ACM STOC*, pages 475–482, 1992.

[36] W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and Computation*, 117:151–155, 1995.

[37] J. Matoušek. Lower bounds for a subexponential optimization algorithm. *Random Structures and Algorithms*, 5(4):591–607, 1994.

[38] J. Matoušek, M. Sharir, and M. Welzl. A subexponential bound for linear programming. In *8th ACM Symp. on Computational Geometry*, pages 1–8, 1992.

[39] J. Matoušek, M. Sharir, and M. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16:498–516, 1996.

[40] R. McNaughton. Finite state infinite games. Technical report, Massachusetts Institute of Technology, September 1965.

[41] R. McNaughton. Infinite games played on finite graphs. *Annals of Pure Applied Logic*, 65:149–184, 1993.

[42] A.W. Mostowski. Games with forbidden positions. Technical Report 78, Uniwersytet Gdański, Instytut Matematyki, 1991.

[43] H. Moulin. Prolongements des jeux de deux joueurs de somme nulle. *Bull. Soc. Math. France*, 45, 1976.

[44] C. Papadimitriou. Algorithms, games, and the internet. In *ACM Annual Symposium on Theory of Computing*, pages 749–753. ACM, July 2001.

[45] V. Petersson and S. Vorobyov. Parity games: interior-point approach. Technical Report 008, Uppsala University / Information Technology, May 2001. `http://www.its.uu.se/research/reports/`.

[46] A. Puri. *Theory of hybrid systems and discrete events systems*. PhD thesis, EECS Univ. Berkeley, 1995.

[47] M. Rabin. Decidability of second order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[48] M. Sharir and E. Welzl. A combinatorial bound for linear programming and related problems. In *9th Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 577 of *Lecture Notes in Computer Science*, pages 569–579, Berlin, 1992. Springer-Verlag.

[49] T. Szabo and E. Welzl. Unique sink orientations of cubes. In *IEEE Symposium on Foundations of Computer Science*, pages 547–555, 2001.

[50] W. Thomas. Languages, automata, and logic. *Handbook of Formal Languages*, 3:389–455, 1997.

[51] C. A. Tovey. Local improvement on discrete structures. In E. Aarts and Lenstra J. K., editors, *Local Search in Combinatorial Optimization*, pages 57–89. John Wiley & Sons, 1997.

[52] J. Vöge and M. Jurdziński. A discrete strategy improvement algorithm for solving parity games. In E. A. Emerson and A. P. Sistla, editors, *CAV'00: Computer-Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*, pages 202–215. Springer-Verlag, 2000.

[53] D. Wiedemann. Unimodal set-functions. *Congressus Numerantium*, 50:165–169, 1985.

[54] K. Williamson Hoke. Completely unimodal numberings of a simple polytope. *Discrete Applied Mathematics*, 20:69–81, 1988.

[55] W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200:135–183, 1998.

[56] U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158:343–359, 1996.

Acta Universitatis Upsaliensis

Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology 3

Editor: The Dean of the Faculty of Science and Technology

A doctoral dissertation from the Faculty of Science and Technology, Uppsala University, is usually a summary of a number of papers. A few copies of the complete dissertation are kept at major Swedish research libraries, while the summary alone is distributed internationally through the series Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology. (Prior to January, 2005, the series was published under the title "Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology".)

Distribution: publications.uu.se
urn:nbn:se:uu:diva-4751

ACTA
UNIVERSITATIS
UPSALIENSIS
UPPSALA
2005

Paper I

A Discrete Subexponential Algorithm for Parity Games*

Henrik Björklund, Sven Sandberg, and Sergei Vorobyov

Computing Science Department, Uppsala University, Sweden

Abstract. We suggest a new randomized algorithm for solving parity games with worst case time complexity roughly

$$\min\left(O\left(n^3\cdot\left(\frac{n}{k}+1\right)^k\right),\ 2^{O(\sqrt{n\log n})}\right),$$

where n is the number of vertices and k the number of colors of the game. This is comparable with the previously known algorithms when the number of colors is small. However, the subexponential bound is an advantage when the number of colors is large, $k=\Omega(n^{1/2+\varepsilon})$.

1 Introduction

Parity games are infinite games played on finite directed bipartite leafless graphs, with vertices colored by integers. Two players alternate moving a pebble along edges. The goal of Player 0 is to ensure that the biggest color visited by the pebble infinitely often is even, whereas Player 1 tries to make it odd. The complexity of determining a winner in parity games, equivalent to the Rabin chain tree automata non-emptiness, as well as to the μ-calculus[1] model checking [5,3], is a fundamental open problem in complexity theory [11]. The problem belongs to NP∩coNP, but its PTIME-membership status remains widely open. All known algorithms for the problem are exponential, with an exception of [12] when the number of colors is large and games are binary.

In this paper we present a new discrete, randomized, subexponential algorithm for parity games. It combines ideas from iterative strategy improvement based on randomized techniques of Kalai [9] for Linear Programming and of Ludwig [10] for simple stochastic games, with discrete strategy evaluation similar to that of Vöge and Jurdziński [15]. Generally, algorithms for parity games are *exponential* in the number of colors k, which may be as big as the number n of vertices. For most, exponentially hard input instances are known [4,3,2,14, 8]. Our algorithm is subexponential in n. Earlier we suggested a subexponential algorithm [12], similar to [10], but based on graph optimization rather than linear programming subroutines. Both algorithms [10,12] become exponential

* Supported by Swedish Research Council Grants "Infinite Games: Algorithms and Complexity", "Interior-Point Methods for Infinite Games".

[1] One of the most expressive temporal logics of programs [3].

H. Alt and M. Habib (Eds.): STACS 2003, LNCS 2607, pp. 663–674, 2003.

for graphs with unbounded vertex outdegree. The present paper eliminates this drawback. There is a well-known reduction from parity to mean payoff games, but the best known algorithms for the latter [7,16,13] are known to be exponential (pseudopolynomial). Reducing parity to simple stochastic games [16] leads to manipulating high-precision arithmetic and to algorithms invariably subexponential in the number of vertices, which is worse than an exponential dependence on colors when colors are few.

A recent iterative strategy improvement algorithm [15] uses a discrete strategy evaluation involving game graph characteristics like colors, sets of vertices, and path lengths. Despite a reportedly good practical behavior, the only known worst-case bound for this algorithm is exponential in the number of vertices, independently of the number of colors.

Our new algorithm avoids any reductions and directly applies to parity games of arbitrary outdegree. We use a discrete strategy evaluation measure similar to, but more economical than the one used in [15]. Combined with Kalai's and Ludwig's randomization schemes this provides for a worst case bound that is simultaneously subexponential in the number of vertices and exponential in the number of colors. This is an advantage when the colors are few.

Outline. After preliminaries on parity games, we start by presenting a simpler, Ludwig-style randomized algorithm in combination with an abstract discrete measure on strategies. This simplifies motivation, exposition, and definitions for the specific tight discrete measure we build upon. We then proceed to a more involved Kalai-style randomized algorithm allowing for arbitrary vertex outdegrees. All proofs can be found in [1].

2 Parity Games

Definition 1 (Parity Games). *A* parity game *is an infinite game played on a finite directed bipartite leafless graph* $G[n,k] = (V_0, V_1, E, c)$*, where* $n = |V_0 \cup V_1|$*,* $E \subseteq (V_0 \times V_1) \cup (V_1 \times V_0)$*,* $k \in \mathbb{N}$*, and* $c : V_0 \cup V_1 \to \{1, \ldots, k\}$ *is a coloring function. The sizes of* V_0 *and* V_1 *are denoted by* n_0 *and* n_1*, respectively. Starting from a vertex, Player 0 and 1 alternate moves constructing an infinite sequence of vertices; Player i moves from a vertex in* V_i *by selecting one of its successors. Player 0 wins if the highest color encountered infinitely often in this sequence is even, while Player 1 wins otherwise.*[2] □

Parity games are known to be *determined*: from each vertex exactly one player has a winning positional strategy, selecting a unique successor to every vertex [5]. All our results straightforwardly generalize to the non-bipartite case.

A *binary* parity game is a game where the vertex outdegree is at most two.

[2] We systematically use n for the number of vertices and k for the number of colors; consequently we usually skip $[n,k]$ in $G[n,k]$.

3 Ludwig-Style Algorithm with a Well-Behaved Measure

Every positional strategy of Player 0 in a binary parity game can be associated with a corner of the n_0-dimensional boolean hypercube. If there is an appropriate way of assigning values to strategies, then we can apply an algorithm similar to [10] to find the best strategy as follows.

1. Start with some strategy σ_0 of Player 0.
2. Randomly choose a facet F of the hypercube, containing σ_0.
3. Recursively find the best strategy σ' on F.
4. Let σ'' be the neighbor of σ' on the opposite facet $\overline{F}$. If σ' is better than σ'', then return σ'. Else recursively find the optimum on $\overline{F}$, starting from σ''.

To guarantee correctness and subexponentiality, the assignment cannot be completely unstructured. Also, evaluating strategies is costly, so a full evaluation should only be performed for strategies that are really better than the current one. In subsequent sections, we present a function Evaluate that given a strategy σ returns an assignment ν_σ of values to vertices of the game that meets the following criteria (where $\prec$ is a comparison operator on the values).

Stability. Let $\sigma(v)$ be the successor of vertex v selected by strategy σ and let $\overline{\sigma}(v)$ be the other successor of v. If $\nu_\sigma(\sigma(v)) \succeq \nu_\sigma(\overline{\sigma}(v))$ for all vertices v of Player 0, then σ is optimal (maximizes the winning set of Player 0).

Uniqueness of optimal values. All optimal strategies have the same valuation. (This is essential for a subexponential bound.)

Profitability. Suppose that $\nu_\sigma(\sigma(u)) \succeq \nu_\sigma(\overline{\sigma}(u))$ for every vertex $u \in V_0 \setminus v$ and $\nu_\sigma(\sigma(v)) \prec \nu_\sigma(\overline{\sigma}(v))$ (attractiveness). Let σ' be the strategy obtained by changing σ only at v (single switch), and let $\nu_{\sigma'}$ be its valuation. Then $\nu_\sigma(v) \prec \nu_{\sigma'}(v)$ and $\nu_\sigma(u) \preceq \nu_{\sigma'}(u)$ for all other vertices u (profitability).

The Ludwig-style algorithm with Evaluate applies to solving binary parity games. The evaluation function has the benefit that in step 4 of the algorithm, σ'' does not have to be evaluated, unless the recursive call is needed.

Ludwig [10] shows that his algorithm for simple stochastic games has a $2^{O(\sqrt{n_0})}$ upper bound on the expected number of improvement steps. With only minor modifications, the same proof shows that the Ludwig-style algorithm together with our Evaluate function has the same bound for parity games.

The value space of Evaluate allows at most $O(n^3 \cdot (n/k+1)^k)$ improvement steps. Since the algorithm makes only improving switches, the upper bound on the number of switches of the combined approach is

$$\min\left(O\left(n^3 \cdot \left(\frac{n}{k}+1\right)^k\right), 2^{O(\sqrt{n_0})}\right).$$

Any parity game reduces to a binary one. This allows for a subexponential algorithm for games with subquadratic total outdegree. For arbitrary games the reduction gives a quadratic explosion in the number of vertices and the Ludwig-style algorithm becomes exponential. In Section 10 we achieve a subexponential bound by employing a more involved randomization scheme from [9].

4 Strategies and Values

For technical reasons, each vertex is assigned a unique value, called a *tint*.

Definition 2 (Tints). *A bijection* $\mathbf{t} : V \to \{1, \ldots, n\}$ *such that* $c(u) \leq c(v) \Rightarrow \mathbf{t}(u) \leq \mathbf{t}(v)$ *assigns tints to vertices. The color of a tint* $s \in \{1, \ldots, n\}$ *equals* $c(\mathbf{t}^{-1}(s))$. □

Note that tints of vertices of the same color form a consecutive segment of natural numbers. Subsequently we identify vertices with their tints, and slightly abuse notation by writing $c(t)$ for the color of the vertex with tint t.

Definition 3 (Winning and Losing Colors and Tints). *Color* i *is* winning for Player 0 *(Player 1 resp.) if it is even (odd resp.). Tint* t *is* winning for Player 0 *(Player 1 resp.) if its color* $c(t)$ *is. A color or tint is* losing *for a player if it is winning for his adversary.* □

Note that tints of different colors are ordered as these colors. Within the same winning (resp. losing) color the bigger (resp. smaller) tint is better for Player 0.

In this section we define the 'value' of a strategy – the target to be iteratively improved. An elementary improvement step is as follows: given a strategy σ of Player 0, its value is a vector of values of all vertices of the game, assuming that the adversary Player 1 applies an 'optimal' response counterstrategy τ against σ. The value of each vertex is computed with respect to the pair of strategies (σ, τ), where the optimality of τ is essential for guiding Player 0 in improving σ. We delay the issue of constructing optimal counterstrategies until Section 9, assuming for now that Player 1 always responds with an optimal counterstrategy.

Definition 4. *A* positional strategy *for Player 0 is a function* $\sigma : V_0 \to V_1$, *such that if* $\sigma(v) = v'$, *then* $(v, v') \in E$. *Saying that Player 0 fixes his positional strategy means that he deterministically chooses the successor* $\sigma(v)$ *each time the play comes to* v, *independently of the history of the play. Positional strategies for Player 1 are defined symmetrically.* □

Assumption. From now on we restrict our attention to positional strategies only. The iterative improvement proceeds by improving positional strategies for Player 0, and this is justified by Profitability, Stability, and Uniqueness Theorems 20, 22, and 23 below. The fact that Player 1 may also restrict himself to positional strategies is demonstrated in Section 9.

Definition 5 (Single Switch). *A* single switch *in a positional strategy* σ *of Player 0 is a change of successor assignment of* σ *in exactly one vertex.* □

When the players fix their positional strategies, the trace of any play is a simple path leading to a simple loop. Roughly speaking, the value of a vertex with respect to a pair of positional strategies consists of a loop value (major tint) and a path value (a record of the numbers of more significant colors on the path to the major, plus the length of this path), as defined below.

Notation 6 *Denote by* V^i *the set of vertices of* color i *and by* $V^{>t}$ *the set of vertices with* tints *numerically bigger than* t. □

Definition 7 (Traces, Values). *Suppose the players fix positional strategies σ and τ, respectively. Then from every vertex u_0 the* trace of the play *takes a simple δ-shape form: an initial simple path (of length $q \geq 0$, possibly empty) ending in a loop:*

$$u_0, u_1, \ldots u_q, \ldots, u_r, \ldots u_s = u_q, \tag{1}$$

where all vertices u_i are distinct, except $u_q = u_s$. The vertex u_r with the maximal tint t on the loop $u_q, \ldots, u_r, \ldots, u_s = u_q$ in (1) is called principal *or* major.

Values for Non-principal Vertices. *If the vertex u_0 is non-principal, then its* value $\nu_{\sigma,\tau}(u_0)$ *with respect to the pair of strategies (σ, τ) has the form (L, P, p) and consists of:*

Loop Value (Tint) *L equal to the principal tint t;*
Path Color Hit Record Relative to t *defined as a vector*

$$P = (m_k, m_{k-1}, \ldots, m_l, \underbrace{0, \ldots, 0}_{l-1 \text{ times}}),$$

where $l = c(t)$ is the color of the principal tint t, and

$$m_i = \left|\{u_0, u_1, \ldots u_{r-1}\} \cap V^i \cap V^{>t}\right|$$

is the number of vertices of color $i \geq l$ on the path to from u_0 to the major u_r (except that for the color l of the major we account only for the vertices with tint bigger than t.)
Path Length $p = r$.

Values for Principal Vertices. *If the vertex u_0 is principal (case $q = r = 0$ in (1)) then its value $\nu_{\sigma,\tau}(u_0)$ with respect to the pair of strategies (σ, τ) is defined as $(t, \bar{0}, s)$, where $\bar{0}$ is a k-dimensional vector of zeros.*

Path Value *is a pair (P, p), where (t, P, p) is a vertex value.* □

The reason of the complexity of this definition is to meet the criteria enumerated in Section 3 and simultaneously obtain the 'tightest possible' bound on the number of iterative improvements. It is clear that such a bound imposed by the value measure from Definition 7 is $O(n^3 \cdot (n/k+1)^k)$.

5 Value Comparison and Attractive Switches

Definition 8 (Preference Orders). *The* preference order *on colors (as seen by Player 0) is as follows: $c \prec c'$ iff $(-1)^c \cdot c < (-1)^{c'} \cdot c'$.*

The preference order *on tints (as seen by Player 0) is as follows:*

$$t \prec t' \text{ iff } (-1)^{c(t)} \cdot t < (-1)^{c(t')} \cdot t'. \qquad \square$$

We thus have $\ldots \prec 5 \prec 3 \prec 1 \prec 0 \prec 2 \prec 4 \prec \ldots$ on colors.

Definition 9 ('Lexicographic' Ordering). *Given two vectors (indexed in descending order from the maximal color k to some $l \geq 1$)*

$$P = (m_k, m_{k-1}, \ldots, m_{l+1}, m_l),$$
$$P' = (m'_k, m'_{k-1}, \ldots, m'_{l+1}, m'_l),$$

define $P \prec P'$ if the vector

$$\big((-1)^k \cdot m_k, (-1)^{k-1} \cdot m_{k-1}, \ldots, (-1)^{l+1} \cdot m_{l+1}, (-1)^l \cdot m_l\big)$$

is lexicographically smaller (assuming the usual ordering of integers) than the vector $\big((-1)^k \cdot m'_k, (-1)^{k-1} \cdot m'_{k-1}, \ldots, (-1)^{l+1} \cdot m'_{l+1}, (-1)^l \cdot m'_l\big)$. □

Definition 10 (Path Attractiveness). *For two vertex values (t, P_1, p_1) and (t, P_2, p_2), where t is a tint, $l = c(t)$ is its color, and*

$$P_1 = (m_k, m_{k-1}, \ldots, m_{l+1}, m_l, \ldots, m_1),$$
$$P_2 = (m'_k, m'_{k-1}, \ldots, m'_{l+1}, m'_l, \ldots, m'_1),$$

say that the path value (P_2, p_2) is more attractive[3] modulo t *than the path value (P_1, p_1), symbolically $(P_1, p_1) \prec_t (P_2, p_2)$, if:*

1. *either $(m_k, m_{k-1}, \ldots, m_{l+1}, m_l) \prec (m'_k, m'_{k-1}, \ldots, m'_{l+1}, m'_l)$,*
2. *or $(m_k, m_{k-1}, \ldots, m_{l+1}, m_l) = (m'_k, m'_{k-1}, \ldots, m'_{l+1}, m'_l)$ and*

$$(-1)^l \cdot p_1 > (-1)^l \cdot p_2. \tag{2}$$

Remark 11. Note that (2) means that shorter (resp. longer) paths are better for Player 0 when the loop tint t is winning (resp. losing) for him. □

Definition 12 (Value Comparison). *For two vertex values define $(t_1, P_1, p_1) \prec (t_2, P_2, p_2)$ if*

1. *either $t_1 \prec t_2$,*
2. *or $t_1 = t_2 = t$, and $(P_1, p_1) \prec_t (P_2, p_2)$.* □

Definition 13 (Vertex Values). *The value $\nu_\sigma(v)$ of a vertex v with respect to a strategy σ of Player 0 is the minimum of the values $\nu_{\sigma,\tau}(v)$, taken over all strategies τ of Player 1.* □

In Section 9 we show that the 'minimum' in this definition can be achieved in all vertices simultaneously by a positional strategy τ of Player 1.

Definition 14. *The value of a strategy σ of Player 0 is a vector of values of all vertices with respect to the pair of strategies (σ, τ), where τ is an optimal response counterstrategy of Player 1 against σ; see Section 9.* □

[3] In the sequel, when saying "attractive", "better", "worse", etc., we consistently take the viewpoint of Player 0.

Definition 15. *A strategy* σ' improves σ*, symbolically* $\sigma \prec \sigma'$*, if* $\nu_\sigma(v) \preceq \nu_{\sigma'}(v)$ *for all vertices* v *and there is at least one vertex* u *with* $\nu_\sigma(u) \prec \nu_{\sigma'}(u)$. □

Proposition 16. *The relations* $\prec$ *on colors, tints, values, and strategies, and* $\prec_t$ *on path values (for each* t*) are transitive.* □

Our algorithms proceed by single attractive switches only.

Definition 17 (Attractive Switch). *Let* (t_1, P_1, p_1) *and* (t_2, P_2, p_2) *be the values with respect to* σ *of vertices* v_1 *and* v_2*, respectively. Consider a single switch in strategy* σ *of Player 0, consisting in changing the successor of* v *with respect to* σ *from* v_1 *to* v_2*. The switch is* attractive *if* $(t_1, P_1, p_1) \prec (t_2, P_2, p_2)$. □

Remark 18. Note that deciding whether a switch is attractive (when comparing values of its successors) we do not directly account for the color/tint of the current vertex. However, this color/tint may be eventually included in the values of successors possibly dependent on the current vertex.

6 Profitability of Attractive Switches

Our algorithms proceed by making single attractive switches. Attractiveness is established locally, by comparing values of a vertex successors with respect to a current strategy; see Definition 17.

Definition 19. *Say that a single switch from* σ *to* σ' *is* profitable *if* $\sigma \prec \sigma'$. □

Profitability of attractive switches is crucial for the efficiency, correctness, and termination of our algorithms, as explained in Sections 3 and 10. Profitability is a consequence of the the preceding complicated definitions of values, value comparison, and strategy evaluation.

Theorem 20 (Profitability). *Every attractive switch is profitable:*

1. *it increases the value of the vertex where it is made, and*
2. *all other vertices either preserve or increase their values,*

i.e., the switch operator is monotone. □

7 Stability Implies Optimality

The Main Theorem 22 of this section guarantees that iterative improvement can terminate once a strategy with no attractive switches is found. In more general terms it states that every local optimum is global. This is one of the main motivations for the complex strategy evaluation definitions.

Definition 21. *Say that a strategy* σ *is* stable *if it does not have attractive switches with respect to* $\tau(\sigma)$*, an optimal counterstrategy of Player 1.* □

In Section 9 we show that all optimal counterstrategies provide for the *same* values. Thus stability of σ in the previous definition may be checked after computing any optimal counterstrategy $\tau(\sigma)$.

Theorem 22 (Stability). *Any stable strategy of Player 0 is optimal: vertices with loop values of even colors form the winning set of Player 0.* □

8 Uniqueness of Optimal Values

Theorem 22 does not guarantee that different stable (hence optimal) strategies provide for the same (or even comparable) vectors of values for the game vertices. The uniqueness of optimal values is however crucial for the subexponential complexity analysis of Sections 3 and 10, and is provided by the following

Theorem 23 (Uniqueness). *Any two stable strategies of Player 0 give the same values for all vertices of the game.* □

9 Computing Optimal Counterstrategies

Let G_σ be the game graph induced by a positional strategy σ of Player 0 (delete all edges of Player 0 not used by σ). Partition vertices of G_σ into classes L_t containing the vertices from which Player 1 can ensure the loop tint t, but cannot guarantee any worse loop tint. This can be done by using finite reachability in G_σ as follows. For each tint t in $\prec$-ascending order, check whether t can be reached from itself without passing any tint $t' > t$. If so, Player 1 can form a loop with t as major. Since the tints are considered in $\prec$-ascending order, t will be the best loop value Player 1 can achieve for all vertices from which t is reachable. Remove them from the graph, place them in class L_t, and proceed with the next tint.

For each class L_t, use dynamic programming to calculate the values of 1-optimal paths of different lengths from each vertex to t. For each vertex, the algorithm first computes the optimal color hit record (abbreviated *chr* in the algorithm) over all paths of length 0 to the loop major (∞ for each vertex except t). Then it calculates the color hit record of optimal paths of length one, length two, and so forth. It uses the values from previous steps in each step except the initial one.[4]

Algorithm 1: Computing path values within a class L_t.

PathValues(L_t)
(1) $t.chr[0] \leftarrow (0, \ldots, 0)$
(2) **foreach** vertex $v \in L_t$ except t
(3) $v.chr[0] \leftarrow \infty$
(4) **for** $i \leftarrow 1$ **to** $|L_t| - 1$
(5) **foreach** vertex $v \in L_t$ except t
(6) $v.chr[i] \leftarrow \min_{\prec_t}\{\text{AddColor}(t, v'.chr[i-1], \mathbf{t}(v)) : v' \in L_t \text{ is a successor of } v)\}$
(7) **foreach** vertex $v \in L_t$ except t
(8) $v.pathvalue \leftarrow \min_{\prec_t}\{(v.chr[i], i) : 0 \leq i < |L_t|\}$
(9) $t.pathvalue \leftarrow \min_{\prec_t}\{v.pathvalue : v \in L_t \text{ is a successor of } t\}$
(10) $t.pathvalue.pathlength \leftarrow t.pathvalue.pathlength + 1$

[4] The algorithm assumes the game is bipartite; in particular, t in line (9) cannot be a successor of itself. It can be straightforwardly generalized for the non-bipartite case.

The function AddColor takes a tint, a color hit record, and a second tint. If the second tint is bigger than the first one, then AddColor increases the position in the vector representing the color of the second tint. The function always returns ∞ when the second argument has value ∞.

The algorithm also handles non-binary games.

Lemma 24 (Algorithm Correctness). *The algorithm correctly computes values of optimal paths. Moreover:*

1. *optimal paths are simple;*
2. *the values computed are consistent with an actual positional strategy that guarantees loop value t.* □

Lemma 25 (Algorithm Complexity). *The algorithm for computing an optimal counterstrategy runs in time $O(|V| \cdot |E| \cdot k)$, where $|V|$ is the number of vertices of the graph, $|E|$ is the number of edges, and k is the number of colors.*

10 Kalai-Style Randomization for Games with Unbounded Outdegree

As discussed in Section 3, any non-binary parity game reduces to a binary one, and the Ludwig-style algorithm applies. However, the resulting complexity gets worse and may become exponential (rather than subexponential) due to a possibly quadratic blow-up in the number of vertices. In this section we describe a different approach relying on the randomization scheme of Kalai [9,6] used for Linear Programming. This results in a subexponential randomized algorithm directly applicable to parity games of arbitrary outdegree, without any preliminary translations. When compared with reducing to the binary case combined with the Ludwig-style algorithm of Section 3, the algorithm of this section provides for a better complexity when the total number of edges is roughly $\Omega(n \log n)$.

Games, Subgames, and Facets. Let $\mathcal{G}(d, m)$ be the class of parity games with vertices of Player 0 partitioned into two sets U_1 of outdegree one and U_2 of an arbitrary outdegree $\delta(v) \geq 1$, with $|U_2| = d$ and $m \geq \sum_{v \in U_2} \delta(v)$. Informally, d is the dimension (number of variables to determine), and m is a bound on the number of edges (constraints) to choose from. The numbers of vertices and edges of Player 1 are unrestricted.

Given a game $G \in \mathcal{G}(d, m)$, a vertex $v \in U_2$ of Player 0, and an edge e leaving v, consider the (sub)game F obtained by fixing e and deleting all other edges leaving v. Obviously, $F \in \mathcal{G}(d-1, m-\delta(v))$ and also, by definition, $F \in \mathcal{G}(d, m)$, which is convenient when we need to consider a strategy in the subgame F as a strategy in the full game G in the sequel. Call the game F a *facet* of G.

If σ is a positional strategy and e is an edge leaving a vertex v of Player 0, then we define $\sigma[e]$ as the strategy coinciding with σ in all vertices, except possibly v, where the choice is e. If σ is a strategy in $G \in \mathcal{G}(d, m)$, then a facet F is *σ-improving* if some *witness* strategy σ' in the game F (considered as a member of $\mathcal{G}(d, m)$) satisfies $\sigma \prec \sigma'$.

The Algorithm takes a game $G \in \mathcal{G}(d, m)$ and an initial strategy σ_0 as inputs, and proceeds in three steps.

1. Collect a set M containing r pairs (F, σ) of σ_0-improving facets F of G and corresponding witness strategies $\sigma \succ \sigma_0$.
   (The parameter r specified later depends on d and m. Different choices of r give different algorithms. The subroutine to find σ_0-improving facets is described below. This subroutine may find an optimal strategy in G, in which case the algorithm returns it immediately.)
2. Select one pair $(F, \sigma_1) \in M$ uniformly at random. Find an optimal strategy σ in F by applying the algorithm recursively, taking σ_1 as the initial strategy.[5]
3. If σ is an optimal strategy also in G, then return σ. Otherwise, let σ' be a strategy differing from σ by an attractive switch. Restart from step 1 using the new strategy σ' and the same game $G \in \mathcal{G}(d, m)$.

The algorithm terminates because each solved subproblem starts from a strictly better strategy. It is correct because it can only terminate by returning an optimal strategy.

How to Find Many Improving Facets. In step 1 the algorithm above needs to find either r different σ_0-improving facets or an optimal strategy in G. To this end we construct a sequence $(G^0, G^1, \ldots, G^{r-d})$ of games, with $G^i \in \mathcal{G}(d, d+i)$. All the $d+i$ facets of G^i are σ_0-improving; we simultaneously determine the corresponding witness strategies σ^j optimal in G^j. The subroutine returns r facets of G, each one obtained by fixing one of the r edges in $G^{r-d} \in \mathcal{G}(d, r)$. All these are σ_0-improving by construction.

Let e be the target edge of an attractive switch from σ_0. (If no attractive switch exists, then σ_0 is optimal in G and we are done.) Set G^0 to the game where all choices are fixed as in $\sigma_0[e]$, and all other edges of Player 0 in G are deleted. Let σ^0 be the unique, hence optimal, strategy $\sigma_0[e]$ in G^0. Fixing any of the d edges of σ^0 in G defines a σ_0-improving facet of G with σ^0 as a witness.

To construct G^{i+1} from G^i, let e be the target edge of an attractive switch from σ^i in G. (Note that σ^i is optimal in G^i but not necessarily in the full game G. If it is, we terminate.) Let G^{i+1} be the game G^i with e added, and compute σ^{i+1} as an optimal strategy in G^{i+1}, by a recursive application of the algorithm above. Note that fixing any of the $i+1$ added target edges defines a σ_0-improving facet of G. Therefore, upon termination we have r such facets.

Complexity Analysis. The following recurrence bounds the expected number of calls to the algorithm solving a game in $\mathcal{G}(d, m)$ in the worst case:

$$T(d, m) \leq \sum_{i=d}^{r} T(d, i) + T(d-1, m-2) + \frac{1}{r} \sum_{i=1}^{r} T(d, m-i) + 1$$

[5] Rather than computing all of M, we may select a random number $x \in \{1, \ldots, r\}$ before step 1 and compute only x improving facets. This is crucial in order for the computed sequence of strategies to be strictly improving, and saves some work.

The first term represents the work of finding r different σ_0-improving facets in step 1. The second term comes from the recursive call in step 2. [6] The last term accounts for step 3 and can be understood as an average over the r equiprobable choices made in step 2, as follows. All facets of G are partially ordered by the values of their optimal strategies (although this order is unknown to the algorithm). Optimal values in facets are unique by Theorem 23, and the algorithm visits only improving strategies. It follows that all facets possessing optimal strategies that are worse, equal, or incomparable to the strategy σ of step 2 will never be visited in the rest of the algorithm. In the *worst* case, the algorithm selects the r worst possible facets in step 1. Thus, in the worst case, in step 3 it solves a game in $\mathcal{G}(d, m-i)$ for $i = 1, \ldots, r$, with probability $1/r$. This justifies the last term.

Kalai uses $r = \max(d, m/2)$ in step 1 to get the best solution of the recurrence. The result is subexponential, $m^{O(\sqrt{d/\log d})}$. By symmetry, we can choose to optimize a strategy of the player possessing fewer vertices.

Let n_i denote the number of vertices of player i. Since m is bounded above by the maximal number of edges, $(n_0 + n_1)^2$, and $d \leq \min(n_0, n_1)$, we get

$$\min\left\{2^{O\left((\log n_1)\cdot\sqrt{n_0/\log n_0}\right)}, 2^{O\left((\log n_0)\cdot\sqrt{n_1/\log n_1}\right)}\right\}$$

as the bound on the number of calls to the algorithm. Combining it with the bound on the maximal number of improving steps allowed by our measure yields

$$\min\left\{2^{O\left((\log n_1)\cdot\sqrt{n_0/\log n_0}\right)}, 2^{O\left((\log n_0)\cdot\sqrt{n_1/\log n_1}\right)}, O\left(n^3\cdot\left(\frac{n}{k}+1\right)^k\right)\right\}.$$

If $n_0 = O(poly(n_1))$ and $n_1 = O(poly(n_0))$ then this reduces to

$$\min\left\{2^{O\left(\sqrt{n_0\log n_0}\right)}, 2^{O\left(\sqrt{n_1\log n_1}\right)}, O\left(n^3\cdot\left(\frac{n}{k}+1\right)^k\right)\right\}.$$

These are the bounds on the number of recursive calls to the algorithm. Within each recursive call, the auxiliary work is dominated by time to compute a strategy value, multiplying the running time by $O(n \cdot |E| \cdot k)$.

Acknowledgements. We thank anonymous referees for valuable remarks and suggestions.

References

1. H. Björklund, S. Sandberg, and S. Vorobyov. A discrete subexponential algorithm for parity games. Technical Report 2002-026, Department of Information Technology, Uppsala University, September 2002. `http://www.it.uu.se/research/reports/`.

[6] Actually, if δ is the outdegree in the vertex where we fix an edge, then the second term is $T(d-1, m-\delta)$. We consider the worst case of $\delta = 2$.

2. A. Browne, E. M. Clarke, S. Jha, D. E Long, and W Marrero. An improved algorithm for the evaluation of fixpoint expressions. *Theor. Comput. Sci.*, 178:237–255, 1997. Preliminary version in CAV'94, LNCS'818.
3. E. A. Emerson. Model checking and the Mu-calculus. In N. Immerman and Ph. G. Kolaitis, editors, *DIMACS Series in Discrete Mathematics*, volume 31, pages 185–214, 1997.
4. E. A. Emerson, C. Jutla, and A. P. Sistla. On model-checking for fragments of μ-calculus. In *Computer Aided Verification, Proc. 5th Int. Conference*, volume 697, pages 385–396. Lect. Notes Comput. Sci., 1993.
5. E. A. Emerson and C. S. Jutla. Tree automata, μ-calculus and determinacy. In *Annual IEEE Symp. on Foundations of Computer Science*, pages 368–377, 1991.
6. Goldwasser. A survey of linear programming in randomized subexponential time. *SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory)*, 26:96–104, 1995.
7. V. A. Gurvich, A. V. Karzanov, and L. G. Khachiyan. Cyclic games and an algorithm to find minimax cycle means in directed graphs. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 28(5):85–91, 1988.
8. M. Jurdzinski. Small progress measures for solving parity games. In *17th STACS*, volume 1770 of *Lect. Notes Comput. Sci.*, pages 290–301. Springer-Verlag, 2000.
9. G. Kalai. A subexponential randomized simplex algorithm. In *24th ACM STOC*, pages 475–482, 1992.
10. W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and Computation*, 117:151–155, 1995.
11. C. Papadimitriou. Algorithms, games, and the internet. In *ACM Annual Symposium on Theory of Computing*, pages 749–753. ACM, July 2001.
12. V. Petersson and S. Vorobyov. A randomized subexponential algorithm for parity games. *Nordic Journal of Computing*, 8:324–345, 2001.
13. N. Pisaruk. Mean cost cyclical games. *Mathematics of Operations Research*, 24(4):817–828, 1999.
14. H Seidl. Fast and simple nested fixpoints. *Information Processing Letters*, 59(3):303–308, 1996.
15. J. Vöge and M. Jurdzinski. A discrete strategy improvement algorithm for solving parity games. In *CAV'00: Computer-Aided Verification*, volume 1855 of *Lect. Notes Comput. Sci.*, pages 202–215. Springer-Verlag, 2000.
16. U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.*, 158:343–359, 1996.

Paper II

Complexity of Model Checking by Iterative Improvement: The Pseudo-Boolean Framework*

Henrik Björklund, Sven Sandberg, and Sergei Vorobyov

Information Technology Department, Box 337, Uppsala University,
751 05 Uppsala, Sweden
{henrikbj,svens}@it.uu.se, vorobyov@csd.uu.se

Abstract. We present several new algorithms as well as new lower and upper bounds for optimizing functions underlying infinite games pertinent to computer-aided verification.

1 Introduction

Infinite two-person adversary full information games are a well established framework for modeling interaction between a system and its environment. A correct system can be naturally interpreted as a player possessing a winning strategy against any strategy of the malicious environment. Respectively, a verification process can be considered as a proof that a system does possess such a strategy. If the system loses, then a winning strategy for the environment encompasses possible system improvements. During the last decades a substantial progress has been achieved both on fitting diverse approaches to computer-aided verification into the game-theoretic paradigm and, simultaneously, on developing efficient algorithms for solving games, i.e., determining the winner and its strategy [6,10,13,18,22,31]. A naturally appealing approach to solving games [12,21,31] consists in taking an initial strategy of the system and gradually 'improving' it, since a non-optimal strategy is outperformed by some counterstrategy of the environment. This allows for improving either the strategy, or the system, or both. In this paper we address the complexity of such an approach in an abstract model based on pseudo-Boolean functions possessing essential properties of games.

Game theory suggests, for numerous types of games, a nice characterization of the optimal solutions for behaviors of rational players, the so-called *Nash equilibria*. The 'only' remaining problem left widely open is: '*What is the exact computational complexity of finding Nash equilibria (optimal strategies) in two-person games with finitely many strategies? Could such games be solved in polynomial time?*' This is a fundamental problem, the solution to which determines whether or not, and to what extent game theory will be applicable

* Supported by Swedish Research Council Grants "Infinite Games: Algorithms and Complexity" and "Interior-Point Methods for Infinite Games".

M. Broy and A.V. Zamulin (Eds.): PSI 2003, LNCS 2890, pp. 381–394, 2003.

to solving and optimizing real large-scale games emerging from practical verification of complex reactive systems. The problem is known to belong to the complexity class NP∩coNP (therefore, most probably not NP-complete, luckily!) but is *not known to be polynomial time solvable*. The best currently known algorithms are *exponential* (sometimes *subexponential*). Clearly, superpolynomial algorithms will not allow for practical solutions of the real-life verification games, no matter how fast the progress in hardware continues. Consequently, this problem is one of the most practical and (together with the NP versus P) most fundamental problems in the foundations of computing, complexity theory, and automata theory [27].

In this paper we address the efficiency of iterative improvement algorithms for solving games in the following abstract setting. Consider a game where one of the players has n binary choices, e.g., selects whether to move left or right[1] in n places, and every combination of choices is given a cost, reflecting how good this strategy (combination of choices) is against a perfect adversary. This results in an n-dimensional valued Boolean hypercube, or a *pseudo-Boolean function*. Consequently, everything boils down to optimizing a pseudo-Boolean function. How fast can we optimize such a function? Despite its simplicity, the question appears extremely difficult for the classes of functions with special properties pertinent to games. Not surprisingly, there are not so many strong and general results of this kind in the literature. The best bound in Tovey's survey [30] is $O(2^{n/2})$; Mansour and Singh [25] suggest an $O(2^{0.773n})$ algorithm (for Markov decision processes), and we independently obtain an improved bound $O(2^{0.453n})$ for a similar algorithm in a more general setting [5,8]. Ludwig [24] suggested a subexponential $O(2^{\sqrt{n}})$ algorithm for binary simple stochastic games, which becomes exponential for games of unbounded (non-binary) outdegree. We suggested several [3,6,7,8] subexponential algorithms for games of arbitrary outdegrees.

The development of our theory is primarily motivated by the so-called *parity games*, which are infinite games played on finite directed bipartite leafless graphs, with vertices colored by integers. Two players alternate moving a pebble along edges. The goal of Player 0 is to ensure that the biggest color visited by the pebble infinitely often is even, whereas Player 1 tries to make it odd. The complexity of determining the winner in parity games, equivalent to the Rabin chain tree automata non-emptiness, as well as to the μ-calculus[2] model checking [13,14], is a fundamental open problem in complexity theory [27]. The problem belongs to NP∩coNP, but its PTIME-membership status remains widely open. *Discounted mean payoff* [34] and *simple stochastic games* [11,12], relevant for verification of probabilistic systems, are two other classes to which our theory applies [8].

We exploit the fundamental fact, apparently unnoticed before, that functions arising from such games possess extremely favorable structural properties, close

[1] The restriction to two successors is no loss of generality, since a game with non-binary choices may be reduced to binary. However, complexity gets worse, and this issue is dealt with in Section 9.

[2] One of the most expressive temporal logics of programs [13].

to the so-called *completely unimodal functions* [19,33,32,30]. These functions are well known in the field of pseudo-Boolean optimization [9,20], an established area of combinatorial optimization, in which local search [1,30] is one of the dominating methods. The complexity of local search for different classes of functions were carefully investigated [30,19,33]. However, all previously known algorithms for completely unimodal functions were *exponential* [30,9]. A fruitful idea came from linear programming. In the early 90's Kalai [23] and Matoušek, Sharir, Welzl [29,26] came up with strongly subexponential randomized algorithms for linear programming. Later Ludwig [24] adapted it to binary simple stochastic games, and we to binary parity games [28]. We also figured out that Kalai's and Sharir–Welzl's randomized schemes fit perfectly well for completely unimodal optimization [2,5,4] and parity games [28]. Later we succeeded to generalize and adapt those subexponential schemes to create a discrete subexponential algorithm for general (non-binary) parity games [6], and extend it to combinatorial structures (which we call *completely local-global*) more directly reflecting games [7,8,3].

In this paper we present new upper bounds on the number of iterations of various iterative improvement algorithms for optimization of *completely unimodal pseudo-boolean functions* (CUPBFs). Such functions possess several remarkable properties: 1) unique minimum and maximum on every subcube, 2) every vertex has a unique neighborhood improvement signature. The problem of optimizing CUPBFs has been studied before [19,33], but only few and weak bounds are known. Only a very restrictive subclass of decomposable CUPBFs is known to allow for polynomial time randomized optimization. The best known upper bounds are exponential, but it is conjectured that any CUPBF can be optimized in polynomial time [33].

We investigated and compared, both theoretically and practically[3], five algorithms for CUPBF optimization: the Greedy Single Switch Algorithm (GSSA), the Random Single Switch Algorithm (RSSA), the All Profitable Switches Algorithm (APSA), the Random Multiple Switches Algorithm (RMSA), and Kalai-Ludwig's Randomized Algorithm (KLRA). The GSSA and the RSSA have been studied by others in the context of CUPBF optimization. We first proposed the use of the APSA and RMSA in this context in [2]. It turns out that complete unimodality allows, knowing all improving neighbors of a vertex, to improve by simultaneously jumping towards all or some of them. (Actually any 0-1 linear combination of improving directions gives an improvement; this allows for non-local search algorithms that behave surprisingly well in practice.) The only nontrivial lower bound that has been shown is an exponential one for the GSSA [32]. Local search iterative improvement algorithms are appealing, easy to implement, and efficient in practice. But no nontrivial upper bounds are known for those algorithms, except for subclasses of all CUPBFs. We settle several such nontrivial bounds in this paper. The fifth algorithm, KLRA, was first proposed in [24] for solving simple stochastic games, with a subexponential upper bound

[3] Our results on practical evaluation and comparison of the algorithms can be found in [2,5,4].

on the expected running time. We show that it can be modified to solve the CUPBF optimization problem, still in expected subexponential time.

Outline of the paper. In Section 2 we recall the definitions and main results concerning completely unimodal functions. Section 3 describes five iterative improvement algorithms for completely unimodal functions. Section 4 is devoted to the random multiple switches algorithm, whereas Section 5 covers single greedy single random, and all profitable switches algorithms. Section 6 adds random sampling to the random multiple switches algorithm, while Section 7 does the same to all other algorithms. Section 8 describes the Kalai-Ludwig-style algorithm. Section 9 generalizes the previous results to completely local-global functions and presents the Sharir-Welzl-style algorithm for these functions.

2 Completely Unimodal Pseudo-Boolean Functions

Parity and simple stochastic games can be solved by maximizing appropriately defined functions on neighborhood graphs representing sets of strategies. The prototypical case of such neighborhood graphs is the Boolean hypercube, and the essential structure of games pertinent to optimization by iterative improvement is captured, in the first approximation, by completely unimodal functions on Boolean cubes [19,32,33,30]. Much of the theory we present can be understood and developed in terms of completely unimodal functions, and until Section 9 we concentrate on this case.

In general, parity and simple stochastic games require less restrictive neighborhood structures and functions. In [7] we succeeded to characterize them as a class we called *completely local-global* (CLG) functions, defined on Cartesian products of arbitrary finite sets, rather than on Boolean hypercubes. These functions, considered in Section 9, were crucial in our development [6,7] of subexponential algorithms for parity games with arbitrary outdegree. This is because reducing such games to binary ones quadratically increases the number of vertices. As a consequence, a straightforward reduction renders subexponential algorithms exponential [6].

Let $H(n)$ denote the *n-dimensional Boolean hypercube* $\{0,1\}^n$, for $n \in \mathbb{N}^+$. A *pseudo-Boolean function* is an injective[4] mapping $H(n) \to \mathbb{R}$ associating a real number to every n-dimensional Boolean vector. For $0 \leq k \leq n$, a *k-dimensional face* of $H(n)$, or a *k-face*, is a collection of Boolean vectors obtained by fixing $n-k$ arbitrary coordinates and letting the k remaining coordinates take all possible Boolean values. Faces of dimension 0 are called *vertices*, faces of dimension 1 are called *edges*. Faces of dimension $n-1$ are called *facets*. Two vertices that share an edge are called *neighbors*. Each vertex v in $H(n)$ has exactly n neighbors, forming the *standard neighborhood* of v on $H(n)$.

[4] The standard injectiveness restriction is lifted in Section 9.

Complete Unimodality. A pseudo-Boolean function f is called *completely unimodal* (CUPBF for short) if one of the following four equivalent conditions holds [19,33]:

1. f has a unique local minimum on each face,
2. f has a unique local maximum on each face,
3. f has a unique local minimum on each 2-face,
4. f has a unique local maximum on each 2-face.

Improving directions. Let $f : H(n) \to \mathbb{R}$ be a CUPBF and let v be a vertex on $H(n)$. Number the dimensions of $H(n)$ from 0 to $n-1$. For each $i \in \{0, 1, ..., n-1\}$, let v_i be the neighbor of v that is reached by moving in coordinate i from v. Let p_i be 1 if $f(v) < f(v_i)$, otherwise 0. Then we call $\overline{p} = [p_0, p_1, ..., p_{n-1}]$ the *vector of improving directions* (VID) of v under f.

In the sequel we will usually abuse terminology by identifying a function and a (valued) hypercube it is defined upon. By 'optimizing' we mean '*maximizing*'.

3 Five Iterative Improvement Algorithms

A *standard local-search improvement algorithm* starts in an arbitrary point v_0 of the hypercube $H(n)$ and iteratively improves by selecting a next iterate with a better value from a *polynomial* neighborhood $N(v_i)$ of the current iterate.

Specific instances of the standard algorithm are obtained when one fixes:

1. the neighborhood structure on $H(n)$,
2. the disciplines of selecting the initial point and the next iterate.

Two major local-search improvement algorithms, the *Greedy Single Switch Algorithm (GSSA)* and the *Random Single Switch Algorithm (RSSA)* have previously been investigated and used for optimizing CUPBFs [30]. We also investigate the *All Profitable Switches Algorithm (APSA)* and the *Random Multiple Switches Algorithm (RMSA)*. Strictly speaking, neither APSA, nor RMSA is a local-search algorithm. The first one operates with neighborhood structures which vary depending on the CUPBF being optimized. The second chooses the next iterate from a non-polynomially bounded (in general) neighborhood of the current iterate. Finally, we show how the *subexponential Kalai-Ludwig's Randomized Algorithm (KLRA)* for solving binary simple stochastic games can be modified to optimize CUPBFs, and show that it is subexponential for this problem as well, with expected worst case behavior $2^{O(\sqrt{n})}$. This algorithm has so far been unknown in the field of CUPBF optimization.

Greedy Single Switch Algorithm (GSSA). This is a local-search algorithm that at every iteration chooses the *highest-valued neighbor* of the current vertex as the next iterate. Recall that every vertex of $H(n)$ has exactly n neighbors (in the standard neighborhood). Unfortunately, this algorithm may take *exponentially many* steps to find the maximum of a CUPBF [32].

Random Single Switch Algorithm (RSSA). This is a local-search algorithm that at every iteration chooses uniformly at random one of the *higher-valued neighbors* of the current vertex as the next iterate. This algorithm may also take *exponentially many* iterations to find the global maximum. Although its expected running time for general CUPBFs is unknown, Williamson Hoke [33] has shown, using a proof technique due to Kelly, that the RSSA has expected quadratic running time on any *decomposable* hypercube. Call a facet F an *absorbing facet* if no vertex on F has a higher-valued neighbor that is not on F. A hypercube is called *decomposable* iff it has dimension 1 or has an absorbing facet that is decomposable. This result, together with the fact that in a CUPBF there is a short improving path from any vertex to the maximum (i.e., Hirsch conjecture holds), form grounds for the polynomial time optimization conjecture for CUPBFs [33] (currently open).

All Profitable Switches Algorithm (APSA). The All Profitable Switches Algorithm (APSA) at every iteration computes the VID s of a current iterate v and inverts the bits of v in positions where s has ones to get the new iterate v' (i.e., $v' := v$ `XOR` s.)

This algorithm may also be seen as a local-search algorithm, but the structure of the neighborhood is not fixed a priori (as for GSSA and RSSA), but rather changes for each CUPBF.

APSA is a stepwise improvement algorithm for CUPBFs because the current iterate v is the unique global minimum on the face defined by fixing all coordinates corresponding to zeros in the VID s. Therefore, the next iterate v' (which belongs to the same face) has a better function value.

Random Multiple Switches Algorithm (RMSA). Like APSA, the Random Multiple Switches Algorithm (RMSA) at every iteration computes the VID s of a current iterate v. However, to get the next iterate v' RMSA inverts bits in v corresponding to a nonempty subset s' of the nonzero bits in s, chosen uniformly at random (i.e., $v' := v$ `XOR` s'.)

RMSA is a stepwise improvement algorithm for CUPBFs because the current point v is the unique global minimum on the face defined by fixing the coordinates in v corresponding to zeros in s'. Thus the next iterate v', belonging to this face, must have a better function value. Note that RMSA selects at random from a neighborhood that may be *exponentially big* in the dimension. So, strictly speaking, this is not a polynomial local-search improvement algorithm.

Kalai-Ludwig's Randomized Algorithm (KLRA). In a major breakthrough Kalai [23] suggested the first *subexponential* randomized simplex algorithm for linear programming. Based on Kalai's ideas, Ludwig [24] suggested the first *subexponential* randomized algorithm for simple stochastic games with binary choices. We show that Ludwig's algorithm without any substantial modifications performs correctly and with the same expected worst-case complexity

$2^{O(\sqrt{n})}$ for optimizing CUPBFs. Kalai-Ludwig's algorithm may informally be described as follows.

1. Start at any vertex v of $H(n)$.
2. Choose at random a coordinate $i \in \{1, \ldots, n\}$ (not chosen previously).
3. Apply the algorithm recursively to find the best point v' with the same i-th coordinate as v_i.
4. If v' is not optimal (has a better neighbor), invert the i-th component in v', set $v := v'$ and repeat.

4 The Random Multiple Switches Algorithm

The only two algorithms that were previously studied for the CUPBF optimization problem are: 1) the GSSA, 2) the RSSA. The GSSA makes an exponential number of steps on CUPBFs constructed in [32]. If extremely unlucky, the RSSA can make an exponential number of steps on CUPBFs generated by Klee-Minty cubes, but nevertheless its expected runtime on such cubes is $O(n^2)$, quadratic in the number of dimensions. The same expected quadratic upper bound holds for the RSSA running on the class of the so-called *decomposable* CUPBFs (of which Klee-Minty's form a proper subclass) [33, p. 77-78]. Besides these results, there are no other known: 1) nontrivial lower bounds for the problem, 2) better upper bounds for any specific algorithms. Nevertheless, [33, p. 78] conjectures that the RSSA is (expected) polynomial on *all CUPBFs*.

It is worth mentioning, however, that for any CUPBF with optimum v^* and any initial vertex v_0 there is always an improving path from v_0 to v^* of length $hd(v_0, v^*)$, the Hamming distance between v_0 and v^*. Therefore, the 'Hirsch conjecture' about short paths to the optimum holds for CUPBFs (thus the potential existence of 'clever' polynomial time algorithms is not excluded).

Simultaneously, nothing except this 'trivial linear' $\Omega(n)$ lower bound is currently known for the CUPBF optimization problem. For the broader classes of all pseudo-Boolean and all unimin functions[5] there are the $\Omega(2^v/\sqrt{n})$ and the $\Omega(2^v/n^{1.5})$ lower bounds, respectively, for any deterministic algorithms, and the $\Omega(2^{n/2} \cdot n)$ lower bound for any randomized algorithms [30, Thm. 14, p. 66, Coroll. 21, p. 71, Coroll. 19, p. 69].

In view of the above results our new, presented in this section, $O(2^{0.773n}) = O(1.71^n)$ upper bound for our new RMSA (Randomized Multiple Switch Algorithm)[6] should be considered an improvement. The key step consists in exploiting complete unimodality and the next simple lemma, giving a lower bound on the per-step improvement of the target function value by the RMSA, exponential in the number of improving directions in the current vertex.

[5] Possessing a unique minimum.

[6] Which is not local-search type. Maybe this is the reason it was not considered in general pseudo-Boolean optimization; it is only correct for CUPBFs.

Lemma 1. *The expected function value improvement of the RMSA step from a vertex v with $i > 0$ improving directions is at least 2^{i-1}.*

This makes the RMSA attractive: we can guarantee that the expected 'value jump' in each step is relatively high, provided, there are many improving directions. If the average number of improving directions per vertex visited during a run of the RMSA were at least k, the algorithm would terminate in at most $O(2^{n-k})$ steps. In particular, $n/2$ average improving directions would give an $O(2^{n/2})$ upper bound. Can we guarantee any nontrivial lower bound on the number of improving directions in any run of the RMSA? Fortunately, the fundamental property that in every completely unimodal cube every possible bit vector of improving directions is present exactly once [33, p. 75-76] allows us to do this. The proof of the following theorem assumes the worst (and seemingly unrealizable) case that the algorithm is always unlucky, selecting a vertex with the *fewest* possible number of improving directions, and the cube generated by these direction is numbered by the *smallest* possible successors of the current value. This forces the worst case and settles an upper bound on the number of RMSA iterations in the worst case.

Theorem 1. *The expected number of iterations made by the RMSA on an n-dimensional CUPBF is less than $2^{0.773n} = 1.71^n$, for sufficiently large n.*

After we obtained the bound from Theorem 1, we were pointed out that a similar bound for the same algorithm, but applied to Markov decision processes, was proved earlier in [25]. Later it became clear that our result is stronger, after we succeeded to reduce simple stochastic games to CLG-functions and CLG-functions to CU-functions [7]. Moreover, we substantially improve the bound from Theorem 1 below $2^{n/2}$ in Section 6 (for a variant of the algorithm).

5 Single Greedy, Single Random, and All Profitable Switches Algorithms

We start with a simple lower bound on the per-step improvement for all three algorithms. This bound is weaker than for the RMSA. Consequently, the upper bounds we can settle for these algorithms are weaker. Nevertheless, the practical behavior of these algorithms shown in experiments [2,5,4] make them very attractive.

Proposition 1. *When the APSA, the GSSA, or the RSSA makes a switch in a vertex with i improving directions the value of the target function increases at least by i (by expected value at least $i/2$ for the RSSA).*

Theorem 2. *For any $0 < \varepsilon < 1$ the APSA, the GSSA,, and the RSSA make fewer than $2^{n-(1-\varepsilon)\log(n)}$ iterations on any n-dimensional CUPBF, for sufficiently large n.*

Remark 1. 1) Note that it is stronger than claiming 'fewer than 2^{n-c} (for a constant $c > 0$)', but 2) weaker than 'fewer than 2^{cn} (for a constant $0 < c < 1$)'. 3) This upper bound is better than the $5/24 \cdot 2^n - 45$ lower bound for any local improvement algorithm on a uniminmax function [30, Thm 23, p. 72].

6 Adding Random Sampling to the RMSA

The RMSA can be considerably improved by adding random sampling. If we start the RMSA from a 'good' vertex, with a value close to the optimum, the RMSA guarantees a reasonably short run before finding it, as is shown in Section 4. The trick is to select a good initial vertex by making an optimal number of random probes picking the one with the best value, and to minimize the overall running time. We call the modified algorithm the RMSA-RS and parameterize it by the number of randomly sampled vertices. For this modified algorithm, a better upper bound can be shown, when we choose the parameter optimally:

Theorem 3. *The RMSA-RS can be parameterized in such a way that its expected running time on an n-dimensional CUPBF is* $O(2^{0.453n}) = O(1.37^n)$.

As described, the RMSA-RS always makes $2^{0.453n}$ random samplings, before starting any optimizations, so its expected best case is $\Omega(2^{0.453n})$. Although other single or multiple switch algorithms we consider have worse known upper bounds, they show much better practical behavior.

7 Adding Random Sampling to the APSA, the GSSA, and the RSSA

As we saw in the previous section, a better bound can be proved for the RMSA when random sampling is added. In this section we show that random sampling also allows for better bounds for the APSA, the GSSA, and the RSSA. The bounds are not as strong, however, as the one for the RMSA with random sampling.

Theorem 4. *For any* $0 < \varepsilon < 1/2$ *the All Profitable Switches, the Greedy Single Switch, and the Randomized Single Switch Algorithms with initial random sampling of* $2^{\frac{n}{2}-(\frac{1}{2}-\varepsilon)\log(n)}$ *vertices make less than* $2^{\frac{n}{2}-(\frac{1}{2}-\varepsilon)\log(n)}$ *iterations on any n-dimensional CUPBF, for sufficiently large n.*

Corollary 1. *With the initial random sampling of* $2^{\frac{n}{2}-(\frac{1}{2}-\varepsilon)\log n}$ *vertices, for any* $\varepsilon \in (0, 1/2)$*, the APSA, the GSSA, and the RSSA have running times (expected in the case of RSSA) that are* $O(2^{\frac{n}{2}-(\frac{1}{2}-\varepsilon)\log n})$.

8 Kalai-Ludwig's Algorithm for CUPBFs

We were the first to observe [2] that the Kalai-Ludwig's Randomized Algorithm (KLRA) initially designed for linear programming [23] and later adapted for simple stochastic games [24], works perfectly for CUPBF optimization, also providing *subexponential* expected running time [7]. Modified for CUPBF optimization the KLRA is shown below.

Algorithm 1: Kalai-Ludwig's Algorithm for CUPBFs
KLRA(CUPBF H, initial vertex v_0)
(1) **if** $dim(H) = 0$
(2) **return** v_0
(3) **else**
(4) choose a random facet F of H containing v_0
(5) $v^* \leftarrow$ KLRA(F, v_0)
(6) **if** neighbor u of v^* on $H \setminus F$ is better than v^*
(7) **return** KLRA($H \setminus F$, u)
(8) **else**
(9) **return** v^*

It turns out that the algorithm is correct and terminating:

Theorem 5. *For every CUPBF, KLRA terminates and returns the global maximum.*

The following adjusts the theorem and proof in [24] to the case of CUPBFs.

Theorem 6. *The expected running time of KLRA on a CUPBF is* $2^{O(\sqrt{n})}$.

Our experiments on randomly generated CUPBFs [2,5,4] indicate that KLRA performs better than its theoretical subexponential upper bound. Surprisingly,

[7] **Added in proof:** Bernd Gärtner pointed out to us that he came up to similar results in [16] (journal version [17]), also building on the ideas of Kalai, Matoušek, Sharir, and Welzl. Rather than using the standard terminology of *completely unimodal functions* [19,32,33,30], B. Gärtner employs a less common and quite inexpressive term *abstract optimization functions* instead, and this unfortunate choice partially explains why his results have not become widely known in pseudo-Boolean optimization. It should be noted, however, that our algorithms and analysis are more general (see the next section), since they apply not only for Boolean cubes, but to hyperstructures (products of simplices) as well. We also allow the functions to take values in *partially* rather than *totally* ordered sets. Additionally, we relax and do not stipulate the "unique sink on every subcube" property. All this is more adequate for games and provides for better complexity bounds. We thank B. Gärtner for his pointers and observations.

the other four algorithms considerably outperform KLRA, although no subexponential bounds are currently known for them. It is reasonable to believe that for some of the algorithms there may be subexponential or better upper bounds. In this work we proved the first nontrivial upper bounds for these algorithms. The bounds we showed are still exponential, but not expected to be tight. Rather, they are to be viewed as a first step towards settling the precise complexity of these algorithms on CUPBFs.

9 Completely Local-Global (CLG) Functions

So far we restricted our attention to binary games. Although every non-binary game can be reduced to a binary one, the resulting number of vertices is proportional to the size of the initial graph. This may give a quadratic blow-up in the number of vertices (e.g., for graphs of linear outdegree), and the $2^{O(\sqrt{n})}$ bound becomes exponential, since n is quadratic in the initial number of vertices. In this section we show how to apply more sophisticated algorithms directly on non-binary structures, maintaining the subexponential bounds. We start with a non-binary generalization of hypercubes.

Definition 1 (Hyperstructure). *For each $j \in \{1, \ldots, d\}$ let $\mathcal{P}_j = \{e_{j,1}, \ldots, e_{j,\delta_j}\}$ be a finite set. Call $\mathcal{P} = \prod_{j=1}^{d} \mathcal{P}_j$ a d-dimensional* hyperstructure*, or* structure *for short.* □

A *substructure* $\mathcal{P}' \subseteq \mathcal{P}$ is a product $\mathcal{P}' = \prod_{j=1}^{d} \mathcal{P}'_j$, where $\emptyset \neq \mathcal{P}'_j \subseteq \mathcal{P}_j$ for all j. If each $\mathcal{P}'_j$ has only two elements, then identify $\mathcal{P}'$ with H. Call $\mathcal{P}'$ a *facet* of $\mathcal{P}$ if there is a j such that $\mathcal{P}'_k = \mathcal{P}_k$ for all $k \neq j$, and $\mathcal{P}'_j$ has only one element. Say that $x, y \in \mathcal{P}$, are *neighbors* iff they differ in only one coordinate. Thus each $x \in \mathcal{P}$ has exactly $\sum_{j=1}^{d}(\delta_j - 1)$ neighbors. The neighbor relation defines a graph with elements of the hyperstructure as nodes, allowing us to talk about paths and distances in the hyperstructure. A structure $\mathcal{P} = \prod_{j=1}^{d} \mathcal{P}_j$ has d dimensions and $n = \sum_{j=1}^{d} \delta_j$ facets.

Throughout this section, let $\mathcal{D}$ be some partially ordered set. We now consider functions defined on $\mathcal{P}$ with values in $\mathcal{D}$. Functions with partially ordered codomains are better suited for games [7].

A *local maximum* of a function $f : \mathcal{P} \to \mathcal{D}$ is a vertex in $\mathcal{P}$ with value bigger than or equal to all its neighbors. A *global maximum* is a vertex with a function value bigger than or equal to the values of all other vertices. In particular, any global maximum is comparable with all other vertices. Local and global minima are defined symmetrically.

Definition 2 (CLG-function on a hyperstructure). *Let $f : \mathcal{P} \to \mathcal{D}$ be a function such that all neighbors have comparable values. Say that f is a* CLG-function *if the following properties hold on every substructure of $\mathcal{P}$:*

1. *any local maximum of f is also global;*
2. *any local minimum of f is also global;*
3. *any two local maxima are connected by a path of local maxima;*

4. any two local minima are connected by a path of local minima.

By a CLG-structure *we mean a CLG-function together with the underlying hyperstructure.*

We note in passing that CLG-functions can be defined on hypercubes as well (with the same four properties), and CUPBFs can be defined: 1) with a partial set $\mathcal{D}$ as co-domain, and 2) on hyperstructures. The relaxation of the co-domain from $\mathbb{R}$ to $\mathcal{D}$ is an advantage (the essential properties rely only on the order of neighbors), important for the applications to games. Many properties are also carried on from hypercubes to hyperstructures, usually with a modified formulation. Multiple switches algorithms, like APSA and RMSA, generalize to hyperstructures as well.

As shown in [7], Algorithm 2 optimizes CLG-functions on hyperstructures in expected time $2^{O(\sqrt{d \log(n/\sqrt{d})} + \log n)}$, where $n = \sum_{j=0}^{d} \delta_j$. For games, where the maximal outdegree is d and $n = O(d^2)$, the bound collapses to $2^{O(\sqrt{d \log d})}$. The algorithm is adapted from the linear programming algorithm by Matoušek, Sharir and Welzl [29,26]. If $\mathcal{P}$ happens to be binary, the algorithm coincides with Kalai-Ludwig's algorithm; see Section 8.

Algorithm 2: MSW-Style Optimization Algorithm
OPTIMIZE(CLG-structure $\mathcal{P}$, initial vertex v_0)
(1) **if** $\mathcal{P} = v_0$
(2) **return** v_0
(3) **else**
(4) choose a random facet F of $\mathcal{P}$, not containing v_0
(5) $v^* \leftarrow$ OPTIMIZE($\mathcal{P} \setminus F, v_0$)
(6) **if** neighbor u of v^* on F is better than v^*
(7) **return** OPTIMIZE(F, u)
(8) **else**
(9) **return** v^*

The importance of CLG-functions stems from the fact that the strategy measures from [24,31,6] are indeed CLG-functions, as shown in [7]. Moreover, CLG-functions on hypercubes can be transformed to CUPBFs by introducing an artificial order on unordered neighbors [7]. Also, CLG-functions on hyperstructures can be transformed analogously to CUPBF-like functions on hyperstructures. Finally, we showed in [7] that any CLG-function reduces to an *LP-type problem*, an abstract framework for linear programming [29] and problems in computational geometry [26,15]. These reductions provide a strong argument for CLG-functions as a link between well-studied areas that look very different on the surface.

CLG-functions simultaneously allow for subexponential and multiple switch algorithms. Although the latter currently have worse known upper bounds, good practical behavior of such algorithms, confirmed by experiments, make them

very attractive and competitive [2,5,4]. We hope that a thorough investigation of random walks on the favorable CU- or CLG-structures will allow for improved bounds for multiple switching algorithms, both on the average and in the worst case.

References

1. E. Aarts and J. K. Lenstra, editors. *Local Search in Combinatorial Optimization*. John Wiley & Sons, 1997.
2. H. Björklund, V. Petersson, and S. Vorobyov. Experiments with iterative improvement algorithms on completely unimodal hypercubes. Technical Report 2001-017, Information Technology/Uppsala University, September 2001. `http://www.it.uu.se/research/reports/`.
3. H. Björklund and S. Sandberg. Algorithms for combinatorial optimization and games adapted from linear programming. In B. ten Cate, editor, *Proceedings of the Eighth ESSLLI Student Session*, 2003. to appear.
4. H. Björklund, S. Sandberg, and S. Vorobyov. An experimental study of algorithms for completely unimodal optimization. Technical Report 2002-030, Department of Information Technology, Uppsala University, October 2002. `http://www.it.uu.se/research/reports/`.
5. H. Björklund, S. Sandberg, and S. Vorobyov. Optimization on completely unimodal hypercubes. Technical Report 018, Uppsala University / Information Technology, May 2002. `http://www.it.uu.se/research/reports/`.
6. H. Björklund, S. Sandberg, and S. Vorobyov. A discrete subexponential algorithm for parity games. In H. Alt and M. Habib, editors, *20th International Symposium on Theoretical Aspects of Computer Science, STACS'2003*, volume 2607 of *Lecture Notes in Computer Science*, pages 663–674, Berlin, 2003. Springer-Verlag. Full preliminary version: TR-2002-026, Department of Information Technology, Uppsala University, September 2002.
7. H. Björklund, S. Sandberg, and S. Vorobyov. On combinatorial structure and algorithms for parity games. Technical Report 2003-002, Department of Information Technology, Uppsala University, January 2003. `http://www.it.uu.se/research/reports/`.
8. H. Björklund, S. Sandberg, and S. Vorobyov. Randomized subexponential algorithms for parity games. Technical Report 2003-019, Department of Information Technology, Uppsala University, April 2003. `http://www.it.uu.se/research/reports/`.
9. E. Boros and P. L. Hammer. Pseudo-boolean optimization. Technical Report RRR 48-2001, RUTCOR Rutger Center for Operations Research, 2001.
10. E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2000.
11. A. Condon. The complexity of stochastic games. *Information and Computation*, 96:203–224, 1992.
12. A. Condon. On algorithms for simple stochastic games. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 13:51–71, 1993.
13. E. A. Emerson. Model checking and the Mu-calculus. In N. Immerman and Ph. G. Kolaitis, editors, *DIMACS Series in Discrete Mathematics*, volume 31, pages 185–214, 1997.
14. E. A. Emerson and C. S. Jutla. Tree automata, μ-calculus and determinacy. In *Annual IEEE Symp. on Foundations of Computer Science*, pages 368–377, 1991.

15. B. Gärtner. A subexponential algorithm for abstract optimization problems. *SIAM Journal on Computing*, 24:1018–1035, 1995.
16. B Gärtner. Combinatorial linear programming: Geometry can help. In *RANDOM'98*, volume 1518 of *Lect. Notes Comput. Sci.*, pages 82–96, 1998.
17. B Gärtner. The random-facet simplex algorithm on combinatorial cubes. *Random Structures and Algorithms*, 20(3):353–381, 2002.
18. E. Grädel, W. Thomas, and T. Wilke, editors. *Automata Logics and Infinite Games. A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.
19. P. L. Hammer, B. Simeone, Th. M. Liebling, and D. De Werra. From linear separability to unimodality: a hierarchy of pseudo-boolean functions. *SIAM J. Disc. Math.*, 1(2):174–184, 1988.
20. P. Hansen, B. Jaumard, and V. Mathon. Constrained nonlinear 0-1 programming (state-of-the-art survey). *ORSA Journal on Computing*, 5(2):97–119, 1993.
21. A. J. Hoffman and R. M. Karp. On nonterminating stochastic games. *Management Science*, 12(5):359–370, 1966.
22. M. Jurdziński. Small progress measures for solving parity games. In H. Reichel and S. Tison, editors, *17th STACS*, volume 1770 of *Lect. Notes Comput. Sci.*, pages 290–301. Springer-Verlag, 2000.
23. G. Kalai. A subexponential randomized simplex algorithm. In *24th ACM STOC*, pages 475–482, 1992.
24. W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and Computation*, 117:151–155, 1995.
25. Y. Mansour and S. Singh. On the complexity of policy iteration. In *Uncertainty in Artificial Intelligence'99*, 1999.
26. J. Matoušek, M. Sharir, and M. Welzl. A subexponential bound for linear programming. In *8th ACM Symp. on Computational Geometry*, pages 1–8, 1992.
27. C. Papadimitriou. Algorithms, games, and the internet. In *ACM Annual Symposium on Theory of Computing*, pages 749–753. ACM, July 2001.
28. V. Petersson and S. Vorobyov. A randomized subexponential algorithm for parity games. *Nordic Journal of Computing*, 8:324–345, 2001.
29. M. Sharir and E. Welzl. A combinatorial bound for linear programming and related problems. In *9th Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 577 of *Lecture Notes in Computer Science*, pages 569–579, Berlin, 1992. Springer-Verlag.
30. C. A. Tovey. Local improvement on discrete structures. In E. Aarts and Lenstra J. K., editors, *Local Search in Combinatorial Optimization*, pages 57–89. John Wiley & Sons, 1997.
31. J. Vöge and M. Jurdziński. A discrete strategy improvement algorithm for solving parity games. In E. A. Emerson and A. P. Sistla, editors, *CAV'00: Computer-Aided Verification*, volume 1855 of *Lect. Notes Comput. Sci.*, pages 202–215. Springer-Verlag, 2000.
32. D. Wiedemann. Unimodal set-functions. *Congressus Numerantium*, 50:165–169, 1985.
33. K. Williamson Hoke. Completely unimodal numberings of a simple polytope. *Discrete Applied Mathematics*, 20:69–81, 1988.
34. U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.*, 158:343–359, 1996.

Paper III

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT
Theoretical Computer Science 310 (2004) 365–378

Theoretical Computer Science

www.elsevier.com/locate/tcs

Memoryless determinacy of parity and mean payoff games: a simple proof ☆

Henrik Björklund, Sven Sandberg, Sergei Vorobyov*

Information Technology Department, Uppsala University, Box 337, SE-751 05 Uppsala, Sweden

Received 25 March 2003; accepted 29 July 2003
Communicated by A.S. Fraenkel

Abstract

We give a simple, direct, and constructive proof of memoryless determinacy for parity and mean payoff games. First, we prove by induction that the finite duration versions of these games, played until some vertex is repeated, are determined and both players have memoryless winning strategies. In contrast to the proof of Ehrenfeucht and Mycielski, *Internat. J. Game Theory*, 8 (1979) 109–113, our proof does not refer to the infinite-duration versions. Second, we show that memoryless determinacy straightforwardly generalizes to infinite duration versions of parity and mean payoff games.

1. Introduction

Parity games are infinite duration games played by two adversaries on finite leafless graphs with vertices colored by nonnegative integers. One of the players tries to ensure that the maximal vertex color occurring in the play infinitely often is *even*, while the other wants to make it *odd*. The problem of deciding the winner in parity games is polynomial time equivalent to the Rabin chain tree automata (or parity tree automata) nonemptyness, and to the model checking problem for the μ-calculus [6], one of the most expressive temporal logics of programs, expressively subsuming virtually

☆ Supported by Swedish Research Council Grants "*Infinite* Games: *Algorithms* and Complexity" and "*Interior-Point* Methods for Infinite Games".
* Corresponding author. Tel.: +46-18-471-10-55; fax: +46-18-55-02-25.
E-mail addresses: henrikbj@it.uu.se (H. Björklund), svens@it.uu.se (S. Sandberg), vorobyov@csd.uu.se (S. Vorobyov).

0304-3975/$ - see front matter
doi:10.1016/S0304-3975(03)00427-4

all known such logics. For these reasons, parity games are of considerable importance and have been extensively studied by the complexity-theoretic, automata-theoretic, and verification communities.

One of the fundamental properties of parity games, which almost all decision algorithms rely upon, is the so-called *memoryless determinacy*. Vertices of every game can be partitioned into winning sets of both players, who possess positional winning strategies from all vertices in their winning sets. This means that for each vertex owned by a player, the player can decide in advance what to do if the play reaches that vertex, by deterministically selecting one of the outgoing edges independently of the history of the play. Moreover, revealing this positional strategy in advance is not a disadvantage.

Emerson [4] sketched the first memoryless determinacy proof for parity games[1] as early as in 1985. His proof is based on a (fairly complicated) simplification by Hossley and Rackoff [8] (relying on König's lemma) of Rabin's original proof [12] of the nonemptyness problem for Rabin automata. A later, more self-contained, determinacy proof by Emerson and Jutla [5] relies heavily on the μ-calculus, and is *non-constructive*. For example, the definition of a strategy in [5] uses properties of all paths in a binary tree, a set of continuum cardinality. Later McNaughton [10] and Zielonka [15] gave alternative proofs discussed below.

Today it is interesting to note that memoryless determinacy of parity games is a one-line consequence (using a simple reduction; see, e.g., Puri [11]) of the earlier more general result of Ehrenfeucht and Mycielski [2,3] on memoryless determinacy of the so-called *mean payoff games*.

Mean payoff games are also infinite duration games played by two adversaries on finite graphs, but with weighted edges. Players try to maximize/minimize the limit mean value of edge weights encountered during the play. It was proved by Ehrenfeucht and Mycielski [2,3] that every mean payoff game has a unique value v such that Player 0 can ensure a gain of at least v and Player 1 can ensure a loss of at most v, i.e., the games are *determined*. Furthermore, each player can secure this value by using a *positional* (*memoryless*) strategy.

The proof for mean payoff games given by Ehrenfeucht and Mycielski [3] relies upon a sophisticated *cyclic* interplay between *infinite* duration games and their *finite* counterparts. Proofs for infinite games rely on properties of finite games and vice versa. The authors asked whether it is possible to give a direct, rather than roundabout proof, a question we succeeded to answer affirmatively in this paper. Memoryless determinacy for mean payoff games was later shown constructively by Gurvich et al. [7], but their proof is rather involved, using estimates of norms of solutions to systems of linear equations, convergence to the limit, and precision arguments.

The purpose of this paper is to give a simple and direct proof that works uniformly for both parity and mean payoff games. Our proof does not involve any auxiliary constructions or machinery (like μ-calculus, or linear algebra, or limits), proceeds by elementary induction, and constructs positional strategies for more complicated games from strategies for simpler ones. Similar to [3], we rely on *finite duration* versions

[1] Actually for Rabin pairs automata, but easily transfers to parity games/automata.

of the games, played until the first vertex repetition. However, in contrast to [3], we completely avoid roundabout proofs of the properties of finite games using infinite ones and vice versa. Our proof is constructive, although the algorithm it provides is not intended to be very efficient. Due to the importance of parity games and mean payoff games, we feel that a straightforward and constructive proof of memoryless determinacy, common to both games, and without involving external powerful methods should be interesting and useful.

Two interesting memoryless determinacy proofs were given by McNaughton [10] and Zielonka [15], and both have very nice features. McNaughton studies a broad abstract class of games on *finite graphs* (including parity games), for which the winning conditions can be stated in terms of winning subsets of vertices. His proof provides a necessary and sufficient condition for such games to possess memoryless determinacy. This proof is constructive, but does not apply directly to mean payoff games, since the set of vertices visited infinitely often in a play does not uniquely determine its value. Zielonka [15] gives two simple and elegant proofs that work for parity games with a possibly *infinite* number of vertices, but not for mean payoff games. The first version of Zielonka's proof is constructive and uses induction on the number of colors and vertices (transfinite induction if there are infinitely many vertices). The second version is shorter but nonconstructive.

In contrast, our argument exploits structural similarities of parity and mean payoff games to give a uniform proof for both. It would be interesting to know whether our proof technique can be extended to more general classes of *discounted* and *simple stochastic games* [16]. Our current assumption on a winning condition, which allows to reduce infinite to finite duration games and conduct the uniform proof, is too strong, and should be relaxed to cover discounted and simple stochastic games.

The interest in parity games and mean payoff games is to a large extent motivated by their complexity-theoretic importance. For both games, the corresponding decision problems belong to the complexity class $\mathrm{NP} \cap \mathrm{coNP}$, but their PTIME membership status remains open. Much of the research in the complexity of solving these games relies on memoryless determinacy, e.g., [1,7,9,13,16]. Some papers rely essentially on memoryless determinacy, as [1,16], while others prove it independently, explicitly or implicitly [7].

1.1. Outline of the paper

Section 2 gives basic definitions concerning parity games. Theorem 3.1 of Section 3 shows determinacy of finite duration parity games, with strategies requiring memory. The proof is by elementary induction on the number of vertices in a finite tree. Section 4 provides intermediate results about the properties of positional strategies in finite duration games, needed in later sections. In Section 5 we prove by induction on the number of edges in the games, that finite duration parity games are solvable in positional strategies. This is the main theorem in the paper. We extend the proof to infinite duration parity games in Section 6. In Section 7 we show how the proof for parity games generalizes to yield memoryless determinacy of mean payoff games.

2. Parity games

We assume the standard definition of *parity games* (PGs). These are *infinite* duration adversary full information games played by Players 0 and 1 on *finite* directed leafless graphs $G=(V,E)$ with vertices colored by natural numbers. The vertices of the graph are partitioned into sets V_0 and V_1, and every vertex has at least one outgoing edge (i.e., there are no leaves or sinks). The game starts in some vertex, and Player i chooses a successor when a play comes to a vertex in V_i. In this way, the players construct an infinite sequence of vertices, called a *play*. The parity of the *largest* color occurring infinitely often in this play determines the winner: even/odd is winning for Player 0/Player 1, respectively.

A (general) *strategy* of Player i is a function that for every finite prefix of a play, ending in a vertex $v\in V_i$, selects a successor of v (move of the player). A *positional strategy* for Player i is a mapping selecting a unique successor of every vertex $v\in V_i$. When a play comes to $v\in V_i$, Player i unconditionally selects the unique successor determined by the positional strategy, independently of the history of the play. Thus positional strategies are *memoryless*.

When considering a positional strategy σ of Player i, it is often useful to restrict to the graph G_σ obtained from G by removing all outgoing edges from vertices in V_i, except those used by σ.

3. Finite duration parity games

Together with the infinite duration parity games defined above, we also consider their *finite duration* counterparts, fundamental to our proofs. Such games are played on the same graphs as PGs, but only until the first loop is constructed. We first establish memoryless determinacy for finite duration games and then extend it to infinite duration ones.

A *finite duration parity game* (FPG) G_a starts in vertex a, and players alternate moves until some vertex v_l is visited for the second time. At this point the loop from v_l to v_l constructed during the play is analyzed, and the maximal vertex color c occurring on this loop determines the winner. If c is even, then Player 0 wins; otherwise, Player 1 wins.

FPGs are finite perfect information zero-sum games (the loser pays \$1 to the winner). By the general theorem of game theory (proved nonconstructively using Brouwer's or Kakutani's fixpoint theorems), every such game has a value. However, for the special case of FPGs, the proof of this fact is very simple and constructive. The argument is quite well known and can be attributed to Zermelo [14]. Nevertheless, to make the paper self-contained and to stress that probabilistic strategies[2] are not needed to decide a winner in a finite duration parity game, we give a direct proof here.

[2] Necessary, for example, in the "stone-paper-scissors" game.

Proposition 3.1. *The vertices of any FPG G can be partitioned into sets $W_0(G)$ and $W_1(G)$ such that Player i can win G_v when $v \in W_i(G)$.*

Definition 3.2. The set $W_i(G)$ in Proposition 3.1 is called the *winning set* for Player i.

Proof. Starting from every vertex v construct an AND-OR tree of all possible developments of an FPG starting at v. This tree is finite, with leaves corresponding to the first vertex repetitions on paths from the root. Mark those leaves with 0 or 1 corresponding to which player wins in the leaf (on the corresponding loop). Evaluate the root of the tree bottom-up by repeatedly using the rules: (1) if a vertex u of Player i has a successor marked i, then mark u with i; (2) if all successors of a vertex u of Player i are marked $1-i$, then mark u with $1-i$. This evaluation uniquely determines the mark of the root, 0 or 1. □

Note that although implementing a winning strategy according to the construction in the above proof does not need randomness, it requires memory to keep the history of the play in order to decide where to move at each step. In the following sections, based on Proposition 3.1, we show how to construct *positional* strategies.

4. Positional strategies in finite duration (parity) games

Finite duration parity games are slightly less intuitive and require different arguments compared to their infinite counterparts. For example, any finite prefix of an infinite play does not matter when determining the winner in an infinite game. In contrast, this prefix is essential in deciding the winner in a finite duration game, because its every vertex is a potential termination point of the game. Some other common infinite-case intuitions fail for finite games, and need some extra care.

This section confirms two basic intuitions about positional strategies in FPGs. The first one is an apparently obvious fact that you can win from a vertex provided you win from some of its successors. It is used to prove the second, demonstrating that when a player uses an optimal positional strategy, the play never leaves her winning set. In infinite games, both lemmas are one-line consequences of memoryless determinacy, but we stress that we need to prove these properties for finite games and without relying on determinacy.

Lemma 4.1. *In any FPG, if Player i has a positional strategy winning from some successor u of $v \in V_i$, then Player i has a positional strategy winning from v.*

(The straightforward construction "play from v to u and then follow the positional winning strategy from u" is not completely obvious. Assume a positional winning strategy from u uses in v an edge different from (v,u). The previous construction simply does not yield a *positional* strategy.)

Proof. Let us briefly discuss the recurring idea of the arguments we rely upon. Suppose Player i fixes a positional strategy σ, and consider the graph G_σ, where all other choices for Player i are removed. Then Player i wins from a vertex x iff Player $1-i$ cannot force a play in G_σ from x into a simple loop losing for Player i.

Let σ be a positional strategy winning for Player i from some successor u of $v \in V_i$. If σ is winning also from v (which can be checked by inspecting the loops reachable from v in G_σ as explained above), the claim follows. Otherwise, there is a loop λ, losing for Player i, reachable by a path π from v in G_σ. We claim that Player $1-i$ cannot force a play in G_σ from u to any vertex on π and λ, including v. Indeed, the opposite would imply that Player i loses from u. Change σ only at v, obtaining σ' with $\sigma'(v)=u$. Player i still wins from u with σ', since the set of plays in G'_σ from u remains the same as in G_σ, and exactly the same loops can be forced from v by Player $1-i$. □

Important observation: We pause here to make an important observation. The argument in the proof of Lemma 4.1 actually applies to the whole class of finite duration games that, like FPGs: (1) stop as soon as some vertex is first revisited, (2) for which the winner is determined by the sequence of vertices on the resulting simple loop, and (3) independently of the initial vertex the loop is traversed from. Condition 3 is not satisfied, e.g., for finite versions of discounted mean payoff games or simple stochastic games [16].

This class includes, in particular, the finite decision version of mean payoff games, considered in Section 7. We encourage the reader to verify that all subsequent proofs in this section and Section 5 apply for this general class of games. Actually, our proof of memoryless determinacy of finite mean payoff games in Section 7.2 relies on this observation and thus recycles the work done for parity games.

The next lemma puts the (yet unproved) assumption "if both players win by positional strategies from every vertex in their respective winning[3] sets" in the premise. Under this assumption it shows that no play consistent with such a winning positional strategy leaves the player's winning set.

Lemma 4.2. *In any FPG, suppose σ_0 and σ_1 are positional strategies for Player* 0 *and* 1 *such that Player i by using σ_i wins every game that starts in $W_i(G)$, for $i \in \{0,1\}$. Then all plays starting in $W_i(G)$ and played according to σ_i stay in $W_i(G)$.*

Proof. Consider a game on the graph G_{σ_i} starting in some vertex from $W_i(G)$. In this game, we can assume that all vertices belong to Player $1-i$ because there are no choices in Player i's vertices. By Lemma 4.1, any edge (u,v) with $v \in W_{1-i}(G)$ must also have $u \in W_{1-i}(G)$, so there are no edges from $W_i(G)$ to $W_{1-i}(G)$ in G_{σ_i}. □

This lemma will be extensively used in the inductive proof of the main theorem in the next section, where its premise will be guaranteed to hold by inductive assumption.

[3] Winning by some, maybe nonpositional strategy.

5. Memoryless determinacy of finite duration parity games

In this section we prove the main theorem of the paper, which implies memoryless determinacy of the infinite duration parity games (Section 6). The proof itself does not use any reference to the infinite games. In Section 7 we extend this theorem to finite and infinite duration mean payoff games. Actually, the proof of Theorem 5.1 is not modified, we just show how to adjust winning conditions so as to reuse the theorem and its proof as they are.

Theorem 5.1. *In every FPG G, there are positional strategies σ_0 of Player 0 and σ_1 of Player 1 such that Player i wins by using σ_i whenever a play starts in $W_i(G)$, no matter which strategy Player $1-i$ applies.*

Proof. The proof is by induction on the number $|E|-|V|$ of choices of both players. We stress once again that the proof applies to a more general class of games, as observed in Section 4, and therefore will be reused to demonstrate memoryless determinacy of the decision version of finite mean payoff games in Section 7.2.

The *base case* is immediate: if $|E|=|V|$ then there are no vertices with a choice, and all strategies are positional.

For the *inductive step* we split into three cases, depending on whether there are vertices x from which the player who owns x can win. Let $V_i^* \subseteq V_i$ (for $i=0,1$) be the sets of vertices of Player i with outdegree more than 1. Assume the theorem holds whenever $|E|-|V|<n$, and consider $|E|-|V|=n$.

Case 1: $V_0^* \cap W_0(G) \neq \emptyset$. Take $x \in V_0^* \cap W_0(G)$ and let e be an edge leaving x such that Player 0 can win G_x after selecting e in the first move. (The rest of the Player 0 strategy does not need to be positional.) Consider the game G' in which we remove all edges leaving x, except e. By inductive assumption, there are positional strategies σ_i such that Player i applying σ_i wins any play in G' that starts in $W_i(G')$. These strategies are also positional strategies in G. We will prove that σ_i is winning for Player i in G if the play starts in a vertex from $W_i(G')$.

Suppose $v \in W_0(G')$ and Player 0 uses σ_0 in G_v. Any play in G_v following σ_0 was also possible in G'_v. Since all such plays are winning for Player 0 in G'_v, all of them are also winning for Player 0 in G_v. Therefore, $W_0(G') \subseteq W_0(G)$.

Now assume $v \in W_1(G')$ and Player 1 uses σ_1 in G_v. Notice that $x \in W_0(G')$, since there is a winning strategy that uses e and the game terminates as soon as x is revisited. By Lemma 4.2, whenever Player 1 uses σ_1, no play in G'_v, hence in G_v, can ever reach x, because G'_v and G_v differ only in edges *leaving* x. But any play in G_v according to σ_1 not reaching x was possible also in G'_v, so Player 1 wins any such play in G_v. Hence, $W_1(G') \subseteq W_1(G)$.

Since $W_i(G') \subseteq W_i(G)$, and $W_0(G')$, $W_1(G')$ form a partition, we thus showed that $W_i(G) = W_i(G')$ and Player i wins G_v using σ_i, for any $v \in W_i(G)$.

Case 2: $V_1^* \cap W_1(G) \neq \emptyset$ *and Case* 1 *does not apply.* The proof is symmetrical to Case 1.

Case 3: $V_0^* \cap W_0(G) = V_1^* \cap W_1(G) = \emptyset$. Since only Player i may have choices in $W_{1-i}(G)$, we will assume that all vertices in $W_i(G)$ belong to V_{1-i}, that is, $W_i(G) = V_{1-i}$

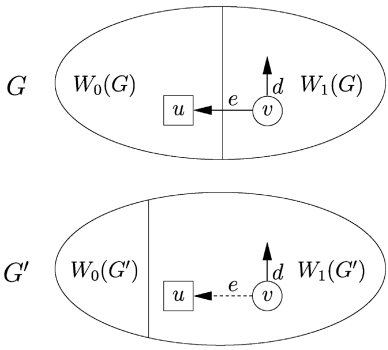

Fig. 1. If we remove e from G, then u changes from losing to winning for Player 0.

for $i \in \{0,1\}$. We may also assume that one of the players has choices; otherwise the base case applies.

We first prove that there are no edges between the winning sets. Suppose, towards a contradiction, that there is an edge e from $v \in W_1(G)$ to $u \in W_0(G)$. Fig. 1 depicts this situation, where round and square vertices belong to Player 0 and 1, respectively (the case of an edge back is symmetric).

Although u is losing for Player 1, there must be a way for Player 1 to win a play that starts in v even if Player 0 moves by e to u. This can *only* be done if Player 1 can force the play back to v, because any other winning play for Player 1 from v via u would be a winning play from u as well. However, Player 1 cannot win if a play starts in u. Therefore, if the game starts in u, and is forced by Player 1 to v, Player 0 must be able to do something else in order to win, rather than selecting e. Thus there must be another edge d leaving v.

Now create the game G' by removing the edge e. Since Player 0 has less choices, we must have $W_0(G') \subseteq W_0(G)$. By inductive assumption, there are positional strategies σ_0 and σ_1, winning from $W_0(G')$ and $W_1(G')$, respectively. Since all vertices in $W_0(G')$ can be assumed to belong to V_1, Lemma 4.2 implies that there are no edges leaving $W_0(G')$. This in turn means that u belongs to $W_1(G')$. Otherwise, Player 0 could have won from v in G by following e, because the play would have never left $W_0(G')$, and any loop formed would have been winning for Player 0.

When e was removed, vertex u turned from losing to winning for Player 1. This implies that in G', Player 1 must force the play from u to v in order to win, since all other plays would have been possible in G as well, and all were losing for Player 1. In G, however, no matter how Player 1 forced the play from u to v, Player 0 could win by using d and some strategy for the remaining play. All these plays are possible in G' as well, so Player 1 cannot win from u in G'. This contradiction shows that there can be no edge e from $W_1(G)$ to $W_0(G)$. By symmetric reasoning, there are no edges from $W_0(G)$ to $W_1(G)$.

Now, since there are no edges between the winning sets, and both players lack choices within their own winning sets, any positional strategy is optimal, i.e., winning from all vertices in the player's winning set. □

6. Extension to infinite duration parity games

We now show that a positional strategy that wins an FPG starting in vertex v also wins, for the same player, the infinite duration parity game starting in v on the same game graph.

Let σ be a winning strategy (positional or not) of Player i (for $i \in \{0,1\}$) in the FPG starting from v. No matter what the opponent does, the first revisit to a previously visited vertex v_l guarantees that the maximal color on the loop from v_l to v_l is winning for Player i. Now forget what happened on the path from v_l to v_l, assume v_l is visited for the first time, and let the FPG develop further. The next time some vertex v'_l (not necessarily equal to v_l) is revisited, we also know that the maximal color on the loop from v'_l to v'_l is winning for Player i. Again, forget what happened on the path from v'_l to v'_l, assume v'_l is visited for the first time, and let the FPG develop further. In this way, using the winning strategy for Player i, we construct an infinite sequence of loops and the corresponding infinite sequence of maximal colors $S = \{c_i\}_{i=1}^{\infty}$ winning for Player i. It follows that the maximal color hit in the infinite game infinitely often is the maximum appearing in S infinitely often. This also means that the winning sets of the players in the infinite and finite games coincide.

The following theorem makes the above argument formal. It also establishes the converse, that positional winning strategies in infinite parity games are winning in finite ones.

Theorem 6.1. *A positional strategy σ of Player i wins in an infinite duration PG G_v if and only if it wins in the corresponding FPG.*

Proof. We will show that σ wins the infinite game if and only if the highest color on every simple loop reachable from v in G_σ is winning for Player i. The latter is equivalent to σ winning the finite game.

If G_σ contains a loop reachable from v with a highest color losing for Player i, then σ is losing: Player $1-i$ can go to this loop and stay in it forever.

Conversely, suppose G_σ does not contain any loops with losing highest color reachable from v. We will prove that there is no infinite path starting from v in G_σ on which the highest color appearing infinitely often is losing for Player i. Assume, towards a contradiction, that there is such a path p on which the highest color occurring infinitely often is c. There must be some vertex u of color c that appears infinitely often on p. But between any two such appearances of u on the path, all other vertices on some simple loop containing u must appear. Among these vertices, at least one should have a winning color higher than c, by assumption. This means that some winning color higher than c appears infinitely often on p, contradicting the assumption. □

As a direct consequence, the memoryless determinacy of Theorem 5.1 holds also for infinite duration parity games.

Corollary 6.2. *In every PG G, there are positional strategies σ_0 of Player 0 and σ_1 of Player 1 such that Player i wins by using σ_i whenever a play starts in $W_i(G)$, no matter which strategy Player $1-i$ applies.*

7. Extension to mean payoff games

This section shows how the memoryless determinacy proof for parity games extends to mean payoff games.

7.1. Finite and infinite duration mean payoff games

Mean payoff games [3,7,16] are similar to parity games. Let $V = V_0 \cup V_1$, $V_0 \cap V_1 = \emptyset$, $E \subseteq V \times V$ (where, for each $u \in V$, there is some v with $(u,v) \in E$), and $c : E \to \mathbb{R}$. Define the game graph $\Gamma = \Gamma(V_0, V_1, E, c)$. Starting in some predefined vertex v_0, the two players move by selecting edges from their respective vertex sets in the same way as in parity games. This yields an infinite play (sequence of vertices) $p = v_0 v_1 v_2 \ldots$ Player 0 wants to maximize

$$v_0(p) = \liminf_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} c(v_i, v_{i+1})$$

and Player 1 wants to minimize

$$v_1(p) = \limsup_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} c(v_i, v_{i+1}).$$

Analogously to the finite version of parity games, we define *finite duration mean payoff games* (FMPGs). Like in FPGs, a play starts in the initial vertex v_0 and ends as soon as a loop is formed. The value $v(p)$ of the play $p = v_0 v_1 \cdots v_m \cdots v_n$, where $v_m = v_n$, is the mean value of the edges on this loop:

$$v(p) = \frac{1}{n-m} \sum_{i=m}^{n-1} c(v_i, v_{i+1}).$$

Player 0 wants to maximize this value and Player 1 wants to minimize it.

FMPGs are, like FPGs, finite, zero-sum, perfect information games, and are therefore determined. Also like FPGs, FMPGs are a special case of such games, for which this fact can be proved in an elementary way.

Proposition 7.1. *For every FMPG starting in any vertex u, there is a* value $v(u)$ *such that Player* 0 *can ensure a gain of at least $v(u)$, and Player* 1 *can ensure a loss of at most $v(u)$, independently of the opponent's strategy.*

Proof. Similar to the proof of Proposition 3.1, consider the finite tree of all possible plays from u. Assign to each leaf the value of the play it represents. Let all internal vertices corresponding to choices of Player 0 be `MAX`-vertices, and let the choices of Player 1 be `MIN`-vertices. The root of the resulting `MIN-MAX`-tree can be straightforwardly evaluated in a bottom-up fashion, and its value is the value $v(u)$ of the game starting from u. □

The following lemma, proved in [3], using only elementary means, shows that a strategy in an FMPG G yields a strategy in the corresponding MPG Γ, guaranteeing

the *same* value. Thus it is enough to show that there are optimal positional strategies in FMPGs.

Lemma 7.2 (Ehrenfeucht-Mycielski [3]). *A strategy σ of Player i in an FMPG G can be modified to a strategy $\tilde{\sigma}$ in the corresponding MPG Γ such that the following properties hold.*
(1) *If σ secures the value v in G, then $\tilde{\sigma}$ secures v in Γ.*
(2) *If σ is positional, then $\tilde{\sigma}=\sigma$ so $\tilde{\sigma}$ is also positional.* □

7.2. Main theorem for the decision version of finite mean payoff games

Consider the *decision* version of finite duration mean payoff games, denoted FMPG(D), in which we are only interested in whether the value of a play is greater than some threshold t, not in the actual value. The winning condition for Player 0 is either $v(p)>t$ or $v(p)\geqslant t$. This allows us to define *winning sets* similarly to the case of finite duration parity games. Say that $u\in W_0(G)$ if Player 0 has a strategy that ensures a value $v(u)\geqslant t$, and $u\in W_1(G)$ otherwise.

As was observed in Sections 4 and 5, the proofs of Lemmas 4.1, 4.2, and of Theorem 5.1 work without any modifications for a broader class of games satisfying the following.

Assumptions. (1) A play on a game graph starts in a vertex and stops as soon as a loop is formed, and
(2) The winner is determined by the sequence of vertices on the loop, modulo cyclic permutations.

These assumptions on the winning condition are sufficient to prove Theorem 5.1, and Lemmas 4.1 and 4.2 upon which it depends, as can be readily verified by inspecting their proofs. It clearly holds for FMPG(D)s, where a sequence of vertices *uniquely* determines a sequence of edges, because there are no multiple edges, and their average cost determines the winner. Therefore, we obtain the following memoryless determinacy result for the decision version of finite mean payoff games.

Theorem 7.3. *In every* FMPG(D), *each player has a positional strategy that wins from all vertices in the player's winning set.* □

By Lemma 7.2, this result extends to the infinite duration mean payoff games.

7.3. Ergodic partition theorem for mean payoff games

In this section we reinforce Theorem 7.3 by proving that each vertex v in an FMPG has a value, which both players can secure by means of positional strategies whenever

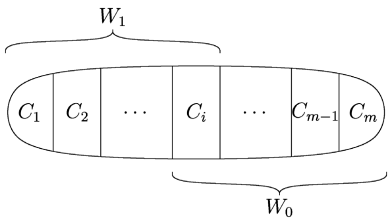

Fig. 2. The ergodic partitions of G.

a play starts in v. Moreover, the same pair of strategies can be used independently of the starting vertex. More formally:

Theorem 7.4 (Memoryless determinacy and ergodic partition). *Let G be an FMPG and $\{C_i\}_{i=1}^m$ be a partition* (*called* ergodic) *of its vertices into classes with the same value x_i, as given by Proposition* 7.1. *There are positional strategies σ_0 and σ_1 for Player* 0 *and* 1 *with the following properties*:

if the game starts from a vertex in C_i, then σ_0 secures a gain $\geqslant x_i$ for Player 0, *and σ_1 secures a loss $\leqslant x_i$ for Player* 1.

Moreover, Player 0 *has no vertices with outgoing edges leading from C_i to C_j with $x_i < x_j$, and Player* 1 *has no vertices with outgoing edges leading from C_i to C_j with $x_i > x_j$.*

Proof. By Proposition 7.1, there exist values $x_1 < x_2 < \cdots < x_m$ and a partition C_1, $C_2, \ldots, C_m$ such that for every starting vertex $u \in C_i$ of an FMPG G both players ensure for themselves (possibly by nonpositional strategies) the value x_i. Now, for every value x_i solve two FMPG(D) problems (using Theorem 7.3), as shown in Fig. 2:

(1) find the winning set W_0 and corresponding strategy σ_0 of Player 0 securing a gain $\geqslant x_i$ when a play starts in W_0.
(2) find the winning set W_1 and corresponding strategy σ_1 of Player 1 securing a loss $\leqslant x_i$ when a play starts in W_1.

Consider $W_0 \cap W_1$. In this (nonempty) set both Player 0 can secure a gain $\geqslant x_i$ by means of σ_0, and Player 1 can secure a loss $\leqslant x_i$ by means of σ_1. In other words, $W_0 \cap W_1 = C_i$.

By Lemma 4.2, any play starting from W_0 always stays in this set, when Player 0 uses σ_0 (Player 1 has no edges out of it), and symmetrically for W_1.

We repeat the argument above for all values $x_1, \ldots, x_m$ getting winning positional strategies σ_0^i, σ_1^i. Using the property from the preceding paragraph we merge all these strategies into positional strategies σ_0 and σ_1 for Player 0 and Player 1 as stated in the theorem. Simply define σ_0 as coinciding with σ_0^i on the set of vertices $V_0 \cap C_i$, and similarly for σ_1. □

By Lemma 7.2, the same results on memoryless determinacy and ergodic partition from Theorem 7.4, hold also for (infinite duration) mean payoff games. Moreover, it is

easy to see that these results hold (with the same proof) in the version of parity games where Player 0 and 1 not only want to win with *some* even/odd color, but want to win with *highest possible even/odd* color. This version of parity games is especially suitable for solving by means of a randomized subexponential algorithm described in [1], as well as by an iterative strategy improvement algorithm from [13].

8. Conclusions

We have presented a new unified proof of the well-known memoryless determinacy results for parity and mean payoff games on finite graphs. There are several previous proofs, but we nonetheless think our proof is useful since it combines several nice properties. It is simple and constructive, providing an easy introduction to the field. Relying only on elementary methods, it illustrates that proving the basic properties of infinite games does not need to attract external notions of limits and approximation. The distinctive feature of our proof is that we first establish memoryless determinacy for the finite duration versions of the games and then extend it to infinite duration. As a consequence, we avoid cyclic, roundabout proofs, as in Ehrenfeucht–Mycielski [3], thus answering positively their question whether one could avoid circularity proving determinacy of infinite and finite duration mean payoff games. Our proof indicates that in this respect finite duration parity and mean payoff games are "more fundamental", directly implying memoryless determinacy for their infinite duration counterparts. Furthermore, by applying to both parity and mean payoff games, the proof stresses the structural similarities between both games.

Can our proof be extended for more general classes of games including *discounted payoff games* and *simple stochastic games* [16]? Our current assumptions on the winning condition (see Section 7.2) do not apply to these games. Discounted payoff games can be formulated in a finite-duration version, but Lemmas 4.1 and 4.2 do not hold for them. We do not know whether there is a finite-duration version of simple stochastic games, and although some version of the lemmas hold in the infinite version, it is unclear whether this suffices for the proof. Nevertheless, we feel that the structures of these games have relevant features in common with parity and mean payoff games. It would be interesting to know whether memoryless determinacy can be proved uniformly for all four games, with our proof technique or another.

References

[1] H. Björklund, S. Sandberg, S. Vorobyov, A discrete subexponential algorithm for parity games, in: H. Alt, M. Habib (Eds.), 20th Internat. Symp. on Theoretical Aspects of Computer Science, STACS'2003, Lecture Notes in Computer Science, Vol. 2607, Springer, Berlin, 2003, pp. 663–674. Full preliminary version: TR-2002-026, Department of Information Technology, Uppsala University, September 2002.

[2] A. Ehrenfeucht, J. Mycielski, Positional games over a graph, Notices Amer. Math Soc. 20 (1973) A–334.

[3] A. Ehrenfeucht, J. Mycielski, Positional strategies for mean payoff games, Internat. J. Game Theory 8 (1979) 109–113.

[4] E.A. Emerson, Automata, tableaux, and temporal logics, in: R. Parikh (Ed.), Proc. Workshop on Logic of Programs, Lecture Notes in Computer Science, Vol. 193, Springer, Berlin, 1985, pp. 79–87.
[5] E.A. Emerson, C.S. Jutla, Tree automata, μ-calculus and determinacy, in: Ann. IEEE Symp. on Foundations of Computer Science, 1991, pp. 368–377.
[6] E.A. Emerson, C. Jutla, A.P. Sistla, On model-checking for fragments of μ-calculus, in: C. Courcoubetis (Ed.), Computer Aided Verification, Proc. 5th Internat. Conf., Lecture Notes in Computer Science, Vol. 697, Springer, Berlin, 1993, pp. 385–396.
[7] V.A. Gurevich, A.V. Karzanov, L.G. Khachiyan, Cyclic games and an algorithm to find minimax cycle means in directed graphs, USSR Comput. Math. & Math. Phys. 28 (5) (1988) 85–91.
[8] R. Hossley, C. Rackoff, The emptiness problem for automata on infinite trees, in: 13th Symp. on Automata and Switching Theory, 1972, pp. 121–124.
[9] M. Jurdziński, Small progress measures for solving parity games, in: H. Reichel and S. Tison (Eds.), 17th STACS, Lecture Notes in Computer Science, Vol. 1770, Springer, Berlin, 2000, pp. 290–301.
[10] R. McNaughton, Infinite games played on finite graphs, Ann. Pure Appl. Logic 65 (2) (1993) 149–184.
[11] A. Puri, Theory of hybrid systems and discrete events systems, Ph.D. Thesis, EECS University, Berkeley, 1995.
[12] M. Rabin, Decidability of second order theories and automata on infinite trees, Trans. Amer. Math. Soc. 141 (1969) 1–35.
[13] J. Vöge, M. Jurdziński, A discrete strategy improvement algorithm for solving parity games, in: E.A. Emerson and A.P. Sistia (Eds.), CAV'00: Computer-Aided Verification, Lecture Notes in Computer Science, Vol. 1855, Springer, Berlin, 2000, pp. 202–215.
[14] E. Zermelo, Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiele, in: E.W. Hobson, A.E.H. Love (Eds.), Proc. 5th Internat. Congr. Mathematics, Vol. 2, Cambridge, 1913, pp. 501–504.
[15] W. Zielonka, Infinite games on finitely coloured graphs with applications to automata on infinite trees, Theor. Comput. Sci. 200 (1998) 135–183.
[16] U. Zwick, M. Paterson, The complexity of mean payoff games on graphs, Theor. Comput. Sci. 158 (1996) 343–359.

Paper IV

A Combinatorial Strongly Subexponential Strategy Improvement Algorithm for Mean Payoff Games

Henrik Björklund, Sven Sandberg, Sergei Vorobyov *

Uppsala University, Information Technology Department, Box 337, 751 05 Uppsala, Sweden

Abstract

We suggest the first strongly subexponential and purely combinatorial algorithm for solving the mean payoff games problem. It is based on iteratively improving the longest shortest distances to a sink in a possibly cyclic directed graph. We identify a new "controlled" version of the shortest paths problem. By selecting exactly one outgoing edge in each of the controlled vertices we want to make the shortest distances from all vertices to the unique sink as long as possible. Under reasonable assumptions the problem belongs to the complexity class NP∩coNP. Mean payoff games are easily reducible to this problem. We suggest an algorithm for computing longest shortest paths. Player Max selects a strategy (one edge in each controlled vertex) and player Min responds by evaluating shortest paths to the sink in the remaining graph. Then Max locally changes choices in controlled vertices looking at attractive switches that seem to increase shortest paths lengths (under the current evaluation). We show that this is a monotonic strategy improvement, and every locally optimal strategy is globally optimal. This allows us to construct a randomized algorithm of complexity $\min(poly \cdot W,\ 2^{O(\sqrt{n \log n})})$, which is simultaneously pseudopolynomial (W is the maximal absolute edge weight) and subexponential in the number of vertices n. All previous algorithms for mean payoff games were either exponential or pseudopolynomial (which is purely exponential for exponentially large edge weights).

⋆ Supported by Swedish Research Council grants "*Infinite Games: Algorithms and Complexity*", "*Interior-Point Methods for Infinite Games*", and by an Institutional grant from the Swedish Foundation for International Cooperation in Research and Higher Education.

⋆⋆Extended abstract in MFCS'2004, preliminary version DIMACS TR 2004-05

* Corresponding author. Phone: +46 18 471 10 55. Fax: +46 18 55 02 25

Email addresses: henrikbj@it.uu.se (Henrik Björklund), vorobyov@csd.uu.se (Sergei Vorobyov).

1 Introduction

Infinite games on finite graphs play a fundamental role in model checking, automata theory, logic, and complexity theory. We consider the problem of solving *mean payoff games* (MPGs) [9,21,20,10,26], also known as *cyclic games* [14,22]. In these games, two players take turns moving a pebble along edges of a directed edge-weighted graph. Player MAX wants to maximize and player MIN to minimize (in the limit) the average edge weight of the infinite path thus formed. Mean payoff games are determined, and every vertex has a *value*, which each player can secure by a uniform positional strategy. Determining whether the value is above (below) a certain threshold belongs to the complexity class NP∩coNP. The well-known parity games, also in NP∩coNP, polynomial time equivalent to model-checking for the μ-calculus [12,11], are polynomial time reducible to MPGs. Other well-known games with NP∩coNP decision problems, to which MPGs reduce, are *simple stochastic* [6] and *discounted payoff* [23,26] games. At present, despite substantial efforts, there are no known polynomial time algorithms for the games mentioned.

All previous algorithms for mean payoff games are either pseudopolynomial or exponential. These include a potential transformation method by Gurvich, Karzanov, and Khachiyan [14] (see also Pisaruk [22]), and a dynamic programming algorithm solving k-step games for big enough k by Zwick and Paterson [26]. Both algorithms are *pseudopolynomial* of complexity $O(poly(n) \cdot W)$, where n is the number of vertices and W is the maximal absolute edge weight. For both algorithms there are known game instances on which they show a worst-case $\Omega(poly(n) \cdot W)$ behavior, where W may be exponential in n. Reduction to simple stochastic games [26] and application of the algorithm from [17] gives subexponential complexity (in the number of vertices n) only if the game graph has bounded outdegree. The subexponential algorithms we suggested for simple stochastic games of arbitrary outdegree in [3,4] make $2^{O(\sqrt{n \log n})}$ iterations, but when reducing from mean payoff games, the weights may not allow each iteration (requiring solving a linear program) to run in strongly polynomial time, independent of the weights. This drawback is overcome with the new techniques presented here, which avoid the detour over simple stochastic games altogether.

We suggest a *strongly subexponential strategy improvement* algorithm, which starts with some strategy of the maximizing player[1] MAX and iteratively "improves" it with respect to some strategy evaluation function. Iterative strategy improvement algorithms are known for the related simple stochas-

[1] The games are symmetric, and the algorithm can also be applied to optimize for the minimizing player. This is an advantage when the minimizing player has fewer choices.

tic [15,7], discounted payoff [23], and parity games [24,2]. Until the present paper, a direct combinatorial iterative strategy improvement for mean payoff games appeared to be elusive. Reductions to discounted payoff games and simple stochastic games (with known iterative strategy improvement) lead to numerically unstable computations with long rationals and solving linear programs. The algorithms suggested in this paper are free of these drawbacks. Our method is discrete, requires only addition and comparison of integers in the same order of magnitude as occurring in the input. In a combinatorial model of computation, the subexponential running time bound is independent of the edge weights. There is also a simple reduction from parity games to MPGs, and thus our method can be used to solve parity games. Contrasted to the strategy improvement algorithms of [24,2], the new method is conceptually much simpler, more efficient, and easier to implement.

We present a simple and discrete randomized subexponential strategy improvement scheme for MPGs, and show that for any integer p, the set of vertices from which MAX can secure a value $> p$ can be found in time

$$\min(O(n^2 \cdot |E| \cdot W),\ 2^{O(\sqrt{n \log n})}),$$

where n is the number of vertices and W is the largest absolute edge weight. The first bound matches those from [14,26,22], while the second part is an improvement when, roughly, $n \log n < \log^2 W$.

The new strategy evaluation for MPGs may be used in several other iterative improvement algorithms, which are also applicable to parity and simple stochastic games [24,15,7]. These include random single switch, all profitable switches, and random multiple switches; see, e.g., [1]. They are simplex-type algorithms, very efficient in practice, but without currently known subexponential upper bounds, and no nontrivial lower bounds.

Outline. Section 2 defines mean payoff games and introduces the associated computational problems. Section 3 describes the longest-shortest paths problem and its relation to mean payoff games. In addition, it gives an intuitive explanation of our algorithm and the particular randomization scheme that achieves subexponential complexity. Section 4 describes the algorithm in detail and Section 5 proves the two main theorems guaranteeing correctness. In Section 6 we explain how to improve the cost per iteration, while detailed complexity analysis is given in Section 7. Possible variants of the algorithm are discussed in Section 8. Section 9 shows that the longest-shortest path problem is in NP$\cap$coNP. In Section 10 an example graph family is given for which the wrong choice of iterative improvement policy leads to an exponential number of iterations. Finally, Section 11 discusses the application of our algorithm to solving parity games.

2 Preliminaries

2.1 Mean Payoff Games

A *mean payoff game* (MPG) [21,20,10,14,26] is played by two adversaries, MAX and MIN, on a finite, directed, edge-weighted, leafless graph $G = (V = V_{\text{MAX}} \uplus V_{\text{MIN}}, E, w)$, where $w : E \to \mathbb{Z}$ is the weight function. The players move a pebble along edges of the graph, starting in some designated initial vertex. If the current vertex belongs to V_{MAX}, MAX chooses the next move, otherwise MIN does. The duration of the game is infinite. The resulting infinite sequence of edges is called a *play*. The *value* of a play $e_1e_2e_3\ldots$ is defined as $\liminf_{k\to\infty} 1/k \cdot \sum_{i=1}^{k} w(e_i)$. The goal of MAX is to maximize the value of the play, while MIN tries to minimize it. In the decision version, the game also has a threshold value p. We say that MAX *wins* a play if its value is $> p$, while MIN wins otherwise. Until Section 7.2 we assume such thresholds to be integral.

A *positional strategy* for MAX is a function $\sigma : V_{\text{MAX}} \to V$ such that $(v, \sigma(v)) \in E$ for all $v \in V_{\text{MAX}}$. Positional strategies for MIN are defined symmetrically. Every mean payoff game has a value and is *memoryless determined*, which means that for every vertex v there is a value $\nu(v)$ and positional strategies of MAX and MIN that secure them payoffs $\geq \nu(v)$ and $\leq \nu(v)$, respectively, when a play starts in v, against any strategy of the adversary [21,20,10,14,22,5]. Moreover, both players have *uniform* positional strategies securing them optimal payoffs independently of the starting vertex. Accordingly, throughout the paper we restrict our attention to positional strategies only. Given a positional strategy σ for MAX, define $G_\sigma = (V, E')$, where $E' = E \setminus \{(v, u) | v \in V_{\text{MAX}} \text{ and } \sigma(v) \neq u\}$, i.e., G_σ results from G by deleting all edges leaving vertices in V_{MAX} except those selected by σ. Note that if both players use positional strategies, the play will follow a (possibly empty) path to a simple loop, where it will stay forever. The value of the play is the average edge weight on this loop [10,14].

2.2 Algorithmic Problems for MPGs

We will address several computational problems for mean payoff games.

The Decision Problem. Given a distinguished start vertex and a threshold value p, can MAX guarantee value $> p$?

p-Mean Partition. Given p, partition the vertices of an MPG G into subsets $G_{\leq p}$ and $G_{>p}$ such that MAX can guarantee a value $> p$ starting from every vertex in $G_{>p}$, and MIN can secure a value $\leq p$ starting from every vertex in $G_{\leq p}$.

Ergodic Partition. Compute the value of each vertex of the game. This gives a partition of the vertices into subsets with the same value. Such a partition is called *ergodic* [14].

Our basic algorithm solves the *0-mean partition* problem, which subsumes the p-mean partition. Indeed, subtracting p from the weight of every edge makes the mean value of all loops (in particular, of optimal loops) smaller by p, and the problem reduces to 0-mean partitioning. The complexity remains the same for integer thresholds p, and changes slightly for rational ones; see Section 7. Clearly, the p-mean partitioning subsumes the decision problem. Section 7.2 extends the basic algorithm to solve the ergodic partition problem. Another problem to which our algorithm may be extended is finding optimal strategies in MPGs.

Our proofs rely on a finite variant of mean payoff games, where the play stops as soon as some vertex is revisited and the mean value on the resulting cycle determines the payoff. Thus, for a play $e_1e_2 \dots e_re_{r+1} \dots e_s$, where $e_r \dots e_s$ is a loop, the value is $\sum_{i=r}^{s} w(e_i)/(s-r+1)$. Ehrenfeucht and Mycielski [10] proved the following (see also [5]).

Theorem 2.1 *The value of every vertex in the finite-duration version of mean payoff games equals its value in the infinite-duration version.* □

The next corollary will be used implicitly throughout the paper.

Proposition 2.2 *A positional strategy σ of* Max *gives value $> p$ in a vertex, if and only if all loops reachable from it in G_σ have average value $> p$.* □

In particular, Max can ensure value > 0 if and only if all reachable loops are positive. Since the partition threshold 0 has a special role in our exposition, we call the $G_{>0}$ partition the *winning set* of Max.

3 A High-Level Description of the Algorithm

We start by informally describing the essential ingredients of our algorithm.

3.1 The Longest-Shortest Paths Problem

The key step in computing 0-mean partitions can be explained by using a "controlled" version of the well-known *single source (target) shortest paths problem* on directed graphs. Suppose in a given digraph some set of *controlled* vertices

is distinguished, and we can select *exactly one edge* leaving every controlled vertex, deleting all other edges from these vertices. Such a selection is called a *positional strategy*. We want to find a positional strategy that maximizes the shortest paths from all vertices to the distinguished sink (also avoiding negative cycles that make the sink unreachable and the distances $-\infty$). For a strategy σ denote by G_σ the graph obtained from G by deleting all edges from controlled vertices except those in σ. Formally, the problem is specified as follows.

The Longest-Shortest Paths Problem (LSP).
Given:

(1) a directed weighted graph G with unique sink t,
(2) some distinguished *controlled* vertices U of G, with $t \notin U$.

Find:

- a positional strategy σ such that in the graph G_σ the shortest simple path from every vertex to t is as long as possible (over all positional strategies).

If a negative weight loop is reachable from a vertex, the length of the shortest path is $-\infty$, which Max does not want. If only positive loops are reachable, and t is not, then the shortest path distance is $+\infty$. [2]

For our purposes it suffices to consider a version of the LSP problem with the following additional input data.

Additionally Given:

- some strategy σ_0, which guarantees that in the graph G_{σ_0} there are no cycles with nonpositive weights.

This additionally supplied strategy σ_0 guarantees that the longest shortest distance from every vertex to the sink t is not $-\infty$; it is not excluded that σ_0 or the optimal strategy will make some distances equal $+\infty$. We make sure that our algorithm never comes to a strategy that allows for nonpositive cycles. The simplifying additional input strategy is easy to provide in the reduction from MPGs, as we show below.

Note that for DAGs, the longest-shortest path problem can be solved in polynomial time using dynamic programming. Start by topologically sorting the

[2] The case of zero weight loops is inconsequential for the application to mean payoff games, and we only need to consider it when proving that the LSP problem is in NP∩coNP in Section 9.

vertices and proceed backwards from the sink (distance 0), using the known longest-shortest distances for the preceding vertices.

3.2 Relating the 0-Mean Partition and Longest-Shortest Paths Problems

The relation between computing 0-mean partitions and computing longest shortest paths is now easy to describe. To find such a partition in an MPG G, add a *retreat* vertex t to the game graph with a self-loop edge of weight 0, plus a 0-weight *retreat* edge from every vertex of MAX to t. From now on, we assume G has undergone this transformation. Clearly, we have the following property.

Proposition 3.1 *Adding a retreat does not change the 0-mean partition of the game, except that t is added to the $G_{\leq 0}$ part.* □

This is because we do not create any new loops allowing MAX to create positive cycles, or MIN to create new nonpositive cycles. MAX will prefer playing to t only if all other positional strategies lead to negative loops.

The key point is now as follows. Break the self-loop in t and consider the LSP problem for the resulting graph, with t being the unique sink. The set V_{MAX} becomes the controlled vertices, and the initial strategy (the "additionally given" clause in the LSP definition above) selects t in every controlled vertex, guaranteeing that no vertex has distance $-\infty$.[3] We have the following equivalence:

Theorem 3.2 *The partition $G_{>0}$ consists exactly of those vertices for which the longest-shortest path distance to t is $+\infty$.* □

As early as in 1991 Leonid Khachiyan (private communication) considered the following variant of Longest-Shortest Paths.

BLOCKING NONPOSITIVE CYCLES. Given a directed edge-weighted leafless graph G, a vertex v, and a set of controlled vertices, where the controller has to choose exactly one outgoing edge, does he have a selection such that in the resulting graph (obtained after deleting all unselected edges from controlled vertices) there are no nonpositive weight cycles reachable from v? □

As an immediate consequence one has the following

[3] Actually, there may exist negative value loops consisting only of vertices from V_{MIN}. Such loops are easy to identify and eliminate in a preprocessing step, using the Bellman–Ford algorithm. In the sequel we assume that this is already done.

Proposition 3.3 *1. The problems 0-Mean Partition in Mean Payoff Games*[4] *and Blocking Nonpositive Cycles are polynomial time equivalent.*

2. Blocking Nonpositive Cycles is in NP∩coNP.

3. Blocking Nonpositive Cycles is polynomial time reducible to Longest-Shortest Paths. □

Note that in Longest-Shortest Paths, except being interested in the $+\infty$ distances to the sink (which corresponds to positive loops in Blocking Nonpositive Cycles) we are additionally interested in computing *finite* distances. Our algorithm iteratively improves these finite distances (until hopefully improving them to $+\infty$).

To our knowledge, there are no other mentions of the Longest-Shortest Paths problem and its relation to mean payoff games in the literature. [5]

Actually, the evaluation of the shortest paths for a fixed positional strategy gives a useful quality measure on strategies that can be used in other iterative improvement schemes. We discuss some possibilities in Section 8.

3.3 The Algorithm

Our algorithm computes longest-shortest paths in the graph resulting from a mean payoff game (after adding the retreat vertex and edges, as explained above), by making iterative strategy improvements. Once a strategy is fixed, all shortest paths are easily computable, using the Bellman-Ford algorithm. Note that there are negative weight edges, so the Dijkstra algorithm does not apply. An improvement to the straightforward application of the BF-algorithm is described in Section 6. Comparing a current choice made by the strategy with alternative choices, a possible improvement can be decided locally as follows. If changing the choice in a controlled vertex to another successor seems to give a longer distance (seems attractive), we make this change. Such a change is called a *switch*.

We prove two crucial properties (Theorems 5.1 and 5.2, respectively):

(1) every such switch really increases the shortest distances (i.e., attractive is improving or profitable);

[4] Recall that this problem consists in finding the set of the game vertices from which the maximizing player can secure a *positive* mean value.

[5] The authors would appreciate any references.

(2) once none of the alternative possible choices is attractive, all possible positive-weight loops MAX can enforce are found (i.e., stable is optimal). [6]

Although our subexponential algorithm proceeds by making just one attractive switch at a time, other algorithms making many switches simultaneously are also possible and fit into our framework. Such algorithms are discussed in Section 8.

Another interpretation of our algorithm is game-theoretic. MAX makes choices in the controlled vertices, and the choices in all other vertices belong to MIN. For every strategy of MAX, MIN responds with an optimal counterstrategy, computing the shortest paths from every vertex to the sink. After that, the algorithm improves MAX's strategy by making an attractive switch, etc.

3.4 Randomization Scheme

The order in which attractive switches are made is crucial for the subexponential complexity bound; see Section 10 for an example of an exponentially long chain of switches. The space of all positional strategies of MAX can be identified with the Cartesian product of sets of edges leaving the controlled vertices. Fixing any edge in this set and letting others vary determines a *facet* in this space.

Now the algorithm for computing the longest-shortest paths in G looks as follows, starting from some strategy σ assumed to guarantee for shortest distances $> -\infty$ in all vertices.

(1) Randomly and uniformly select some facet F of G not containing σ. Throw this facet away, and recursively find a best strategy σ^* on what remains. This corresponds to deleting an edge not selected by σ and finding the best strategy in the resulting subgame.
(2) If σ^* is optimal in G, stop (this is easily checked by testing whether there is an attractive switch from σ^* to F). The resulting strategy is globally optimal, providing for the longest-shortest distances.
(3) Otherwise, switch to F, set $G = F$, denote the resulting strategy by σ, and repeat from step 1.

This is the well-known randomization scheme for linear programming due to Matoušek, Sharir, and Welzl [18,19]. When applied to the LSP and MPG problems, it gives a subexponential $2^{O(\sqrt{n \log n})}$ expected running time bound

[6] The case when zero-weight loops are interpreted as good (winning) for MAX is considered in Section 9.

[18,4]. An essential condition for the analysis to work is as follows. The strategy evaluation we use satisfies the property that facets are partially ordered by the values of their best strategies. After finding the optimum on one facet, the algorithm will never visit a facet with an optimum that is not strictly better in the partial order. It follows that the so-called *hidden dimension* decreases randomly, and the subexponential analysis of [18,19] applies. Another possibility would be to use the slightly more complicated randomization scheme of Kalai [16], as we did in [2] for parity games, which leads to the same subexponential complexity bound.

4 Retreats, Admissible Strategies, and Strategy Measure

As explained above, we modify an MPG by allowing MAX to "surrender" in every vertex. Add a retreat vertex t of MIN with a self-loop of weight 0 and a retreat edge of weight 0 from every vertex of MAX to t. Clearly, MAX secures a value > 0 from a vertex in the original MPG iff the same strategy does it in the modified game. Assume from now on that the retreat has been added to G. Intuitively, the "add retreats" transformation is useful because MAX can start by a strategy that chooses the retreat edge in every vertex, thus "losing only 0". We call strategies "losing at most 0" *admissible*.

Definition 4.1 *A strategy σ of* MAX *in G is* admissible *if all loops in G_σ are positive, except the loop over the retreat vertex t.* □

Our algorithm iterates only through admissible strategies of MAX. This guarantees that the only losing (for MAX) loop in G_σ is the one over t.

4.1 Measuring the Quality of Strategies

We now define a measure that evaluates the "quality" of an admissible strategy. It can be computed in strongly polynomial time, as shown in Section 6.

Given an admissible strategy σ, the best MIN can hope to do is to reach the 0-mean self-loop over t. Any other reachable loop will be positive, by the definition of an admissible strategy. The shortest path from every vertex v to t is well-defined, because there are no nonpositive cycles in G_σ (except over t). Therefore, we define the value of a strategy in a vertex as follows.

Definition 4.2 *For an admissible strategy σ of* MAX*, the value* $\mathrm{val}_\sigma(v)$ *of vertex v is defined as the shortest path distance from v to t in G_σ, or $+\infty$ if t is not reachable.* □

It follows that for admissible strategies finite values may only result from shortest paths leading to the sink (retreat) t.

Note that there is a positional counterstrategy of MIN that guarantees the shortest paths are taken in each vertex, namely the strategy defined by the shortest path forest; see, e.g., [8]. The relative quality of two admissible strategies is defined componentwise.

Definition 4.3 *Let σ and σ' be two admissible strategies. Say that σ is* better than σ'*, formally denoted $\sigma > \sigma'$, if $\mathrm{val}_\sigma(v) \geq \mathrm{val}_{\sigma'}(v)$ for all vertices $v \in V$, with strict inequality for at least one vertex. Say that $\sigma \geq \sigma'$, if $\sigma > \sigma'$ or they have equal values in all vertices.* □

The notation below will be useful for describing switches.

Notation 4.4 If σ is a strategy of MAX, $x \in V_{\text{MAX}}$, and $(x, y) \in E$, then the *switch in x to y* changes σ to the new strategy $\sigma[x \mapsto y]$, defined as

$$\sigma[x \mapsto y](v) \stackrel{\text{def}}{=} \begin{cases} y, & \text{if } v = x; \\ \sigma(v), & \text{otherwise.} \end{cases}$$ □

The following definition makes a distinction between switches that improve the strategy value, and switches that merely look like they do. Later (Corollary 5.5) we will prove that the two notions are equivalent.

Definition 4.5 *Given an admissible strategy σ, a switch $\sigma[v \mapsto u]$ is:*

(1) attractive*, if $w(v, u) + \mathrm{val}_\sigma(u) > \mathrm{val}_\sigma(v)$;*
(2) profitable*, if $\sigma[v \mapsto u]$ is admissible and $\sigma[v \mapsto u] > \sigma$.* □

4.2 Requirements for the Measure

The algorithm relies on the following properties.

(1) If σ is an admissible strategy, and there is no better admissible strategy, then σ is winning from all vertices in MAX's winning set $G_{>0}$. This is evident from the definitions.
(2) Every combination of attractive switches is profitable (Theorem 5.1).
(3) If an admissible strategy has no attractive switches, then there is no better admissible strategy (Theorem 5.2).

Property (2) guarantees monotonicity, termination, and a pseudopolynomial upper bound. Another advantage of (2) is as follows. To find profitable switches,

we only need to test attractivity, which is efficient as soon as the measure has been computed. Testing profitability would otherwise require recomputing the measure for every possible switch.

5 Correctness of the Measure

In this section we state the two major theorems, guaranteeing that every step is improving and that the final strategy is the best, respectively. Afterwards, we give two corollaries that are not strictly necessary for the algorithm to work, but which with little extra effort give an additional insight into the problem.

5.1 Attractiveness Implies Profitability

Our first theorem states that any combination of attractive switches is profitable. This means that we never have to actually evaluate other strategies before selecting the next iterate. Instead we can let the improvement scheme be guided by attractiveness. Monotonicity, guaranteed by Theorem 5.1, implies that every sequence of attractive switches will always terminate. Recall that an admissible strategy does not permit any negative *or* zero value loops.

Theorem 5.1 *If σ is an admissible strategy then any strategy obtained by one or more attractive switches is admissible and better. Formally, if the switches in s_i to t_i are attractive for $1 \leq i \leq r$ and $\sigma' \stackrel{def}{=} \sigma[s_1 \mapsto t_1][s_2 \mapsto t_2] \cdots [s_r \mapsto t_r]$, then σ' is admissible and $\sigma' > \sigma$.*

Proof. It is enough to prove that all loops in $G_{\sigma'}$ are positive and the value does not decrease in any vertex. Then it follows that σ' is admissible and the value in every s_i increases strictly, hence $\sigma' > \sigma$, because

$$\begin{aligned} \mathrm{val}_{\sigma'}(s_i) &= w(s_i, t_i) + \mathrm{val}_{\sigma'}(t_i) && \text{[by definition]} \\ &\geq w(s_i, t_i) + \mathrm{val}_{\sigma}(t_i) && [t_i\text{'s value does not decrease (to be shown)}] \\ &> \mathrm{val}_{\sigma}(s_i). && \text{[the switch in } s_i \text{ to } t_i \text{ is attractive]} \end{aligned}$$

First, we prove that every loop present in $G_{\sigma'}$, but not in G_σ, is positive. Second, we prove that for every path from some vertex v to t present in $G_{\sigma'}$, but not in G_σ, there is a shorter path in G_σ.

1. New loops are positive. Consider an arbitrary loop present in $G_{\sigma'}$, but not in G_σ. Some of the switching vertices s_i must be on the loop; denote them by $\{v_0, \ldots, v_{p-1}\} \subseteq \{s_1, \ldots, s_r\}$, in cyclic order; see Figure 1.

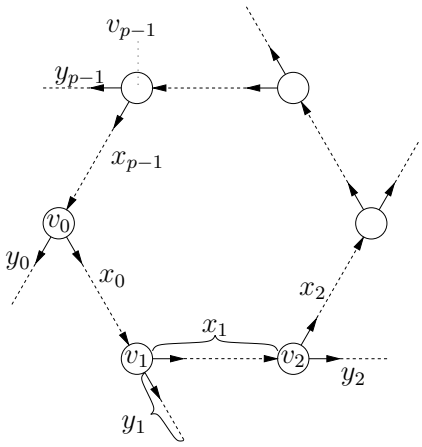

Figure 1. A loop in $G_{\sigma'}$ not in G_σ is depicted. Strategy σ breaks the cycle in vertices $v_0, \ldots, v_{p-1}$, and instead follows the dashed paths to t, each of total edge weight y_i. The weight of the segments between two adjacent v_i's is x_i.

To simplify notation, let $v_p \stackrel{\text{def}}{=} v_0$. Since the switch in v_i $(0 \leq i \leq p-1)$ is attractive, $\mathrm{val}_\sigma(v_i)$ is finite, and there is a path in G_σ from v_i to t. Denote by x_i the sum of weights on the shortest path from v_i to v_{i+1} under σ' and let $y_i = \mathrm{val}_\sigma(v_i)$, i.e., y_i is the sum of weights on the path from v_i to t under σ. Moreover, $x_p \stackrel{\text{def}}{=} x_0$ and $y_p \stackrel{\text{def}}{=} y_0$. Note that

$$y_i = \mathrm{val}_\sigma(v_i) < w(v_i, \sigma'(v_i)) + \mathrm{val}_\sigma(\sigma'(v_i)) \leq x_i + y_{i+1},$$

where the first inequality holds because the switch in v_i to $\sigma'(v_i)$ is attractive and the second because $\mathrm{val}_\sigma(\sigma'(v_i))$ is the length of a shortest path from $\sigma'(v_i)$ to t and $x_i - w(v_i, \sigma'(v_i)) + y_{i+1}$ is the length of another path. Combining these p equalities for every i, we get

$$\begin{aligned} y_0 &< x_0 + y_1 \\ &< x_0 + x_1 + y_2 \\ &< x_0 + x_1 + x_2 + y_3 \\ &\;\;\vdots \\ &< x_0 + x_1 + \cdots + x_{p-1} + y_0. \end{aligned}$$

Therefore, $x_0 + \cdots + x_{p-1} > 0$, hence the loop is positive.

2. New paths to the sink are longer. Consider an arbitrary shortest path from a vertex v to t, present in $G_{\sigma'}$ but not in G_σ. It must contain one or more switching vertices, say $\{v_1, \ldots, v_p\} \subseteq \{s_1, \ldots, s_r\}$, in order indicated; see Figure 2.

Under strategy σ', denote by x_0 the sum of edge weights on the path from v to

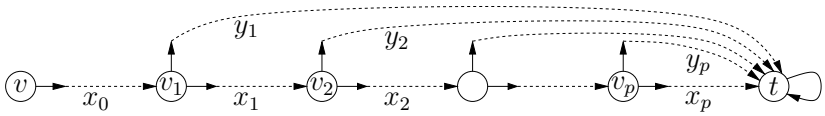

Figure 2. The path going right occurs in $G_{\sigma'}$ but not in G_σ. Strategy σ breaks the path in vertices $v_1, \ldots, v_p$ by going up and following the curved paths to t, each of total edge weight y_i. The edge weight of the segments between two adjacent v_i's is x_i.

v_1, by x_p the sum of weights on the path from v_p to t, and by x_i $(1 \leq i \leq p-1)$ the sum of weights on the path from v_i to v_{i+1}. Let $y_i = \mathrm{val}_\sigma(v_i)$ (attractiveness of switches in v_i implies that y_i are finite and determined by paths to the sink). As above, note that, for $1 \leq i \leq p-1$ one has

$$y_i < w(v_i, \sigma'(v_i)) + \mathrm{val}_\sigma(\sigma'(v_i)) \leq x_i + y_{i+1},$$

for the same reason as in the previous case, and if we let $y_{p+1} = 0$ it holds for $i = p$ as well. Combining these p inequalities we obtain

$$\begin{aligned} x_0 + y_1 &< x_0 + x_1 + y_2 \\ &< x_0 + x_1 + x_2 + y_3 \\ &< x_0 + x_1 + x_2 + x_3 + y_4 \\ &\vdots \\ &< x_0 + \cdots + x_p. \end{aligned}$$

Thus the path from v to t in G_σ taking value $x_0 + y_1$ is shorter than the new path in $G_{\sigma'}$. □

5.2 Stability Implies Optimality

Our second theorem shows that an admissible strategy with no attractive switches is at least as good as any other admissible strategy. This means that if we are only looking for the vertices where Max can enforce positive weight loops (when solving mean payoff games) we can stop as soon as we find an admissible stable strategy. It also follows that if there are no zero-weight loops in G, then any stable admissible strategy is optimal. Section 9 deals with the case where zero-weight loops are considered good for Max.

Theorem 5.2 *If σ is an admissible strategy with no attractive switches, then $\sigma \geq \sigma'$ for all admissible strategies σ'.*

Proof. The proof is in two steps: first we prove the special case when Min does not have any choices, and then we extend the result to general games.

1. One-player games. Assume MIN does not have any choices, i.e., the out-degree of every vertex in V_{MIN} is one. Let σ be an admissible strategy with no attractive switches. We claim that every admissible strategy σ' is no better than σ, i.e., $\sigma' \leq \sigma$.

1a. Finite values cannot become infinite. First, we prove that if $\text{val}_\sigma(v) < \infty$ then $\text{val}_{\sigma'}(v) < \infty$. Consider, toward a contradiction, an arbitrary loop formed in $G_{\sigma'}$, not formed in G_σ (recall that a positive loop is the only way to create an infinite value). There is at least one vertex on this loop where σ and σ' make different choices; assume they differ in the vertices $v_0, \ldots, v_{p-1}$ in cyclic order. Figure 1 shows the situation and again let $v_p \stackrel{\text{def}}{=} v_0$. Denote by x_i the sum of edge weights on the path from v_i to v_{i+1} under strategy σ', and let $y_i = \text{val}_\sigma(v_i)$, i.e., y_i is the sum of edge weights on the path from v_i to t under strategy σ. Let $x_p \stackrel{\text{def}}{=} x_0$ and $y_p \stackrel{\text{def}}{=} y_0$. The condition "no switch is attractive for σ" says exactly that $y_i \geq x_i + y_{i+1}$. Combining these p inequalities, we obtain

$$\begin{aligned}
y_0 &\geq x_0 + y_1 && \text{[by non-attractiveness in } v_0\text{]} \\
&\geq x_0 + x_1 + y_2 && \text{[by non-attractiveness in } v_1\text{]} \\
&\geq x_0 + x_1 + x_2 + y_3 && \text{[by non-attractiveness in } v_2\text{]} \\
&\vdots \\
&\geq x_0 + x_1 + \cdots + x_{p-1} + y_0. && \text{[by non-attractiveness in } v_{p-1}\text{]}
\end{aligned}$$

Thus $x_0 + x_1 + \cdots + x_{p-1} \leq 0$. Since σ' is admissible, there can be no such (nonpositive) loops, a contradiction.

1b. Finite values do not improve finitely. Assume $\text{val}_\sigma(v) < \text{val}_{\sigma'}(v) < \infty$.[7] As in the previous proof, consider the path from v to t under σ'. The strategies must make different choices in one or more vertices on this path, say in $v_1, \ldots, v_p$, in order; Figure 2 applies here as well.

Under strategy σ', denote by x_0 the sum of weights on the path from v to v_1, by x_p the sum of weights on the path from v_p to t, and by x_i $(1 \leq i \leq p-1)$ the sum of weights on the path from v_i to v_{i+1}. Let $y_i = \text{val}_\sigma(v_i)$, i.e., y_i is the sum of weights on the path from v_i to t under σ. The condition "no switch is attractive for σ" says exactly that $y_i \geq x_i + y_{i+1}$, for $1 \leq i \leq p-1$, and

[7] Recall that σ and σ' are admissible, so finite values may only result from paths leading to t.

$y_p \geq x_p$. Combining these p inequalities, we obtain

$$\begin{aligned}
y_1 &\geq x_1 + y_2 && \text{[by non-attractiveness in } v_1\text{]} \\
&\geq x_1 + x_2 + y_3 && \text{[by non-attractiveness in } v_2\text{]} \\
&\geq x_1 + x_2 + x_3 + y_4 && \text{[by non-attractiveness in } v_3\text{]} \\
&\vdots \\
&\geq x_1 + x_2 + \cdots + x_{p-1} + y_p && \text{[by non-attractiveness in } v_{p-1}\text{]} \\
&\geq x_1 + x_2 + \cdots + x_{p-1} + x_p, && \text{[by non-attractiveness in } v_p\text{]}
\end{aligned}$$

and in particular $x_0+y_1 \geq x_0+\cdots+x_p$. But $x_0+y_1 = \mathrm{val}_\sigma(v)$ and $x_0+\cdots+x_p = \mathrm{val}_{\sigma'}(v)$, so indeed we cannot have better finite values under σ' than under σ.

2. Two-player games. Finally, we prove the claim in full generality, where Min may have choices. The simple observation is that Min does not need to use these choices, and the situation reduces to the one-player game we already considered. Specifically, let σ be as before and let τ be an optimal counterstrategy of Min, obtained from the shortest-path forest for vertices with value $< \infty$ and defined arbitrarily in other vertices. Clearly, in the game G_τ, the values of all vertices under σ are the same as in G, and σ does not have any attractive switches. By Case 1, in G_τ we also have $\sigma \geq \sigma'$. Finally, σ' cannot be better in G than in G_τ, since the latter game restricts choices for Min. □

As a consequence, we obtain the following

Corollary 5.3 *In any MPG G every admissible stable strategy is winning for player* Max *from all vertices in $G_{>0}$.* □

The following property is not necessary for correctness, but completes the comparison of attractiveness versus profitability.

Corollary 5.4 *If an admissible strategy σ' is obtained from another admissible strategy σ by one or more* non-attractive *switches, then $\sigma' \leq \sigma$.*

Proof. Consider the game (V, E', w'), where $E' = E \setminus \{(u,v) : u \in V_{\text{Max}} \wedge \sigma(u) \neq v \wedge \sigma'(u) \neq v\}$ and w' is w restricted to E'. In this game, σ has no attractive switches, and both σ and σ' are admissible strategies in the game, hence by Theorem 5.2, $\sigma \geq \sigma'$. □

As a special case (together with Theorem 5.1), we obtain the following equivalence.

Corollary 5.5 *A single switch (between two admissible strategies) is attractive if and only if it is profitable.* □

6 Efficient Computation of the Measure

For an admissible strategy σ the shortest paths in G_σ (strategy measure) can be computed by using the Bellman–Ford algorithm for single-sink shortest paths in graphs with possibly negative edge weights; see, e.g., [8]. This algorithm runs in $O(n \cdot |E|)$ time. Every vertex can have its shortest path length improved at most $O(n \cdot W)$ times (W is the largest absolute edge weight; distances are increased in integral steps). Since there are n vertices, the number of switches cannot exceed $O(n^2 \cdot W)$. Together with the $O(n \cdot |E|)$ bound per iteration this gives total time $O(n^3 \cdot |E| \cdot W)$. Here we show how to reuse the shortest distances computed in previous iteration to improve this upper bound by a linear factor to $O(n^2 \cdot |E| \cdot W)$. Since there are no known nontrivial lower bounds on the number of improvement steps, it is practically important to reduce the cost of each iteration.

We first compute the set of vertices that have different values under the old and the new strategies σ and σ', respectively, and then recompute the values only in these vertices, using the Bellman–Ford algorithm. If the algorithm improves the value of n_i vertices in iteration i, we only need to apply the Bellman–Ford algorithm to a subgraph with $O(n_i)$ vertices and at most $|E|$ edges; hence it runs in time $O(n_i \cdot |E|)$. Since the maximum possible number of integral distance improvements in n vertices is $n^2 \cdot W$, the sum of all n_i's does not exceed $n^2 \cdot W$, so the total running time becomes at most $O(n^2 \cdot |E| \cdot W)$, saving a factor of n. It remains to compute, in each iteration, which vertices need to change their values. Algorithm 1 does this, taking as arguments a game G, the shortest distances $d : V \to \mathbb{N} \cup \{\infty\}$ computed with respect to the old strategy σ, the new strategy σ', and the set of switched vertices $U \subseteq V$, where σ' differs from σ.

Algorithm 1. Mark all vertices v for which $\mathrm{val}_\sigma(v) \neq \mathrm{val}_{\sigma'}(v)$.

MARK-CHANGING-VERTICES($G, U \subseteq V, d : V \to \mathbb{N} \cup \{\infty\}$)

(1) mark all vertices in U
(2) **while** $U \neq \emptyset$
(3) remove some vertex v from U
(4) **foreach** unmarked predecessor u of v in $G_{\sigma'}$
(5) **if** $w(u, x) + d[x] > d[u]$ for all unmarked successors x of u in $G_{\sigma'}$
(6) mark u
(7) $U \leftarrow U \cup \{u\}$

Theorem 6.1 *If an attractive (multiple) switch changes an admissible strategy σ to σ', then for every vertex $v \in V$, the following claims are equivalent.*

(1) Algorithm 1 marks v.
(2) Every shortest path from v to t in G_σ passes through some switch vertex.

(3) $\mathrm{val}_\sigma(v) \neq \mathrm{val}_{\sigma'}(v)$.

Proof. (1) $\implies$ (2). By induction on the set of marked vertices, which expands as the algorithm proceeds. The base case holds since the vertices marked before the while loop are the switch vertices; these clearly satisfy (2). For the induction step, assume the claim holds for every marked vertex and that vertex u is about to be marked on line 6. Let x be any successor of u included in some shortest path from u to t in G_σ. Since $w(u,x) + d[x] = d[u]$, and by the condition on line 5, x must be already marked. Hence, by the induction hypothesis, every shortest path through x passes through U. This completes the induction step.

(2) $\implies$ (1). For an arbitrary vertex v, consider all its shortest paths in G_σ. Denote by $\overline{v}$ the maximal number of edges passed by such a path before a vertex in U is reached (so $\overline{v}$ is the unweighted length of an initial segment). The proof is by induction on $\overline{v}$. The base case is clear: $\overline{v} = 0$ iff $v \in U$, and all vertices in U are marked. Assume that the claim holds for all vertices u with $\overline{u} < k$ and consider an arbitrary vertex v with $\overline{v} = k$. By the inductive hypothesis, all successors of v that occur on a shortest path are marked. Hence, when the algorithm removes the last of them from U, the condition on line 5 is triggered and v is marked.

(3) $\implies$ (2). If some shortest path from v to t in G_σ does not pass through a switch vertex, then the same path is available also in $G_{\sigma'}$, hence $\mathrm{val}_\sigma(v) = \mathrm{val}_{\sigma'}(v)$.

(2) $\implies$ (3). Assume (2) and consider an arbitrary shortest path from v to t in $G_{\sigma'}$. If it contains any switch vertices, let u be the first of them. The same path from v to u, followed by the path in G_σ from u to t, gives a shorter path in G_σ, since the length of shortest paths strictly increase in switch vertices. If the path does not contain any switch vertices, then by (2) it is longer than every shortest path in G_σ. □

We thus showed that Algorithm 1 does what it is supposed to. To finish the argument, we show that it runs in time $O(|E|)$, so it is dominated by the time used by the Bellman–Ford algorithm.

Proposition 6.2 *Algorithm 1 can be implemented to run in time* $O(|E|)$.

Proof. Every vertex can be added to U and analyzed in the body of the **while** loop at most once. The condition on line 5 can be tested in constant time if we keep, for each vertex u, the number of unmarked successors x of u with $w(u,x) + d[x] = d[u]$. Thus, the time taken by the **foreach** loop is linear in the number of predecessors of v (equivalently, in the number of edges entering v), and the claim follows. □

7 Complexity of the Algorithm

Section 2 lists several computational problems for mean payoff games. We first show that our basic 0-mean partition algorithm with small modifications also solves the p-mean partition and the splitting into three sets problems with the same asymptotic running time bound. In Section 7.2, we show how to solve the ergodic partition problem, which introduces a small extra polynomial factor in the complexity.

7.1 Complexity of Partitioning with Integer Thresholds

Our basic algorithm in Section 3 solves the *0-mean partition* problem for MPGs. The *p-mean partition* problem with an integer threshold p can be solved by subtracting p from all edge weights and solving the 0-mean partition problem. As a consequence, we also solve the *decision* problem for integer p. Zwick and Paterson [26] consider a slightly more general problem of *splitting into three sets* around an integer threshold p, with vertices of value $< p$, $= p$, and $> p$, respectively. We can solve this by two passes of the p-mean partition algorithm. First, partition the vertices into two sets with values $\leq p$ and $> p$, respectively. Second, invert the game by interchanging V_{MIN} and V_{MAX} and negating all edge weights, and solve the $(-p)$-mean partition problem. These two partitions correspond to the $< p$ and $\geq p$ partitions of the original game, and combining the two solutions we get the desired three-partition for only twice the effort.

We now analyze the running time of our algorithms, asymptotically the same for all versions of the problem mentioned in the previous paragraph. The complexity of a strategy improvement algorithm consists of two parts: the cost of computing the measure times the number of iterations necessary. Section 6 demonstrates that this combined cost is at most $O(n^2 \cdot |E| \cdot W)$. This is the same complexity as for the algorithm by Zwick and Paterson for the splitting into three sets problem [26, Theorem 2.4]. If W is very big, the number of iterations can of course also be bounded by $\prod_{v \in V_{\text{MAX}}} \mathit{outdeg}(v)$, the total number of strategies for MAX.

Using the randomization scheme of Matoušek, Sharir, and Welzl from Section 3.4 we obtain the simultaneous bound $2^{O(\sqrt{n \log n})}$, *independent* of W. Combining the bounds, we get the following.

Theorem 7.1 *The decision, p-mean partition, and splitting into three sets problems for mean payoff games can be solved in time*

$$\min\left(O(n^2 \cdot |E| \cdot W),\ 2^{O(\sqrt{n \log n})}\right). \qquad \square$$

Note that a more precise estimation replaces n by $|V_{\text{MAX}}|$ in the subexponential bound, since we only consider strategies of MAX. Also, $n \cdot W$ can be replaced by the length of the longest shortest path, or any upper estimate of it. For instance, one such bound is the sum, over all vertices, of the maximal positive outgoing edge weights.

7.2 Computing the Ergodic Partition

We now explain how to use the solution to the p-mean partition problem to compute the ergodic partition. We first describe the algorithm, which uses an algorithm for the p-mean partition problem with rational thresholds as a subroutine. We then analyze our algorithm for the case of rational thresholds, and finally bound the total running time, including all calls to the p-mean partition algorithm.

Denote by w^- and w^+ the smallest and biggest edge weights, respectively. Then the average weight on any loop (i.e., the value of any vertex in the MPG) is a rational number with denominator $\leq n$ in the interval $[w^-, w^+]$. We can find the value for each vertex by dichotomy of the interval, until each vertex has a value contained in an interval of length $\leq 1/n^2$. There is at most one possible value inside each such interval (the difference between two unequal mean loop values is at least $1/n(n-1)$), and it can be found easily [14,26]. The interval is first partitioned into parts of integer size. After that we deal with rational thresholds p/q, where $q \leq n^2$. We therefore have to solve the p-mean partition problem when the threshold p is rational and not integral. As is readily verified, our algorithm directly applies to this case: only the complexity analysis needs to be slightly changed.

The subexponential bound does not depend on thresholds being integers, but we need to analyze the depth of the measure. After subtracting p/q from each (integral) weight w, it can be written in the form $(qw - p)/q$, so all weights become rationals with the same denominator. The weight of every path of length k to t has the form $(\sum_{i=1}^{k} w_i) - kp/q$. The sum $\sum_{i=1}^{k} w_i$ can take at most $n \cdot W$ different values, and k can take at most n values, so each vertex can improve its value at most $n^2 \cdot W$ times. Thus, solving the 0-mean problem for rational thresholds takes at most n times longer.

During the dichotomy process, we consider subproblems with different thresh-

olds. The thresholds are always in the range $[w^-, w^+]$, so the largest absolute edge weight is linear in W. We will bisect the intervals at most $O(\log(W \cdot n^2))$ times. The total number of vertices in the subproblems solved after bisecting k times is at most n. The complexity function T is superlinear in n, so that $T(i) + T(j) \leq T(i + j)$. Hence the total time for the subproblems after k bisections does not exceed that of solving one n-vertex game. This shows that the whole computation takes time $O(\log(W \cdot n) \cdot T)$, where T is the time for solving the p-mean partition problem. We summarize this in the following theorem.

Theorem 7.2 *The ergodic partition problem for mean payoff games can be solved in time*

$$\min\left(O(n^3 \cdot |E| \cdot W \cdot \log(n \cdot W)), \quad (\log W) \cdot 2^{O(\sqrt{n \log n})}\right). \qquad \square$$

Zwick's and Paterson's algorithm for this problem has the worst-case bound $O(n^3 \cdot |E| \cdot W)$ [26, Theorem 2.3], which is slightly better for small W, but worse for large W. Note that the algorithm [26] exhibits its worst case behavior on simple game instances.

8 Variants of the Algorithm

Theorem 5.1 shows that any combination of attractive switches improves the strategy value, and thus any policy for selecting switches in each iteration will eventually lead to an optimal strategy. In particular, all policies that have been suggested for parity and simple stochastic games apply. These include the all profitable, random single, and random multiple switch algorithms; see, e.g., [1]. In our experiments with large random and non-random game instances these algorithms witness extremely fast convergence. In this section we suggest two alternative ways of combining policies.

Initial Multiple Switching. We can begin with the strategy where MAX goes to the sink t in each vertex and rather than starting the randomized algorithm described in Section 3.4 immediately, instead make a polynomially long sequence of random (multiple) attractive switches, selecting them at each step uniformly at random. Use the last strategy obtained, if not yet optimal, as an initial one in the randomized algorithm above. There is a hypothesis due to Williamson Hoke [25] that every *completely unimodal* function (possessing a unique local maximum on every boolean subcube) can be optimized by the random single switch algorithm in polynomially many steps. The longest-shortest paths problem is closely related to CU-functions. Investigating these

problems may shed light on possibilities of polynomial time optimization for both.

Proceeding in Stages. Another version of the algorithm starts as described in the previous subsection, and afterward always maintains a partition of the vertices (from which the longest shortest distance is not yet $+\infty$) into two sets: R of vertices where MAX is still using the conservative strategy of retreating immediately to the sink t, and N of all other vertices. In all vertices in $N \cap V_{\text{MAX}}$, MAX already switched away from retreating. Since the value of a vertex can only increase, MAX will never change back playing to retreats in those vertices (so retreat edges from vertices in N may be safely removed without influencing the 0-mean partition). We fix the choices in R and proceed as in Section 3.4, to find the best strategy in controlled vertices in N. If the resulting strategy is globally optimal (contains no attractive switches in R), we stop. Otherwise we make some or all attractive switches in vertices in R. Each stage of this version of the algorithm is subexponential, and there are only linearly many such stages, because a vertex leaving R never returns back.

9 The LSP Problem is in NP∩coNP

The decision version of the LSP problem, restricted to determine whether the longest shortest path from a distinguished vertex s to the sink is bigger than a given bound D is another example of a problem in NP∩coNP.

Recall that in the definition of the LSP problem (Section 3.1) we have not stated how the shortest distance to the sink is defined when the MAX player can enforce a zero-weight loop. This was unneeded because such loops are uninteresting for MAX in mean payoff games for reaching a *positive* mean value. Zero-weight loops are impossible in admissible strategies, and only such strategies are needed (visited) by our algorithms to compute zero-mean partitions in games. However, in the LSP problem it is natural to postulate that whenever MAX can enforce a zero-weight cycle, the distance to the (unreachable) sink becomes $+\infty$. We show here that a minor modification allows us to reuse Theorems 5.1 (attractiveness implies profitability) and 5.2 (stability implies optimality), as well as subexponential algorithms from Sections 3.3, 3.4, to compute longest shortest paths, and to prove the NP∩coNP-membership.

The necessary modification is achieved by making the following simplifying assumption about the instances of the LSP problem.

Assumption. *A graph in an LSP problem instance does not contain* zero-*weight loops.* □

This assumption may be done without loss of generality. Indeed, if the graph G has n vertices, we can multiply all edge weights by $n+1$ and add 1. As a result, zero loops will disappear (become positive), and the lengths l, l' of all paths/simple loops in G and the modified graph G' will only differ within the factor of $(n+1)$, i.e., $l(n+1) < l' \leq l(n+1)+n$. Consequently, all negative/positive loops in the original graph will preserve their signs.

Proposition 9.1 *The decision version of the LSP problem (subject to the assumption above) is in* NP∩coNP.

Proof. First note that we can add retreat edges to any LSP problem instance similarly as to MPGs: from any controlled vertex, make an extra edge to the sink with value $-2 \cdot n \cdot W - 1$. Thus, we guarantee that there is an admissible strategy, namely the one always using the retreat edge. In a solution to the transformed LSP problem, a vertex has value $< -n \cdot W$, iff it can only reach the sink through a retreat edge, iff it has value $-\infty$ in the original problem.

Both for YES- and NO-instances, the short witness is an optimal (stable) positional strategy σ in controlled vertices of the transformed problem. By computing the shortest paths to the sink in G_σ, i.e., computing the strategy measure, it can be verified in polynomial time that no switch is attractive and thus that the strategy is optimal by Theorem 5.2. This can be used as a witness for YES-instances by testing if the value is $> D$, and for NO-instances by testing whether the value is $\leq D$. □

The absence of zero-weight loops is essential in the proof above. The example in Figure 3 demonstrates a zero-weight loop, and a stable strategy, which does not provide the optimal (in the LSP sense) solution.

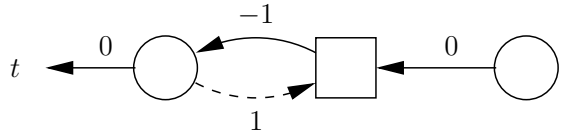

Figure 3. Switching to the dotted edge is not attractive, but improves the values in all vertices to $+\infty$.

For mean payoff games we were satisfied with the definition of optimality that only stipulates that Max can secure *positive-weight* loops whenever possible. Thus, in Figure 3 the strategy "go to t" in the leftmost vertex is MPG-optimal, but not LSP-optimal. In contrast, the LSP-optimality makes zero-weight loops attractive (shortest distance to the sink equals $+\infty$). This was essentially used in the proof of Proposition 9.1 and can always be achieved by making the initial transformation to satisfy the Assumption.

10 LSP: Exponential Sequences of Attractive Switches

One might conjecture that any sequence of attractive switches converges fast on any LSP problem instance, and consequently MPGs are easily solvable by iterative improvement. It is not so easy to come up with "hard" LSP examples. In this section we present a set of instances of the LSP problem and an improvement policy, selecting an attractive switch in every step, leading to exponentially long sequences of strategy improvements. This shows that the LSP problem is nontrivial, and the choice of the next attractive switch is crucial for the efficiency.

Consider the $(2n+2)$-vertex graph in Figure 4, where round vertices belong to MAX, square ones to MIN, and the leftmost vertex is the sink. The optimal strategy of MAX is marked by the dashed edges. Adding N is unnecessary here (N may be set to 0 for simplicity), but will be needed later.

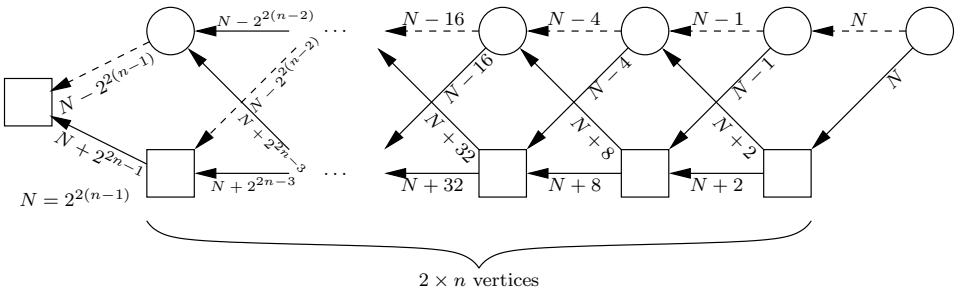

Figure 4. LSP instances allowing for exponentially long chains of improving switches.

If the iterative improvement algorithm starts from the strategy "go left everywhere" and always makes the *rightmost* attractive single switch, it visits all 2^n possible strategies of MAX, as can be readily verified.

The LSP instances above can be generated from MPGs as follows. In Figure 4, add a self-loop of weight zero in the leftmost vertex. Add the retreat vertex and edges as explained in Section 3.2. If the algorithm initially switches from the "go to the retreat everywhere" strategy to the "go left everywhere", and then always chooses the rightmost attractive single switch, it follows exactly the exponentially long chain of improving strategies, as in the LSP case. Adding N now is essential to keep all the paths nonnegative; otherwise, the initial "switches from the retreat" would be non-attractive.

Actually, the LSP-instances in Figure 4 are "trivial", because the graphs are acyclic and longest-shortest distances are easily computable by dynamic programming in polynomial time, as mentioned in the end of Section 3.1. These LSP-instances are also easily solvable by the "random single switches" policy, selecting an attractive switch uniformly at random. The reason is as follows. The second from the left dashed edge remains always attractive, and once the

algorithm switches to this edge, it will never switch back. Such an edge defines a so-called *absorbing facet*. Obviously, the random single switch algorithm is expected to switch to the absorbing facet in polynomially many, namely $O(n)$, steps. The problem that remains has one dimension less, and the preceding dashed edge determines an absorbing facet. The algorithm converges in $O(n^2)$ expected iterations. Currently we are unaware of any LSP instances that require superpolynomially many steps of the single random, multiple random, or all profitable (attractive) switches algorithms.

The example of this section is inspired by the one due to V. Lebedev mentioned in [14], and kindly provided by V. Gurvich. Unlike the GKK-algorithm [14], which is deterministic and bound to perform the exponential number of iterations, corresponding exactly to the exponential sequence determined by the rightmost attractive switches above, our randomized algorithms quickly solve the examples from this section.

11 Application to Parity Games

The algorithm described in this paper immediately applies to parity games, after the usual translation; see, e.g., [23]. Parity games are similar to mean payoff games, but instead of weighted edges they have vertices colored in nonnegative integer colors. Player Even (Max) wants to ensure that in every infinite play the largest color appearing infinitely often is *even*, and player Odd (Min) tries to make it *odd*. Parity games are determined in positional strategies [13,5].

Transform a parity game into a mean payoff game by leaving the graph and the vertex partition between players unchanged, and by assigning every vertex [8] of color c the weight $(-n)^c$, where n is the total number of vertices in the game. Apply the algorithm described in the preceding sections to find a 0-mean partition. The vertices in the partition with value > 0 are winning for Even and all other are winning for Odd in the parity game.

Actually, the new MPG algorithm, together with a more economic translation and more careful analysis, gives a considerable improvement for parity games over our previous algorithm from [2], which has complexity

$$\min\left(O(n^4 \cdot |E| \cdot k \cdot (n/k+1)^k),\ 2^{O(\sqrt{n \log n})}\right),$$

where n is the number of vertices, $|E|$ is the number of edges, and k is the

[8] Formally, we have to assign this weight to every edge leaving a vertex, but it makes no difference.

number of colors. Thus, in the worst case the first component in the upper bound may be as high as $O(|E| \cdot n^5 \cdot 2^n)$, when the number of colors equals the number of vertices. The MPG algorithm improves the first component in the upper bound to $O(n^2 \cdot |E| \cdot (n/k+1)^k)$, thus gaining a factor of $n^2 \cdot k$. To prove this bound, we assign weights to vertices more sparingly than in the mentioned reduction. It is readily verified that we may equally well give a vertex of color c the weight $(-1)^c \cdot \prod_{i=0}^{c-1}(n_i+1)$, where n_i is the number of vertices of color i. In the worst case (when all n_i are roughly equal), this gives $W = O((n/k+1)^k)$.

There are two reasons for this improvement. First, each vertex can improve its value at most $n \cdot (n/k+1)^k$ times, compared with $n^2 \cdot (n/k+1)^k$ in [2] (this is because the measure for every vertex in [2] is a triple, rather a single number, containing additionally the loop color and the path length, now both unneeded; however the shortest distance may be as big as the sum of all positive vertices). Second, we now apply the simpler and more efficient counterstrategy algorithm from Section 6, which saves an extra factor nk.

Moreover, translating a parity game into an MPG allows us to assign weights even more sparingly, and this additionally improves over $(n/k+1)^k$. Indeed, if we want to assign a minimum possible weight to a vertex of color i, we may select this weight (with an appropriate sign) equal to the total absolute weight of all vertices of preceding colors of opposite parity, plus one. This results in a sequence $w_0 = 1$, $w_1 = -(n_0 \cdot w_0 + 1)$, $w_{i+2} = -(n_{i+1} \cdot w_{i+1} + n_{i-1} \cdot w_{i-1} + \cdots + 1) = -n_{i+1} \cdot w_{i+1} + w_i$. In the case of $k = n$ (one vertex per color) it results in the Fibonacci sequence with alternating signs: $w_0 = 1$, $w_1 = -2$, $w_{i+2} = -w_{i+1} + w_i$, which is, in absolute value, asymptotically $O(1.618^n)$, better than 2^n.

Finally, the new MPG-based algorithm for parity games is easier to explain, justify, and implement.

12 Conclusions

We defined the longest shortest path (LSP) problem and showed how it can be applied to create a discrete strategy evaluation for mean payoff games. Similar evaluations were already known for parity [24,2], discounted payoff [23], and simple stochastic games [17], although not discrete for the last two classes of games. The result implies that any strategy improvement policy may be applied to solve mean payoff games, thus avoiding the difficulties of high precision rational arithmetic involved in reductions to discounted payoff and simple stochastic games, and solving associated linear programs.

Combining the strategy evaluation with the algorithm for combinatorial linear programming suggested by Matoušek, Sharir, and Welzl, yields a $2^{O(\sqrt{n \log n})}$ algorithm for the mean payoff game decision problem.

An interesting open question is whether the LSP problem is more general than mean payoff games, and if it can find other applications. We showed that it belongs to NP $\cap$ coNP, and is solvable in expected subexponential randomized time.

The strategy measure presented does not apply to all strategies, only admissible ones, which do not allow nonpositive weight loops. This is enough for the algorithm, but it would be interesting to know if the measure can be modified or extended to the whole strategy space, and in this case if it would be completely local-global, like the measures for parity and simple stochastic games investigated in [4].

The major open problem is still whether any strategy improvement scheme for the games discussed has polynomial time complexity.

Acknowledgments. We thank DIMACS for providing a creative working environment. We are grateful to Leonid Khachiyan, Vladimir Gurvich, and Endre Boros for inspiring discussions and illuminating ideas.

References

[1] H. Björklund, S. Sandberg, and S. Vorobyov. Complexity of model checking by iterative improvement: the pseudo-Boolean framework. In M. Broy and A. Zamulin, editors, *Andrei Ershov Fifth International Conference "Perspectives of System Informatics"*, volume 2890 of *Lecture Notes in Computer Science*, pages 381–394, 2003.

[2] H. Björklund, S. Sandberg, and S. Vorobyov. A discrete subexponential algorithm for parity games. In H. Alt and M. Habib, editors, *20th International Symposium on Theoretical Aspects of Computer Science, STACS'2003*, volume 2607 of *Lecture Notes in Computer Science*, pages 663–674, Berlin, 2003. Springer-Verlag.

[3] H. Björklund, S. Sandberg, and S. Vorobyov. On combinatorial structure and algorithms for parity games. Technical Report 2003-002, Department of Information Technology, Uppsala University, 2003. `http://www.it.uu.se/research/reports/`.

[4] H. Björklund, S. Sandberg, and S. Vorobyov. Randomized subexponential algorithms for parity games.

Technical Report 2003-019, Department of Information Technology, Uppsala University, 2003. http://www.it.uu.se/research/reports/.

[5] H. Björklund, S. Sandberg, and S. Vorobyov. Memoryless determinacy of parity and mean payoff games: A simple proof. *Theoretical Computer Science*, 310(1-3):365–378, 2004.

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

[7] A. Condon. On algorithms for simple stochastic games. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 13:51–71, 1993.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill Book Company, Cambridge, MA, 2nd edition, 2001.

[9] A. Ehrenfeucht and J. Mycielski. Positional games over a graph. *Notices Amer. Math Soc.*, 20:A–334, 1973.

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

[11] E. A. Emerson. Model checking and the Mu-calculus. In N. Immerman and Ph. G. Kolaitis, editors, *DIMACS Series in Discrete Mathematics*, volume 31, pages 185–214, 1997.

[12] E. A. Emerson, C. Jutla, and A. P. Sistla. On model-checking for fragments of μ-calculus. In C. Courcoubetis, editor, *Computer Aided Verification, Proc. 5th Int. Conference*, volume 697, pages 385–396. Lect. Notes Comput. Sci., 1993.

[13] E. A. Emerson and C. S. Jutla. Tree automata, μ-calculus and determinacy. In *Annual IEEE Symp. on Foundations of Computer Science*, pages 368–377, 1991.

[14] V. A. Gurvich, A. V. Karzanov, and L. G. Khachiyan. Cyclic games and an algorithm to find minimax cycle means in directed graphs. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 28(5):85–91, 1988.

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

[16] G. Kalai. A subexponential randomized simplex algorithm. In *24th ACM STOC*, pages 475–482, 1992.

[17] W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and Computation*, 117:151–155, 1995.

[18] J. Matoušek, M. Sharir, and M. Welzl. A subexponential bound for linear programming. In *8th ACM Symp. on Computational Geometry*, pages 1–8, 1992.

[19] J. Matoušek, M. Sharir, and M. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16:498–516, 1996.

[20] H. Moulin. Extensions of two person zero sum games. *J. Math. Analysis and Applic.*, 55:490–508, 1976.

[21] H. Moulin. Prolongements des jeux de deux joueurs de somme nulle. *Bull. Soc. Math. France*, 45, 1976.

[22] N. Pisaruk. Mean cost cyclical games. *Mathematics of Operations Research*, 24(4):817–828, 1999.

[23] A. Puri. *Theory of hybrid systems and discrete events systems.* PhD thesis, EECS Univ. Berkeley, 1995.

[24] J. Vöge and M. Jurdziński. A discrete strategy improvement algorithm for solving parity games. In E. A. Emerson and A. P. Sistla, editors, *CAV'00: Computer-Aided Verification*, volume 1855 of *Lect. Notes Comput. Sci.*, pages 202–215. Springer-Verlag, 2000.

[25] K. Williamson Hoke. Completely unimodal numberings of a simple polytope. *Discrete Applied Mathematics*, 20:69–81, 1988.

[26] U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.*, 158:343–359, 1996.

Paper V

DIMACS Technical Report 2004-41
September 2004

The Controlled Linear Programming Problem

by

Henrik Björklund, Olle Nilsson, Ola Svensson, Sergei Vorobyov [1] [2] [3] [4] [5]

[1]DIMACS visitors, August, 2004.
[2]Uppsala University, Information Technology Department, Box 337, 751 05 Uppsala, Sweden.
[3]Supported by the Grant IG2003-2 067 from the Swedish Foundation for International Cooperation in Research and Higher Education (STINT, `www.stint.se`) and the Swedish Research Council grants "*Infinite Games: Algorithms and Complexity*" and "*Interior-Point Methods for Infinite Games*".
[4]DIMACS hosts: Leonid Khachiyan and Endre Boros
[5]Email: `vorobyov@csd.uu.se`

ABSTRACT

We introduce and investigate the new Controlled Linear Programming Problem. In a system of linear constraints of the form $x_i \leq p_i^j(\bar{x}) + w_i^j$, where p_i^j are linear homogeneous polynomials with nonnegative coefficients, some variables are *controlled* and the controller wants to select exactly one constraint for each controlled variable in a way that makes $\max \sum x_i$ subject to the remaining constraints as large as possible (over all selections). We suggest several iterative strategy improvement policies (simplex-like and interior-point), prove optimality conditions in several important cases, describe subexponential algorithms, and interior point ones, and show that the decision version of the problem is a member of NP∩coNP whenever stability yields optimality, and also when all coefficients are nonnegative integers.

It turns out that the well known mean payoff, discounted payoff, and simple stochastic games are easily reducible to the Controlled LP-Problem. This suggests a simple unifying view of all these games as particular restricted instances of the problem.

We show that a slight generalization of the problem allowing for negative coefficients in the constraint polynomials p_i^j leads to NP-hardness.

Contents

1 Introduction

Consider a succinctly represented family $\mathcal{F} = \{P_i\}_{i \in I}$ of instances of a polynomially solvable optimization problem. Find an instance with the *largest* possible value of the associated objective function. We are better off with a concrete example.

Longest Shortest Paths. Let P_i's be weighted digraphs with distinguished source s and sink t. Find the graph with the *longest* shortest s-t-distance. As an example of succinct representation, consider a digraph G with a distinguished subset of *controlled* vertices C, and generate P_i's by selecting exactly one outgoing edge per controlled vertex. This gives a compact representation of an exponential in $|C|$ family of graphs. This problem, called the *Longest Shortest Paths (LSP)* problem, has useful applications and is investigated in [4].

The example of a *controlled* problem above, in which the controller selects options in order to maximize the target function over all possible selections, seems deceptively simple to generalize. Take *Max-Bipartite Matching* or *Max-Flow* and allow the controller to choose edges from some vertices so as to minimize the resulting maximum matching/flow. Natural and simple as these problems sound, they turn out to be NP-hard; see Section 12.

The questions which immediately appear are as follows. Can we equip the family $\mathcal{F}$ with a meaningful polynomial neighborhood structure and corresponding target/quality function[1] allowing for a polynomial local search (PLS)? Combinatorially we get a directed graph with family members as vertices and edges corresponding to quality improvements. How fast can we find local maxima in this graph? It turns out that our chances are better than exponential in the LSP problem [5]. How about the quality of local maxima in this structure? How can we verify that a given local maximum is global? Can we guarantee that *every* local maximum is global? All those questions have satisfactory positive solutions in the case of the LSP problem.

Now the main question. Are there any other useful *controlled* optimization problems with the positive (satisfactory) answers to the questions above? Do they have good and interesting applications?

In this paper we introduce and investigate another problem of this kind, dubbed the Controlled LP-Problem. In a system of linear constraints of the form $x_i \le p_i^j(\bar{x}) + w_i^j$, where p_i^j are linear homogeneous polynomials with nonnegative coefficients and $w_i^j \in \mathbb{R}$, some variables are *controlled* and the controller wants to select exactly one constraint for each controlled variable in a way that makes $\max \sum x_i$, subject to the remaining constraints, as large as possible (over all selections).

It turns out that the well known mean payoff, discounted payoff, and simple stochastic games [9, 11, 7, 22], as well as the LSP problem [5], are easily reducible to the Controlled LP-Problem. This suggests a simplified unifying view of all these games as particular restricted instances of the problem, and proves its usefulness. The subexponential algorithms for the CLPP (Section 11) are immediately applicable to solve all the above games.

The CLPP with nonnegative coefficients demonstrates a robust structure amenable to a

[1] Note that there are target functions in each instance, now we are talking about the "second-order" function comparing quality of instances.

polynomial local search (PLS). Looking at unselected controller constraints, which have less restrictive right-hand sides in the optimal solution, compared to the current selection, allows for defining a numerous monotonic iterative improvement schemes (discrete and interior point attractive switches; Sections 3, 4). As a result, the problem of finding local maxima for the CLPP in this case can be solved in subexponential time (Section 11). In many interesting cases of the CLPP with nonnegative coefficients, we also prove that a stable (not locally improvable) pure strategy (selection) of constraints is also globally optimal (Section 7). In these cases we can find global maxima in subexponential time, and the corresponding decision problem (is there a selection providing for the optimum value above the threshold) is in NP∩coNP (Section 7.4). We also investigate a case of the CLPP with nonnegative integer coefficients, which is in NP∩coNP, but without the "stability ⇒ optimality" property and without known subexponential iterative improvement algorithms (Section 8). This shows, in particular, that there is a CLPP subclass, which belongs to NP∩coNP, while being more general than any of the games mentioned above.

The next natural step in generalization leads to (presumable) intractability. We show show that a slight generalization of the CLPP, allowing for negative coefficients in the polynomials p_i^j, leads to NP-hardness. As a result, we have a sufficiently sharp classification of NP∩coNP and NP cases in terms of nonnegative/arbitrary coefficients in the right-hand sides of the CLPP constraints. Our examples of the Controlled Bi-Partitite Matching and Controlled Max-Flow confirm this classification (Section 12).

Notations, Conventions. By boldface $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{1}$ we denote column vectors of variables x_i, y_i, and ones, and by $\mathbf{x}^T$ the transposed row vector. A linear polynomial is called *homogeneous* if it contains no free members, i.e., of the form $\mathbf{c}^T\mathbf{x}$.

2 The Controlled Linear Programming Problem

Let $\mathbf{x} = (x_1, \ldots, x_n)$, $\mathbf{y} = (y_1, \ldots, y_m)$ be two vectors of variables, called *controlled* and *uncontrolled*, respectively. Let $p_i^j(\mathbf{y})$ and $q_k^l(\mathbf{x})$ be homogeneous linear polynomials with *non-negative* real coefficients, and $w_i^j, w_k'^l \in \mathbb{R}$, for $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, n_i\}$, $k \in \{1, \ldots, m\}$, $l \in \{1, \ldots, m_k\}$. A linear constraint is called:

- *optional* if it has form $x_i \leq p_i^j(\mathbf{y}) + w_i^j$, and
- *compulsory* if it has form $y_k \leq q_k^l(\mathbf{x}) + w_k'^l$.

(For notational convenience we assume "bi-partiteness": x_i's and y_j's alternate in constraint sides. This is no loss of generality and can always be achieved by introducing new variables.)

Definition 2.1 (Controlled LP-Problem) *Given the data as above, find a tuple, called a* positional strategy, $(j_1, \ldots, j_n) \in \prod_{j=1}^{n} \{1, \ldots, n_j\}$ *that renders maximal (over all possible such tuples) the value of the objective function* $\sum_{i=1}^{n} x_i + \sum_{i=1}^{m} y_i$ *subject to the system of constraints* $S(j_1, \ldots, j_n)$ *consisting of all compulsory and the optional constraints* $x_i \leq p_i^{j_i}(\mathbf{y}) + w_i^{j_i}$ *for* $i = 1, \ldots, n$. □

Quite often we want to maximize the values of all variables. This is not the same as maximizing their sum in case some variable is unbounded. Section 3.1 explains how to proceed in this case.

Proviso. To avoid immediate unboundedness of a feasible constraint system, we assume that every variable occurs in the left-hand side of some constraint. □

The name for this problem comes from the intuition that the controller selects *exactly one* optional constraint for each controlled variable x_i to maximize the objective function. In general, some selections of optional constraints may lead to infeasible systems (the value of objective function is $-\infty$), some to polyhedra unbounded in some coordinate directions (the value $+\infty$), and some to nonempty polyhedra with a finite optimum of the target function.

It would be more appropriate (but would overload the terminology) to call the problem the Positive Controlled LP-Problem, since we impose the positiveness of the coefficients in polynomials. We show that this restriction is crucial for efficiency; see Section 9. Whenever it causes no confusion, we omit the adjective "positive".

Definition 2.1 introduces the optimization problem. As usual, the corresponding decision version is defined by giving an additional bound and asking whether there exists a selection rendering the objective function larger than that bound.

3 Discrete Switching Algorithms

A selection $\lambda = (j_1, \ldots, j_n)$ of optional constraints, is called *a pure*[2] *strategy*. Denote $S(j_1, \ldots, j_n)$ the resulting system of constraints obtained by taking all compulsory constraints and leaving only the j_k-th optional constraint for each variable x_k, $k = 1, \ldots, n$. Say that a strategy λ is *admissible* if $S(\lambda)$ is feasible (has a solution).

Consider an optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ to a feasible system $S(j_1, \ldots, j_n)$, maximizing the target function $\mathbf{1}^T\mathbf{x} + \mathbf{1}^T\mathbf{y}$, symbolically,

$$(\mathbf{x}^*, \mathbf{y}^*) = \arg\max\{\mathbf{1}^T\mathbf{x} + \mathbf{1}^T\mathbf{y} \text{ subject to } S(j_1, \ldots, j_n)\}.$$

Say that a *switch* for variable x_k from j_k-th to j'_k-th constraint is *attractive*, if

$$p_k^{j_k}(\mathbf{y}^*) + w_k^{j_k} < p_k^{j'_k}(\mathbf{y}^*) + w_k^{j'_k}. \tag{1}$$

Say that a sequence of switches for variables $x_1, \ldots, x_n$ from $(j_1, \ldots, j_n)$ to $(j'_1, \ldots, j'_n)$ is *profitable*, if for each variable v in $\mathbf{x}$, $\mathbf{y}$ one has

$$\max\{v \text{ subject to } S(j_1, \ldots, j_n)\} \;\leq\; \max\{v \text{ subject to } S(j'_1, \ldots, j'_n)\}, \tag{2}$$

and for some variable the inequality (2) is *strict*.

We have the following two important theorems, underlying a wide range of iterative improvement algorithms for the controlled LP-problem. The first theorem guarantees strict monotonic improvement of attractive switches. It will be the corollary of the more general theorem in Sections 5.

Theorem 3.1 *Every sequence of attractive switches is profitable and transforms an admissible strategy into an admissible one.* □

Say that an admissible strategy is *stable* if it does not have attractive switches. For many important cases of the Controlled LP-Problem, discussed below, we have the following polynomial-time optimality test.

Theorem 3.2 *Every stable strategy is optimal.* □

Theorems 3.1 and 3.2 immediately suggest the following correct and complete generic algorithm for solving, in some cases, the Controlled LP-Problem.

Generic Algorithm for Controlled LP

1. Start with any admissible initial strategy (we discuss below how to select such strategies.)

[2] or discrete

2. Proceed by making any attractive switches, until obtaining a stable strategy. (Simplify the system if necessary by getting rid of unbounded variables and associated constraints. It is essential that this simplification takes place only after reaching a stable strategy; see Section 3.1.)

3. In many important cases (Sections 7) the resulting stable strategy is also *optimal*.

Specific iterative strategy improvement algorithms are obtained by imposing a policy on selecting attractive switches at every iteration. Some possible policies are: single random switch, all attractive switches, random multiple attractive switches (Section 11.1). We also discuss two randomized subexponential algorithms for the CONTROLLED LP-PROBLEM, based on adapting the schemes for LP due to Kalai [13] and Matoušek, Sharir, Welzl [16] (Section 11.2). In general, there are examples of "poor" switching policies leading to an exponential number of switches [5], so the problem is by no means trivial.

3.1 Eliminating Unbounded Variables and Constraints

In many circumstances we want to maximize the values of all variables, rather than just the sum of all variables. In this case we need to get rid of the unbounded variables, simplify the system, and proceed with the remaining variables. Here is the recipe.

Theorem 3.3 *Eliminating unbounded variables and constraints from the system $S(\lambda)$, does not affect the optimal values of the remaining variables, provided that λ is stable.*

Proof. Assume the value of a controlled variable x_i is bounded in $S(\lambda)$. Since x_i has no attractive switches, every unbounded variable must have the coefficient 0 in every constraint for x_i. Therefore, x_i will not be affected by the elimination of the unbounded variables. Suppose the value of an uncontrolled variable y_i is bounded in $S(\lambda)$. Then it has a bounded constraint. Since the unbounded constraints never will tighten the value of y_i, even if the bounded constraints go unbounded, we can safely remove them. □

Eliminating unbounded variables allows us to assume in the premises of theorems below, without loss of generality, that a system has a finite optimal solution under a current strategy.

It is easy to construct an example where an elimination in a system with an unstable strategy has a devastating effect to the solution. Consider the unstable strategy ((1,0),(1,0), (0,1)) in Figure 1 in Section 6 and the notational explanations there. Eliminating the unbounded variables and constraints will give v_3 the final value of 1, while its correct value is infinity.

3.2 Retreats: Boundedness and Feasibility

An alternative way of avoiding unboundedness is to add *retreat* constraints for the uncontrolled variables. They are called retreats because we can imagine a minimizing player

(opponent) choosing constraints for the uncontrolled variables, and using the retreats only when the value for a variable threatens to become unbounded.

To do this, compute an upper bound M on the largest *finite* value a variable can take in the solution to the linear program arising from any strategy of the maximizer. Then add a constraint $y_k \leq M$, for each uncontrolled variable y_k. After solving the new system, any variable that gets a value close to M has an unbounded value in the optimal solution to the original problem.

Similarly, we can add retreats for the maximizer, in order to ensure that there is at least one strategy that yields a feasible linear program. Add a constraint $x_i \leq -M$ for each controlled variable x_i. Now the strategy that for every controlled variable selects the retreat constraint is *admissible*.

4 Interior Point Algorithms

The previous discrete switching algorithms can be seen as a simplex-like, visiting the vertices of the Cartesian product $\prod_{i=1}^{n}\{1,\ldots,n_i\}$ (the space of pure/discrete strategies), monotonically improving the target function value. Let us extend this by allowing the algorithm to visit the *interior* of the product of simplices. The adequate setting for this is as follows (previously suggested and investigated by us in the context of simple stochastic games in [19, 18]).

For every controlled vertex x_i let a nonnegative vector $\bar{\lambda}_i = (\lambda_i^1,\ldots,\lambda_i^{n_i})$ have the l_1-norm equal to 1, i.e., $\sum_{j=1}^{n_i}\lambda_i^j = 1$. Combine all the constraints for each controlled variable x_i into a weighted sum:

$$\sum_{j=1}^{n_i}\lambda_i^j(p_i^j(\mathbf{y}) + w_i^j). \tag{3}$$

Collect all the constraints (3), for all controlled variables, together with all compulsory constraints, and denote the resulting system by $S(\lambda) = S(\bar{\lambda}_1,\ldots,\bar{\lambda}_n)$. Call the vector of vectors $\lambda = (\bar{\lambda}_1,\ldots,\bar{\lambda}_n)$ *a mixed strategy*. Each such mixed strategy is an interior point of the product of simplices. In contrast a strategy $\bar{\lambda}_i$ with exactly one component λ_i^j equal 1 is called *pure*. Call a mixed strategy λ *admissible* if the system of constraints $S(\lambda)$ is feasible.

Similar to the discrete case, we will introduce a notion of attractive interior point switch, and will prove three facts, guaranteeing the correctness of a wide range of interior point algorithms:

1. for an admissible strategy, any combination of attractive switches leads to an admissible strategy and a better target function value (Section 5);

2. for any stable mixed strategy (no attractive switches) there is a pure no-worse stable strategy (Section 6);

3. every pure stable strategy is optimal, in many important cases discussed in Section 7.

Now the Generic Algorithm from Section 3 works perfectly well for the generalized case of interior-point switches.

5 Attractiveness and Profitability

Definition 5.1 *Given an admissible interior point strategy* $\lambda = (\bar{\lambda}_1, \bar{\lambda}_2, ..., \bar{\lambda}_n)$*, let* $(\mathbf{x}^*, \mathbf{y}^*)$ *be an optimal solution to the system* $S(\lambda)$*. Then an interior point (single) switch for variable* x_i *from* $\bar{\lambda}_i$ *to* $\bar{\lambda}'_i$ *is called* attractive*, if*

$$\sum_j \lambda_i^j (p_i^j(\mathbf{y}^*) + w_i^j) < \sum_j \lambda_i'^j (p_i^j(\mathbf{y}^*) + w_i^j). \tag{4}$$

Say that the switch from λ *to another strategy* $\lambda' = (\bar{\lambda}'_1, \ldots, \bar{\lambda}'_n)$ *is* profitable*, if* λ' *is admissible and for each variable* v *in* $\mathbf{x}$*,* $\mathbf{y}$ *one has*

$$\max\{v \textit{ subject to } S(\lambda)\} \;\leq\; \max\{v \textit{ subject to } S(\lambda')\}, \tag{5}$$

and for some variable the inequality (5) *is* strict*.* □

Theorem 5.2 *Any combination of attractive switches is profitable.*

Proof. Assume attractive switches were made, from strategy λ to λ'. Let $(\mathbf{x}^*, \mathbf{y}^*)$ be the optimal solution to the old LP $S(\lambda)$. For every switched variable x_i, we have the old constraint $x_i \leq \sum_j \lambda_i^j (p_i^j(\mathbf{y}) + w_i^j)$, and the new constraint $x_i \leq \sum_j \lambda_i'^j (p_i^j(\mathbf{y}) + w_i^j)$. By definition of an attractive switch, we have (4) satisfied.

Then it is easy to see that $(\mathbf{x}^*, \mathbf{y}^*)$ is a solution to the new LP $S(\lambda')$, because every constraint is either the same or more generous, i.e., has a larger right-hand-side, by (4).

Take one switched variable x_i, let $x'_i = x_i + \varepsilon$, for a small enough $\varepsilon > 0$. Then $(\mathbf{x}^*, \mathbf{y}^*)$, where x_i in $\mathbf{x}^*$ is replaced by x'_i is a solution to the new LP $S(\lambda)$. This follows from the two facts:

1. x'_i does not violate its constraint because the attractive switch made it more generous.

2. The increase of x_i does not offend any other constraints, because the polynomials $p_i^j(\mathbf{y})$ and $q_k^l(\mathbf{x})$ have only non-negative coefficients, so no constraint is tightened by an increase in x_i.

We have thus shown that the resulting strategy is admissible and that there exists a better solution to the new LP. Therefore, the switch is profitable. In addition, we see that the values of variables where attractive switches are made are guaranteed to increase in an optimal solution to the new system.

Note that the argument above essentially uses the fact that the polynomials p_i^j and q_i^j in right-hand sides of constraints have nonnegative coefficients. The above argument fails, if negative coefficients are allowed. □

Now Theorem 3.1 follows immediately, as a corollary.

6 Discretization

In this section we show that every stable mixed (interior-point) strategy λ can be made discrete (pure). This simplification allows us to concentrate on discrete/pure strategies while proving optimality of stable strategies in Section 7.

Theorem 6.1 *For every mixed stable strategy there exists a discrete/pure strategy providing for a no-worse value of the target function.*

Proof. Assume $\lambda = (\bar{\lambda}_1, \ldots, \bar{\lambda}_n)$ is mixed and stable, and the optimal solution to the system $S(\lambda)$ is $(\mathbf{x}^*, \mathbf{y}^*)$. For every non-discrete $\bar{\lambda}_i = (\lambda_i^1, \ldots, \lambda_i^{n_i})$ consider all $\lambda_i^j > 0$. For all such λ_i^j the values of the corresponding right-hand sides of the constraints $p_i^j(\mathbf{y}^*) + w_i^j$ are the same, since there are no attractive switches. Denote this value z.

Obtain $\bar{\lambda}'_i$ from $\bar{\lambda}_i$ by discretization as follows. Take any k such that $\lambda_i^k > 0$, set $\lambda_i'^k$ equal one, and all other components of $\bar{\lambda}'_i$ equal zero. Since $x_i \le \sum_{j=1}^{n_i} \lambda_i^j (p_i^j(\mathbf{y}^*) + w_i^j) = \sum_{j=1}^{n_i} \lambda_i^j z = 1 \cdot z = p_i^k(\mathbf{y}^*) + w_i^k$, switching from $\bar{\lambda}_i$ to $\bar{\lambda}'_i$ leaves the solution $(\mathbf{x}^*, \mathbf{y}^*)$ feasible. Therefore, the value of the target function does not get worse. Repeat the same argument for each non-discrete component $\bar{\lambda}_i$. □

In Figure 1 we give the graph representation of the following controlled linear program, where all constraints are optional:

$$\left\{ \begin{array}{lcl} v_1 & \le & v_2, \\ v_1 & \le & 0, \end{array} \right. \qquad \left\{ \begin{array}{lcl} v_2 & \le & v_1, \\ v_2 & \le & 0, \end{array} \right. \qquad \left\{ \begin{array}{lcl} v_3 & \le & v_2, \\ v_3 & \le & 1. \end{array} \right.$$

As an example, a mixed (actually, pure) strategy $((1,0),(1,0),(0,1))$ selects the first constraints for $v_{1,2}$ and the second for v_3.

It is conceivable, as Figure 1 shows, that after discretization described in the proof above the resulting strategy becomes *unstable*. In this case it will be enough to proceed with attractive switches, and apply the argument again. This will eventually result in a discrete stable strategy.

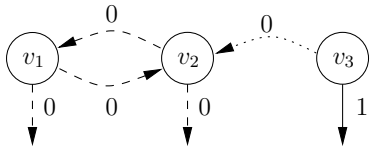

Figure 1: In this controlled LP there is a strategy that is stable before but not after discretization. Round vertices are controlled.

Consider the mixed strategy $((1/2,1/2),(1/2,1/2),(0,1))$, in which the controller selects for $v_{1,2}$ each of the optional constraints with equal probability $1/2$, and for v_3 purely selects $v_3 \le 1$. This strategy is stable yielding the optimal target value 1, but its discretization to $((1,0),(1,0),(0,1))$ becomes unstable, the attractive switch in v_3 to $v_3 \le v_2$ makes all three variables unbounded.

7 Stability and Optimality

As discussed in Section 7.5, in general, a stable strategy in an instance of the Positive Controlled LP-Problem is *not necessarily optimal*. In this and subsequent sections we also present several important particular cases when the desired implication "stability ⇒ optimality" holds, i.e., a stable strategy is guaranteed to be optimal. This suggests an easy polynomial-time verification of optimality, used by all our algorithms in Section 11.

7.1 Univariate Discounted Case

Let $0 < \delta < 1$ be fixed, and consider the Positive Controlled LP-Problem with restricted right-hand sides of one of the forms:

$$x \leq w + \delta \cdot y, \tag{6}$$
$$y \leq w + \delta \cdot x, \tag{7}$$
$$y \leq 0, \tag{8}$$

where x is a controlled variable, y is a noncontrolled variable, and $w \in \mathbb{R}$. In other words, we restrict the number of variables in the right hand sides to at most one, with the coefficient smaller than one. Note that constraints of the form $y \leq w$ may be introduced as abbreviations for $y \leq w + \delta x'$, $x' \leq 0 + \delta y'$, $y' \leq 0$.

Theorem 7.1 *Every discrete/pure stable strategy of the controller (*Max*) in the* Positive Controlled LP-Problem *instance with constraints of the form* (6) – (8) *is optimal.*

Proof. Let λ be a stable strategy in a problem instance S. Assume $S(\lambda)$ is feasible, and let $(\mathbf{x}^*, \mathbf{y}^*)$ be an optimal solution. We claim and prove shortly that this optimal solution is *finite*.

A function p mapping variables in $\mathbf{x}$ and $\mathbf{y}$ to $\mathbb{R}$ is called a *potential function* for the system $S(\lambda)$ if for every constraint $u \leq w + \delta \cdot v$ ($y \leq 0$, resp.) one has $p(u) \leq w + \delta \cdot p(v)$ ($p(y) \leq 0$, resp.).

It is easy to see that the components of a finite optimal solution to $S(\lambda)$ determine potentials for the system $S(\lambda)$. Let us take the potential function p obtained this way and make the following *potential transformation* to the whole system S:

- if $u \leq w + \delta \cdot v$ is a constraint, change it to $u \leq w' + \delta \cdot v$, where $w' = w - p(u) + \delta \cdot p(v)$.

Since the strategy λ is stable, after the potential transformation, for every constraint $x \leq w' + \delta \cdot y$ that is *not* in λ one has $w' \leq 0$, with $w = 0$ for constraints in λ. This follows from the fact that stability means choosing a constraint with the maximal right-hand side (no attractive switches). Define an *optimal counterstrategy* $\tau(\lambda)$ of the controller's opponent against λ as a selection, for each variable, of one of the compulsory constraints that are *tight* in the optimal solution for $S(\lambda)$ (i.e., satisfied as equalities). This means that in a strategy $\tau(\lambda)$ in system $S(\lambda)$ there are also *no attractive switches* for the opponent (defined

symmetrically as for the controller). It follows that after the potential transformation for each constraint of the controller's opponent $y \leq w' + \delta \cdot x$ one has $w' \geq 0$, with $w' = 0$ for constraints in $\tau(\lambda)$.

Suppose the controller and his opponent, starting in a variable v_0, play a version of GENERALIZED GEOGRAPHY constructing a (finite or infinite) sequence of variables $v_0, v_1, v_2, \ldots$ as follows. The owner of v_i selects a constraint $v_i \leq w_{i,i+1} + \delta \cdot v_{i+1}$; this determines the next variable in the sequence, or the sequence terminates, if the constraint $v_i \leq 0$ is chosen.

The cost $c(v_0, v_1, \ldots)$ of each such finite or infinite sequence is determined as the sum $\sum_i \delta^i w_{i,i+1}$. Note immediately (by telescoping) that the potential transformation $p(.)$ changes the costs to:

1. $c'(v_0, \ldots, v_{l+1}) = c(v_0, \ldots, v_{l+1}) - p(v_0) + \delta^l \cdot p(v_{l+1})$, for a finite sequence, and
2. $c'(v_0, v_1 \ldots) = c(v_0, v_1, \ldots) - p(v_0)$, for an infinite one.

Therefore, the costs of paths after a potential transformation of S shift by a constant, in particular, when the controller and the opponent follow the pair of positional strategies λ and $\tau(\lambda)$ the cost of the path π they form equals zero. Therefore, the cost of the same path before the transformation is $p(v_0) - p(v_{l+1})$ or $p(v_0)$, depending on whether it is finite or infinite.

Suppose now, toward the contradiction that the strategy λ is not optimal, and there is a better strategy λ' providing for a larger value of the target function, consequently, for some variable v_0. Let the controller use this strategy, but the opponent sticks to the previous strategy $\tau(\lambda)$. Then the cost of the path π' they construct in the transformed S consists of the zero-weight edges consistently chosen by the opponent plus nonpositive edge weights chosen according to λ'. Therefore, the cost of this path π' is nonpositive, i.e., no better than of π. It follows, that the same relation holds for the cost of paths π and π' in the system S before the transformation, i.e., π' is no better for the controller. Therefore, λ' is no better than λ. This contradiction shows that λ is optimal.

It remains to show that for any feasible $S(\lambda)$ its optimal solution is finite. Indeed, no matter which constraints the parties select, the cost of a finite or infinite path followed is bounded by $W \cdot \sum_i \delta^i \leq W/(1-\delta)$, where W is the biggest constant w occurring in the constraints S. □

7.2 Multivariate Discounted Case

Consider the *multivariate discounted case*, when constraints are allowed to have form either $u \leq c$ (c a constant), or:

$$u \;\leq\; \sum_i a_i \cdot v_i, \tag{9}$$

where u and v_i's are variables, and a_i's are positive constants such that $\sum_i a_i < 1$.

Note that we can equivalently allow for only one type of constraints, namely:

$$u \;\leq\; \sum_i a_i \cdot v_i + w, \tag{10}$$

because such a constraint is expressible via $u \leq \sum a_i v_i + a'v'$ and $v' \leq w/a'$, where $a' > 0$ is small enough.

Theorem 7.2 *Let S be an instance of the multivariate discounted* Positive Controlled LP-Problem. *Then every discrete/pure stable strategy λ is optimal.*

Proof. Suppose λ is a stable discrete strategy, the system $S(\lambda)$ is feasible, and let $(\mathbf{x}^*, \mathbf{y}^*)$ be an optimal solution to $S(\lambda)$. We will shortly prove that this optimal solution is *finite*.

Generalizing the previous definition, a function p mapping variables in $\mathbf{x}$ and $\mathbf{y}$ to $\mathbb{R}$ is called a *potential function* for the system $S(\lambda)$ if for every constraint $u \leq \sum_i a_i v_i$ ($y \leq c$, resp.) one has $p(u) \leq \sum_i a_i p(v)$ ($p(y) \leq c$, resp.).

It is easy to see that the components of a finite optimal solution to $S(\lambda)$ determine potentials for the system $S(\lambda)$. Let us take the potential function p obtained this way and make the *potential transformation* defined below to the whole system S (rather than $S(\lambda)$).

To define an appropriate potential transformation we first need to introduce auxiliary terminology. For our purposes, a directed *hypergraph* consists of vertices (one per variable in S) and directed hyperedges of the form $(u, a_1 v_1, \ldots, a_k v_k, w)$ where u is the source, $v_1, \ldots, v_k$ are weighted targets, and $w \in \mathbb{R}$ is the hyperedge weight. A strategy for the controller (and the opponent) is a selection of a exactly one hyperedge corresponding to the associated constraint.

Now the desired potential transformation is defined by changing the hyperedge $(u, a_1 v_1, \ldots, a_k v_k, w)$ weights as follows

$$w' \;\; = w - p(u) + \textstyle\sum_i a_i p(v_i) \tag{11}$$

(Note that $y \leq c$ also forms a hyperedge with source y, no targets, of weight c, which is transformed to $c' = c - p(y)$.)

Let hyperedge $(u, a_1 v_1, \ldots, a_k v_k, w)$ be selected in λ, and $(u, a'_1 v'_1, \ldots, a'_k v'_k, \bar{w})$ be any other hyperedge from u. Then, by definition of $p(.)$ and since λ is stable, we have

$$p(u) = \sum_i a_i p(v_i) + w \geq \sum_i a'_i p(v'_i) + \bar{w}.$$

As a consequence, all hyperedges in λ get weight exactly 0, and all other controller hyperedges get nonpositive weights.

Define an optimal counterstrategy $\tau(\lambda)$ against λ for the opponent as a selection, for every variable, of a hyperedge corresponding to any of his tight constraints (holding as equalities) in the optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ for $S(\lambda)$. For each such hyperedge $(u, a_1 v_1, \ldots, a_k v_k, w)$ and any other hyperedge λ, and $(u, a'_1 v'_1, \ldots, a'_k v'_k, \bar{w})$ from u, we have

$$p(u) = \sum_i a_i p(v_i) + w \leq \sum_i a'_i p(v'_i) + \bar{w}.$$

Therefore, the potential transformation makes weights of hyperedges in $\tau(\lambda)$ exactly zero, and all other opponent hyperedges get nonnegative weights.

Now suppose, toward a contradiction, that λ is not optimal, and there is a better strategy λ' for the controller, providing for a larger value of the target function, hence, a larger value for some variable v_0.

When the controller and his opponent, starting from v play a Tree-Like Generalized Geography, then the cost of such a play is defined as a sum of weights of all hyperedges traversed. In particular, when they follow the pair of strategies λ, $\tau(\lambda)$, the total sum of all hyperedges is zero, because only 0-weight hyperedges are used by both parties. Suppose, however, that the controller deviates from λ to the allegedly better λ', but the opponent consistently sticks to $\tau(\lambda)$. Then the cost of the game cannot become positive, since the controller does not have positive-weight hyperedges, and the opponent only plays 0-weight hyperedges. Therefore, λ' cannot provide an improvement (larger value) compared to what λ does. This contradiction shows that λ is optimal. To complete the proof it remains to notice that:

1. The cost of the same play π in a potentially transformed and non-transformed systems differ only by a constant (by telescoping).

2. The cost of any play is bounded.

These facts follow from the assumption that $\max \sum_i a_i = \delta < 1$ and therefore the total cost of a play cannot exceed $C \cdot \sum_i \delta^i = C/(1-\delta)$. □

7.3 Univariate Non-Discounted Case

Let us consider the case of "one variable with coefficient one" in the right-hand sides, i.e., all the constraints have the form:

$$u \;\leq\; w + v, \tag{12}$$

where u, v are variables, and $w \in \mathbb{Z}$. Besides, we allow one (or more) compulsory constraint of the form

$$t \;\leq\; 0. \tag{13}$$

Due to our proviso that every variable appears in the left-hand-side of some constraint, the variables in (13) are the only possible sinks in a directed graph built on variables with directed edges corresponding to inequalities.

For reasons, which become clear later, we need to stipulate another important

Condition. In the directed graph determined by the constraints there are no *zero-weight cycles.* A (simple) cycle is a sequence of edges $u \xrightarrow{w} v$ corresponding to constraints $u \leq w+v$. As usual, the weight of this cycle is the sum of edge weights. □

Imposing this condition does not lead to a loss of generality. Indeed, suppose there are zero-weight loops in a graph or, we do not know[3]. We will describe the edge weight transformation $\Delta^{+1}_{\times n}$ that transforms all zero-weight loops into positive-weight ones[4], and preserves the signs of weights of all other loops (positive are turned into positive and negative to negative). The desired transformation is as follows. Multiply all edge weights by n and add one, where n is the number of vertices (variables). Every negative weight loop in the original graph remains negative, because its weight is the transformed graph is $\leq -1 \cdot n + 1 \cdot (n-1) = -1$. All non-negative loops obviously become positive. Now, if an optimal loop in the system before the transformation has weight 0 (the target function has the $+\infty$ value in this case) then the same loop will be "slightly" positive and optimal in the $\Delta^{+1}_{\times n}$-transformed system, and also results in an unbounded target value.

Theorem 7.3 *Let λ be a discrete stable strategy, the graph determined by S does not have zero-weight cycles, $S(\lambda)$ is feasible and has a finite[5] optimal solution. Then λ is optimal.*

Proof. Suppose, λ is a stable discrete strategy. Consider the potential p defined on variables by giving them the respective values of an optimal solution. For every edge/constraint $u \leq w + v$ the potentials satisfy $p(u) \leq w + p(v)$, with equalities for the tight constraints (e.g., for those selected in λ.) Define an optimal counterstrategy $\tau(\lambda)$ for the opponent as any selection of exactly one tight constraint per his variable. Make a potential transformation of the graph of S, by changing the weight of each edge according to $w'_{u,v} = w_{uv} - p(u) + p(v)$. After this transformation: 1) all edges in λ become 0-weight, and all other controller edges become non-positive (by stability); 2) all edges in $\tau(\lambda)$ become 0-weighted, and all other edges of the opponent become non-negative; 3) weights of all loops remain *unchanged*; 4) weights of finite paths $v_0, \ldots, v_l$ change by a constant $-p(v_0) + p(v_l)$ (telescoping), hence the relation $(<, =, >)$ between costs of two paths to a sink remains unchanged.

Let us assume, toward a contradiction, that λ is not optimal and there exists a better strategy λ', providing for a larger value of the target function, hence of some variable v_0. Suppose, the controller switches to this strategy λ', but the opponent persistently uses the strategy $\tau(\lambda)$. Now in the trace created from v_0 uniquely determined by the pair of strategies λ' and $\tau(\lambda)$ every edge is *nonpositive* (we assume the potential transformation has been performed). This trace may be:

1. either *finite*, terminating in a sink (13); in this case the value for v_0 is non-positive, and this contradicts to the assumption that λ' provides a better value for v_0 than λ;

2. or *infinite*; in this case the value for v_0 must be *negative*; indeed, due to our construction, the graph does not contain zero-weight cycles, and this property is preserved

[3]Note that our definition of a zero-weight loop assumes that the controller and the opponent cooperate in constructing such loops, which they actually may not want.

[4]For other applications in Section 10.3 we analogously turn, by $\Delta^{-1}_{\times n}$ (which subtracts rather than adds 1), zero-weight loops to negative-weighted and preserve weight signs of all other loops.

[5]This assumption is no loss of generality; see the end of Section 3.1.

during the potential transformation; since only non-positive edges are used by λ' and $\tau(\lambda)$, the only possible cycles formed by λ' and $\tau(\lambda)$ are negative; hence v_0 will be a negative value (or $-\infty$) in this case.

This contradiction shows that λ is indeed an optimal strategy. □

The proof shows that the Condition is essential. Indeed, if there are zero-weight loops, there may be stable but non-optimal strategies. Consider the following simple example.

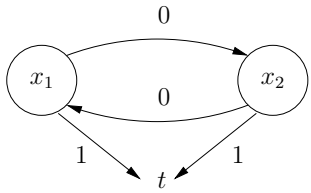

Figure 2: In this controlled LP there is a strategy that is stable, but not optimal. Round vertices are controlled.

The corresponding system of constraints is:

$$\begin{cases} x_1 \leq 1+t, \\ x_1 \leq x_2, \end{cases} \qquad \begin{cases} x_2 \leq 1+t, \\ x_2 \leq x_1, \end{cases} \qquad t \leq 0.$$

The strategy choosing the first constraints for each of x_1, x_2 is stable, and gives finite values 1,1 for both. However, a better strategy consists in selecting the second constraints, which leads to an unbounded solution, $x_1 = x_2 =$ anything. Indeed, by deviating from the first stable strategy the controller creates a zero loop, which allows for unboundedness.

7.4 NP∩coNP-Membership

In all the above cases with the property that stability implies optimality, the corresponding decision problem for the Positive Controlled LP-Problem is in NP∩coNP. For a YES-instance, just guess a pure strategy λ and verify that the value of the optimal solution in the resulting LP $S(\lambda)$ is above the threshold.

For a NO-instance, guess a strategy, verify in polynomial time that it is stable and thus optimal and confirm that the corresponding LP has a value below the threshold.

Hence, the univariate discounted, multivariate discounted, and univariate non-discounted cases are all in NP∩coNP.

7.5 The General Positive Controlled LP Case

As we have shown, in many important special cases of Positive Controlled LP-Problem stability implies optimality. This has very nice algorithmic and complexity-theoretic consequences. For the general case however, stability does not imply optimality. We show this

by giving a counterexample. Compared to the example in Section 7.3, the one presented here demonstrates that the unboundedness is not essential and we cannot even attain finite optimal values by stable strategies.

Consider the following controlled linear program:

$$\text{Maximize } v_1 + v_2 + v_3 \text{ subject to}$$

$$\begin{array}{l}\text{Compulsory constraints:}\\ \begin{cases} v_3 \le v_2 \\ v_3 \le 2 \end{cases}\\ \text{Optional constraints:}\\ \begin{cases} v_1 \le v_3 \\ v_1 \le 1 \end{cases}\\ \begin{cases} v_2 \le v_1 \\ v_2 \le 1 \end{cases}\end{array}$$

The corresponding graph is shown in Figure 3.

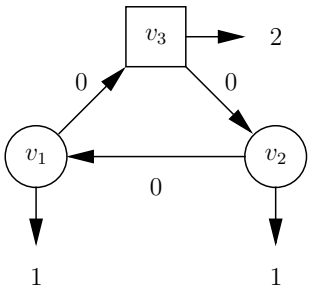

Figure 3: In this controlled LP there is a strategy that is stable but not optimal. Round vertices are controlled.

If the controller selects the constraints $v_1 \le 1$ and $v_2 \le 1$, the strategy is stable and the value is 3 (value 1 for each variable). However, if he had selected $v_1 \le v_3$ and $v_2 \le v_1$, we would have had the linear program:

$$\text{Maximize } v_1 + v_2 + v_3 \text{ subject to}$$

$$\begin{array}{l}\begin{cases} v_3 \le v_2 \\ v_3 \le 2 \end{cases}\\ v_1 \le v_3\\ v_2 \le v_1,\end{array}$$

which evaluates to 6 (value two for each variable).

Notice that this example does not contain any hyperedges, i.e., constraints with more than one variable on the right-hand side.

8 NP∩coNP Case: Positive Integer Coefficients

In this section we investigate the controlled LP problem, restricted to have only *nonnegative integers* as coefficients in the constraints, and prove the following result.

Theorem 8.1 *The controlled LP problem with nonnegative integer coefficients belongs to the complexity class* NP∩CO-NP.

For this case, stability does not imply optimality (Section 7.5). Therefore, more advanced reasoning is needed, compared with Section 7.4. Throughout this section we assume that the systems have retreats for both players. This guarantees that there are feasible solutions and that the optimal values are finite. We begin by formally defining cyclic dependencies.

Definition 8.2 *A system S with only one constraint per uncontrolled variable has a* cyclic constraint *over variable v if there is a controller strategy σ such that in $S(\sigma)$ there is a cyclic sequence of controlled variables $v = v_1, v_2, \ldots, v_k = v$ such that for each $i \in \{1, 2, \ldots, k-1\}$, there is a constraint of the form $v_i \leq poly(\overline{y}, v_{i+1}) + w_i$.*[6] *If there is a strategy with at least two* distinct *such sequences, then S has* multiple cyclic constraints *over v.* □

In the sequel we will need the following technical result.

Lemma 8.3 *Suppose that system S has only one constraint for each variable. If S has a cyclic constraint over some variable, then S is either infeasible or unbounded.*

Proof. If S is feasible and has a cyclic constraint over x, then, by combining constraints, we get $x \leq c \cdot x + poly(\mathbf{x} \setminus \{x\}, \mathbf{y})$. For any finite solution to S, the polynomial $poly(\mathbf{x} \setminus \{x\}, \mathbf{y})$ evaluates to some finite number. Since c is a positive integer constant, the value of x can be increased arbitrarily without violating the constraints. □

The next lemma follows from the previous one and our assumptions.

Lemma 8.4 *Given a problem instance S, let σ be an optimal strategy for* MAX. *Then there is a selection τ of exactly one constraint for each non-controlled variable from the tight constraints in $S(\sigma)$, such that in $S(\sigma, \tau)$, the system with only the constraints from σ and τ, there are no cyclic constraints.*

Proof. Let $(\mathbf{x}^*, \mathbf{y}^*)$ be an optimal solution to $S(\sigma)$. Since there are retreats and σ is optimal, $S(\sigma)$ has a feasible solution and $(\mathbf{x}^*, \mathbf{y}^*)$ assigns every variable a finite value.

If, for every choice of τ from the tight compulsory constraints, there exists a cyclic dependency, then by Lemma 8.3 the system is either infeasible or unbounded. Either way, we have a contradiction. □

[6] Since the opponent has no choices, each of his constraints $y \leq Q$ can be substituted in the right-hand sides of the controller's constraints and all non-controlled variables can be eliminated. For notational simplicity we assume here that this transformation is already done.

We call a selection τ as in the above lemma an *optimal counterstrategy* for MIN.

Lemma 8.5 *Let S be a system with only one constraint per uncontrolled variable, without multiple cyclic constraints and without cyclic constraints with coefficients larger than 1. Let σ be stable in S. Suppose there is a cyclic constraint with sequence $x_0, x_1, \ldots, x_k = x_0$ that* MAX *can enforce. Further, assume that by enforcing it, switching from σ* only *in variables in $\{x_1, \ldots, x_k\}$,* MAX *can get unbounded values. Call the strategy after switching σ'. If we do potential transformation on S, using the value of each variable under σ as the potentials, then every constraint for a variable $x_i, i \in \{1, \ldots, k\}$ in σ' is homogeneous.*

Proof. For each x_i, let c_i be the constraint for x_i in $S(\sigma')$. Also, let w_i be the constant term in c_i and q_i the value of the rest of the right-hand side of c_i, except x_{i+1}, under σ. First note that no variable except x_{i+1} in any constraint c_i in the cycle can depend on any variable in $\{x_1, \ldots, x_k\}$. If this were the case, there would be multiple cyclic constraints. Suppose that $x_{i_1}, x_{i_2}, \ldots, x_{i_r}$ are the variables for which the constraints differ between σ and σ'. For each variable v, let $p(v)$ be its value under σ. For every $j \in \{1, \ldots, r\}$ we have $p(x_{i_j}) \geq w_{i_j} + q_{i_j} + w_{i_j+1} + q_{i_j+1} + \cdots + p(x_{i_{j+1}})$, since σ is stable. By combining such constraints we get $\sum_{i \in \{1,\ldots,k\}} w_i + q_i \leq 0$. Since σ' gives unboundedness, the sum must actually equal 0.

By combining constraints in a similar way we see that for every $j \in \{1, \ldots, r\}$, the value $p(x_{i_j})$ equals $w_{i_j} + q_{i_j} + w_{i_j+1} + q_{i_j+1} + \cdots + p(x_{i_{j+1}})$. The new constant term in the constraint c_{i_j} is $w_{i_j} - p(x_{i_j}) + q_{i_j} + w_{i_j+1} + q_{i_j+1} + \cdots + p(x_{i_{j+1}}) = p(x_{i_j}) - p(x_{i_j}) = 0$. For every other constraint in $\{c_i | 1 \leq i \leq k\}$ the claim follows because they were tight under σ. □

Lemma 8.6 *Assume the same situation as in Lemma 8.5, but switching to σ', enforcing the cyclic constraint,* MAX *creates an infeasible system. Then, after the same potential transformation, at least one constraint for a variable $x_i, i \in \{1, \ldots, k\}$ in σ' has a negative constant term.*

Proof. Follows by an argument similar to that in the proof of Lemma 8.5. □

Together, Lemmas 8.5 and 8.6 show that any cyclic constraint such that MAX can achieve unbounded values by only switching to the constraints associated with the cyclic constraint gets homogeneous constraints after potential transformation with respect to a stable strategy. If there is such a cyclic constraint, it is easy to detect.

Lemma 8.7 *Let S be a system with only one constraint per uncontrolled variable. Let σ and σ' be two strategies such that neither $S(\sigma)$, nor $S(\sigma')$ has any cyclic constraints. Assume that σ is stable. Then σ' is no better than σ.*

Proof. Since $S(\sigma')$ has no cyclic constraints, the dependence of the variables on each other can be described by a DAG G, where the sinks are the variables with constraints of the form $y \leq w$, for a constant w, in $S(\sigma')$. For each variable v, let $p(v)$ and $p'(v)$ be the value of

v under σ and σ', respectively. Let the sequence T of variables be a topological sorting of G. We prove by induction on the number of later elements in T that $p'(v) \leq p(v)$ for every variable v. If v is last in T, then v is a sink in G and has a constraint of the form $v \leq w$, for constant w, in $S(\sigma')$. Since σ is stable we have $w = p'(v) \leq p(v)$.

Assume that for all elements u after v in T, we have $p'(u) \leq p(u)$. Let the constraint for u in $S(\sigma')$ be $v \leq c_1 \cdot x_1 + \ldots c_k \cdot x_k + w$. Then $p'(v) = c_1 \cdot p'(x_1) + \ldots c_k \cdot p'(x_k) + w$. Since σ is stable we have $p(v) \geq c_1 \cdot p(x_1) + \ldots c_k \cdot p(x_k) + w$. By inductive assumption this is no less than $c_1 \cdot p'(x_1) + \ldots c_k \cdot p'(x_k) + w = p'(v)$. □

The following procedure can be used to verify that a system S does *not* have a solution with value at least k.

1. Guess an optimal strategy pair (σ, τ).
2. Check that σ is stable.
3. Check that $S(\tau)$ has a finite optimal solution, and thus that σ is optimal.

If these steps can all be computed in polynomial time (as we actually show below), then the problem is in CO-NP.

Verifying that a strategy is stable is easy, so the question is whether we can determine if a system with only one constraint per uncontrolled variable allows MAX to create unbounded values.

The first thing we can check is whether there is a strategy σ' such that $S(\sigma', \tau)$ has a cyclic constraint over variables $x_1, \ldots, x_k$ such that in some constraint $x_i \leq poly(\overline{y}, x_{i+1})$, the coefficient in front of x_{i+1} is larger than 1. This can be done in polynomial time by search on a graph representation of $S(\tau)$.

The next step is to check whether there is a strategy σ', such that $S(\sigma', \tau)$, has a multiple cyclic constraint over some variable. This can be done in the following way. Consider the system $S(\tau)$. For each pair (y, z) of variables appearing together in some constraint, say one for x, use graph reachability to check whether x can be reached from both y and z in the graph representation of $S(\tau)$. The running time for this step is polynomial.

Once we know that there are no cyclic constraints with coefficients larger than 1 and no multiple cyclic constraints over any variable, it remains to check whether there is some single cyclic constraint with only unit coefficients that allows MAX to get an unbounded value. After potential transformation with respect to σ, we search for a cyclic constraint with only homogeneous constraints. If it exists, the system is unbounded by Lemma 8.6, and the strategy τ we guessed was incorrect. If, on the other hand, there is no such cyclic constraint, then we claim that σ is optimal. Indeed, if there are no cyclic constraints at all, the claim follows from Lemma 8.7. Otherwise, we can pick a cyclic constraint, say over $x_1, x_2, \ldots, x_k$, and remove all constraints for the variables x_i, and variables who depend on them. This does not affect the other variables. By applying the argument recursively to this smaller system S', we know that σ restricted to S' is optimal. This means that no strategy

which enforces the cyclic constraint over $x_1, \ldots, x_k$ can give a feasible system. The same must, for the same reasons, hold for any other cyclic constraint. Now by Lemma 8.5, σ is optimal in S. Thus we have short witnesses for NO-instances, and Theorem 8.1 follows.

Corollary 8.8 *The controlled LP problem with all coefficients equal to 0 or larger than 1 belongs to the complexity class* NP∩CO-NP.

9 The General Controlled LP Problem is NP-Hard

We now prove that allowing one negative coefficient in the right-hand side polynomials makes the CONTROLLED LP-PROBLEM NP-hard.

Definition 9.1 *The* GENERAL CONTROLLED LP-PROBLEM *is defined as in Definition 2.1, with the difference that we allow negative coefficients in the polynomials $p_i^j(\mathbf{y})$ and $q_k^l(\mathbf{x})$.*

Theorem 9.2 *The* GENERAL CONTROLLED LP-PROBLEM *is NP-hard.*

Proof. We prove NP-hardness by reducing from 3-SAT. Given an instance ϕ of 3-SAT with the variables $(x'_1, ..., x'_n)$ and the clauses $(C'_1, ..., C'_m)$, construct a controlled linear program with the variables $(x_1, ..., x_n)$, $(\bar{x}_1, ..., \bar{x}_n)$, and $(C_1, ..., C_m)$, where x_i represents x'_i, $\bar{x}_i$ represents $\neg x'_i$ and C_j represents the truth value of C'_j. For every variable x'_i in ϕ construct the following compulsory constraints:

$$\begin{aligned} x_i &\leq 1, \\ \bar{x}_i &\leq 1 - x_i. \end{aligned}$$

For every clause $C'_j = l_1 \vee l_2 \vee l_3$ in ϕ add the following optional constraint:

$$\left\{ \begin{array}{lcl} C_j & \leq & l_1, \\ C_j & \leq & l_2, \\ C_j & \leq & l_3. \end{array} \right.$$

The objective function to be maximized is $\sum_{i=1}^{n}(x_i + \bar{x}_i) + \sum_{j=1}^{m} C_j$. It is easy to see that every optimal solution will have $x_i + \bar{x}_i = 1$. Thus $\sum_{i=1}^{n}(x_i + \bar{x}_i) + \sum_{j=1}^{m} C_j = n + \sum_{j=1}^{m} C_j$.

Assume the optimal solution to the LP is $n + m$. Then $n + \sum_{j=1}^{m} C_j = n + m$ implies $\sum_{j=1}^{m} C_j = m$, and we know that $\forall i : (x_i \leq 1 \wedge \bar{x}_i \leq 1)$, which implies $\forall j : C_j \leq 1$. Then if $\sum_{j=1}^{m} C_j = m$, there must exist a variable in every clause that is equal to 1, i.e., ϕ is satisfiable.

Assume ϕ is satisfiable. There must exist a variable assignment that assures that every clause is satisfied, i.e., $\forall j : C_j = 1$, which implies $\sum_{j=1}^{m} C_j = m$. Thus the optimal solution to the LP program is $n + m$. □

10 Controlled LP Applications

10.1 Simple Stochastic Games

In this section we show how controlled LPs express *simple stochastic games* (SSG), a subclass of stochastic games introduced and studied by Shapley [21] (see [7, 8, 10] for a thorough treatment). A simple stochastic game is given by a directed graph $G = (V \cup \{0, 1\}, E)$, where the distinguished vertices 0 and 1 are called the 0-*sink* and the 1-*sink*, and the vertex set V is partitioned into subsets V_1, V_0, V_{avg} of the maximizing player 1, minimizing player 0, and probabilistic ones. Starting in the initial vertex players move a pebble selecting outgoing edges from their vertices. When the pebble comes to a probabilistic vertex, one of the outgoing edges is randomly selected, according to a specified probability distribution. The play stops when the pebble comes to a sink. Player 1 wants to maximize and player 0 to minimize the probability of getting to the 1-sink. SSGs turn out to possess a *value* ν and are solvable in positional strategies, optimal for both players, securing them the equilibrium probability ν of getting to the 1-sink.

The natural decision problem consists in determining, given an SSG and a probability $p \in [0, 1]$, whether the value of the game is $\geq p$. Condon [7] showed that this problem is in NP∩coNP, and suggested several algorithms for finding values and optimal strategies.

Actually, the first step Condon [7] applies is polynomial reduction[7] of an arbitrary SSG to a stopping SSG with a negligibly different value. Say that an SSG stops with probability 1 if no matter how players select moves, the game comes to a sink with probability 1. Roughly (see [7, 22] for details), every edge (u, v) with assigned probability p in the original game[8] transforms into two edges: (a) one going to the 0-sink, with probability pq, and (b) the other, with probability $p(1-q)$, from u to v, where q is appropriately selected exponentially small probability.

Suppose an SSG has undergone such a transformation. After throwing away all edges of type (a) the corresponding instance of the controlled LP problem is easy to construct. Write optional constraints $x \leq p \cdot y$ for every edge (x, y) from V_1, and compulsory constraints $y \leq p \cdot x$ for every edge from V_0, where $p < 1$ is the probability assigned to the edge in the transformation above. If (u, v_i) are edges from a vertex in V_{avg} with assigned probabilities p_i, $i \in I$, then write a compulsory constraint $u \leq \sum_{i \in I} p_i \cdot v_i$ (clearly, $\sum_{i \in I} p_i < 1$ by construction). Add compulsory constraints $v_0 \leq 0$ and $v_1 \leq 1$ for sinks.

Clearly, the resulting controlled LP is just a particular case of the discounted multivariate case considered in Section 7.2[9]. An additional advantage is that we do not need to transform the game into a binary one, which may increase the number of vertices quadratically, and turn a subexponential (in the number of variables) algorithm into an exponential one.[10]

[7] Corrected in [22] to become polynomial.

[8] Edges of players 0 and 1 are given probability 1.

[9] Actually, Shapley in the original definition of SGs stipulated discountedness; see [12].

[10] One may argue that such a quadratic blow-up in the number of vertices may only happen when there are quadratically many edges, and therefore, Ludwig's algorithm [14] remains *subexponential in the length of input*. More precisely, let n and $m = \Theta(n^2)$ be the numbers of vertices and edges in the original game; hence

Since every polynomial defining a constraint, except for the 1-sink, is homogeneous, any choice of optional constraints will yield a feasible linear program. In fact, for any such LP, setting all variables to zero gives a feasible solution, so there is no problem with selecting an initial admissible strategy. As an additional benefit, we also have interior point algorithms available.

10.2 Discounted Payoff Games

In this section we show how to describe Discounted Payoff Games as univariate discounted Positive Controlled LP-Problem. Let the game $G = (V = V_{\text{Min}} \cup V_{\text{Max}}, E, w, \delta)$ be given; see [22] for definitions. The objective of the adversaries is to maximize/minimize

$$(1-\delta)\sum_{i=0}^{i=\infty} \delta^i \cdot w(e_i),$$

and it has been shown that the optimal value vector of a DPG is the unique solution to the set of equations [22]:

$$x_i = \begin{cases} \max_{(i,j)\in E}\{(1-\delta)w_{ij} + \delta x_j\} & \text{if} \quad i \in V_{\text{Max}}, \\ \min_{(i,j)\in E}\{(1-\delta)w_{ij} + \delta x_j\} & \text{if} \quad i \in V_{\text{Min}}. \end{cases}$$

It is straightforward to see that this corresponds to the following controlled LP:

Maximize $\sum_i v_i$ subject to

Compulsory constraints:
$v_i \le (1-\delta)(w_{ij} + \delta v_j)$, *for* $v_i \in V_{\text{Min}}$ *and* $(v_i, v_j) \in E$.

Optional constraints:
$v_i \le (1-\delta)(w_{ij} + \delta v_j)$, *for* $v_i \in V_{\text{Max}}$ *and* $(v_i, v_j) \in E$.

This controlled LP is exactly of the form discussed in Section 7.1, hence stability implies optimality. Any stable vector in the linear program will also satisfy the system of equations determining the optimal vector of a Discounted Payoff Game. Therefore, a DPG can be expressed as a univariate discounted Positive Controlled LP-Problem.

---

the size of the input graph is $|G| = \Theta(n^2)$. Reducing to the game of outdegree 2 results in $n' = \Theta(n^2)$ vertices, and Ludwig's algorithm solves such a game in $2^{O(\sqrt{n'})} = 2^{O(n)} = 2^{o(|G|)}$ time, subexponential in input size $|G|$. This argument is, however, faulty, because in the scenario described above *any* straightforward algorithm is subexponential in the length of input. Indeed, there may be at most $n^n = 2^{n \log n} = 2^{o|G|}$ strategies, hence *every* brute-force search for the best strategy in this space does the job in time subexponential in $|G|$. Therefore, a more adequate meaning of *subexponential* in this context should be given/estimated in the *number of vertices*, rather than the input size. In particular, in the above scenario Ludwig's algorithm should be considered *exponential* in n. In Section 11 we give precise bounds for our algorithms in terms of the numbers of vertices n and edges m, like $2^{O(\log m \cdot \sqrt{n/\log n})}$ and $2^{O(\sqrt{n \log(m/\sqrt{n})} + \log m)}$, *subexponential* in n whenever $m = \Theta(poly(n))$.

10.3 The Longest Shortest Paths

The Longest Shortest Paths Problem (LSP) [5] consists in finding, for a given weighted digraph with a sink and a set of controlled vertices, a selection of exactly one edge from each controlled vertex maximizing the lengths of the shortest paths from all vertices to the sink.

An instance I of the LSP problem reduced to two instances L^- and L^+ of the univariate non-discounted Positive Controlled LP-Problem. The need for two instances arises from the different interpretations of zero-weight loops in the LSP and controlled LP. Such loops give zero distance in the LSP, but yield unbounded values of the associated variables in the LP.

The two instances L^- and L^+ are obtained by first applying to I the transformations $\Delta^{-1}_{\times n}$ and $\Delta^{+1}_{\times n}$, respectively, from Section 7.3.

Afterward, to get L^- and L^+ write optional constraints

$$x \leq w_{(x,y)} + y$$

for each controlled vertex with successor y via edge (x, y) of weight $w_{(x,y)}$. Write compulsory constraints

$$y \leq w_{(y,x)} + x$$

for each uncontrolled vertex y. Add the constraint $t \leq 0$ for the sink and the "retreat" constraints $x \leq -M$ for each controlled vertex, where $M > 0$ is big enough.

In L^- every nonpositive loop from I becomes negative, and every positive loop remains positive. Starting with the admissible strategy "retreat everywhere", run any version of iterative improvement by attractive switches on L^-, throwing away unbounded variables and constraints, as described in Section 3.1. To find out the original value for finite paths, we divide by n and round up to the nearest integer.

After solving L^- we know that all vertices corresponding to unbounded variables are part of positive loops in the LSP instance I. All vertices corresponding to finite values bigger than zero have also chosen the optimal strategy and obtained optimal values. If the finite values are less then zero then it is possible that there exists a zero-weight loop in I, which was not found in L^-.

To find out whether this is the case, we solve the second instance L^+, and for every variable with a finite value less than zero in L^- we check whether its value in L^+ is unbounded or not. If it is, we know that such variable have value 0 in I.

All previous results and algorithms now apply to the LSP problem.

10.4 Mean Payoff Games

Mean Payoff Games are adversary two-player full information infinite games played on finite, directed, edge-weighted, leafless graphs. The objective of the players is to maximize/minimize, in the limit, the sum of the edge weights divided by the number of edges traversed. These games are interesting in their own right but are also motivated by the

fact that Parity Games (and thus model checking) reduce to them. For details see [17, 9, 11, 20, 6]. Many important decision and optimization problems are reducible [4] to the Longest Shortest Paths problem, discussed in Section 10.3. Therefore it follows directly that they can be expressed by the univariate non-discounted Positive Controlled LP-Problem.

11 Algorithms

11.1 Single and Multiple Switch Algorithms

In Section 3 we discussed a generic switching algorithm for the Positive Controlled LP-Problem. There are several switching schemes that can be employed in this template algorithm, and we describe some of them briefly here. For more detailed descriptions and applications of these schemes see [1].

- *Single Random Switch*: select uniformly at random exactly one out of the attractive switches in each iteration.
- *Single Greedy Switch*: select "greedily" exactly one most attractive switch (switch to the most relaxing constraint) in each iteration.
- *All Attractive Switches*: perform all attractive switches at every iteration.
- *Random Multiple Attractive Switches*: flip a coin for every attractive switch in each iteration (or uniformly at random choose a subset of all available attractive switches.)

11.2 Randomized Subexponential Algorithms

Khachiyan was the first to show that linear programming can be solved in polynomial time. His algorithm, as well as the one of Karmarkar, are polynomial in the size of the input, but the running times cannot be bounded solely in the numbers of constraints and variables. The question whether there is an algorithm that is (strongly) polynomial in the combinatorial model of computation remains open, and research in this area continues. In a major breakthrough, Kalai showed in 1992 that the problem does at least have a strongly subexponential solution [13]. Matoušek, Sharir, and Welzl independently proved the same result for another algorithm [15]. In 1995, Ludwig was the first to propose using these techniques in a game-theoretic setting [14], but in a restricted form, only for games on graphs of outdegree 2.

11.2.1 Kalai-Style Randomization for Controlled LP

Kalai's approach can be successfully adapted for solving Positive Controlled LP-Problem in subexponential time (in the number of variables and constraints).

Given an instance S of PCLP, a *subproblem* F is obtained by fixing one optional constraint for some controlled variable. In combinatorial terms, F is a *facet* in the space of positional strategies of the controller in S. This establishes the "constraints $\leftrightarrow$ facets" relation.

The Kalai-style algorithm for the PCLP proceeds as follows (cf., [2, 3]). Given an instance S and a strategy σ, it applies recursively to find a big enough set $\mathcal{S}$ of *better* subproblems (facets) that have optimal strategies better than σ (with a larger value of the target function). This is accomplished by starting with the subproblem where all optional constraints,

except those in σ, are removed, and iteratively adding improving (attractive) constraints; each such constraint defines a better subproblem. (The algorithm may terminate if during this phase it finds an optimal strategy in S, or $+\infty$ results.) Sampling uniformly at random from $\mathcal{S}$, the algorithm selects a subproblem F, in which it recursively finds an optimal strategy σ'. All facets in $\mathcal{S}$ are ordered according to the values of their best strategies. Due to randomization, all positions for F in this ordering are equiprobable. The algorithm never again visits a facet worse than F. It follows that the *hidden dimension* (the unknown number of better subproblems remaining after σ' is obtained) decreases randomly. This results in a randomized algorithm with subexponential $2^{O(\log m \cdot \sqrt{n/\log n})}$ expected number of iterations for the POSITIVE CONTROLLED LP-PROBLEM (whenever stability implies optimality; otherwise we find a local optimum in subexponential time), where n and m are the numbers of controlled variables and optional constraints, respectively. If we assume that m is polynomially bounded in n, the bound is $2^{O(\sqrt{n \log n})}$.

11.2.2 Matoušek-Sharir-Welzl-Style Randomization for Controlled LP

Associating optional constraints with facets in the space of controller's positional strategies, provides the intuitions necessary for applying other approaches from combinatorial linear programming to the PCLP. In particular, we can adapt the algorithm by Matoušek, Sharir, and Welzl (MSW). This algorithm is more intuitive, and easier to explain for PCLP than Kalai's. The function *opt* $S(\sigma)$ works as the target function being optimized.

Algorithm 1: MSW-Style Algorithm for PCLP
PCLP-MSW(S, σ_0)
(1) **if** every controlled variable has only one optional constraint in S
(2) **return** σ_0
(3) choose a random optional constraint $x \le e$ not selected by σ_0
(4) $\sigma^* \leftarrow$ PCLP-MSW$(S \setminus \{x \le e\}, \sigma_0)$
(5) $\sigma_1 \leftarrow$ SWITCH$(\sigma^*, x \le e)$
(6) **if** *opt* $S(\sigma_1) >$ *opt* $S(\sigma^*)$
(7) $S' \leftarrow S \setminus \{x \le e' : e' \ne e\}$
(8) **return** PCLP-MSW(S', σ_1)
(9) **else return** σ^*

The SWITCH function takes a strategy σ and an optional constraint $x \le e$ and returns the strategy obtained by changing the choice of σ for x to $x \le e$. In combinatorial optimization terms, line 3 of the algorithm corresponds to selecting a facet in the space of positional strategies. Line 4 corresponds to recursive optimization on the rest of this space.

The expected number of iterations of this algorithm is $2^{O(\sqrt{n \log(m/\sqrt{n})+\log m})}$, where n and m are the numbers of controlled variables and optional constraints, respectively, similar to the bound for Kalai's algorithm. Again, if m is polynomial in n, the bound is $2^{O(\sqrt{n \log n})}$.

12 Controlled Max Bipartite Matching and Max Flow

Previous sections show that a number of problems that belong to P have controlled versions in NP∩co-NP, but also that the general controlled LP problem (where negative coefficients are allowed) is NP-complete. It actually seems that the latter case is more common, and here we give two additional examples.

In the Controlled Max Bipartite Matching problem, we are given a bipartite, undirected graph $G = (V = (A \cup B), E)$, a subset $C \subseteq V$ of controlled vertices, and a function $c : C \to \mathbb{N}$ that for each controlled vertex says how many incident edges the controller is allowed to remove.

The Controlled Max Bipartite Matching problem is NP-complete. An optimal strategy of the controller can serve as a short witness. We show hardness by reduction from Set Cover. Given an instance (X, F, k) of Set Cover, where X is a set, $F \subseteq \mathcal{P}(X)$, and $k \in \mathbb{N}$, the question is whether there is a subset $S \subseteq F$ of size at most k, such that every element in X is a member of some set from S. To solve it we produce an instance of Controlled Max Bipartite Matching as follows.

The two vertex sets are X and F. An edge $(u, v) \in X \times F$ belongs to the edge set E iff $u \in v$. For each vertex $u \in X$ the controller can remove all but one adjacent edge ($c(u) = deg(u) - 1$). Edges can be removed so that the maximum bipartite matching is $\leq k$ if and only if there was a set cover of size at most k in (X, F, k).

The more general, but still polynomial, Maximum Flow problem can be easily expressed by linear programs, but *not in the form of* positive constraint we use in PLCP. Skew symmetry and flow preservation constraints require negative coefficients. Actually, no matter which representation tricks one tries to use, negative coefficients are unavoidable. Indeed, Controlled Maximum Flow is also NP-hard.

In the Controlled Maximum Flow problem, a flow network is given, together with a subset C of the vertices of the network. There is also a function $c : C \to \mathbb{N}$. The problem is to remove $c(v)$ or fewer outgoing edges from every $v \in C$ in a way that minimizes the maximum flow in the remaining network. NP-hardness can be proved by reduction from CNF-Non-Validity; the problem of determining whether a given CNF formula is *not* a tautology. CNF-Non-Validity is NP-complete, since CNF-Validity is co-NP-complete.

13 Conclusions

We briefly enumerate several problems remaining to be investigated.

1. Interior point schemes should be investigated further, as per their potential advantage over discrete schemes. Although we know how to make interior point search subexponential, there are further problems. It is conceivable that the interior point algorithms can make exponentially many discretization steps before finding the optimal solution. We hope this is not the case in the restricted cases of the CONTROLLED LP-PROBLEM.

2. For many special cases, local search algorithms, similar to those for the related games, give optimal results. In the general case, local search does not give optimal solutions. In the case of integer coefficients, however, we could still prove NP∩coNP-membership. Can this problem also be solved in expected subexponential time?

3. Does the general positive controlled LP problem belong to coNP?

4. The positive controlled LP problem catches many important optimization problems, including several games. Are there any other interesting optimization problems of this kind?

5. Can the subexponential upper bound for the problem be improved to $2^{o(\sqrt{n \log n})}$, or even to a polynomial one?

References

[1] H. Björklund, S. Sandberg, and S. Vorobyov. Complexity of model checking by iterative improvement: the pseudo-Boolean framework. In M. Broy and A. Zamulin, editors, *Andrei Ershov Fifth International Conference "Perspectives of System Informatics"*, volume 2890 of *Lecture Notes in Computer Science*, pages 381–394, 2003.

[2] H. Björklund, S. Sandberg, and S. Vorobyov. A discrete subexponential algorithm for parity games. In H. Alt and M. Habib, editors, *20th International Symposium on Theoretical Aspects of Computer Science, STACS'2003*, volume 2607 of *Lecture Notes in Computer Science*, pages 663–674, Berlin, 2003. Springer-Verlag.

[3] H. Björklund, S. Sandberg, and S. Vorobyov. Randomized subexponential algorithms for parity games. Technical Report 2003-019, Department of Information Technology, Uppsala University, 2003. `http://www.it.uu.se/research/reports/`.

[4] H. Björklund, S. Sandberg, and S. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. Technical Report DIMACS-2004-05, DIMACS: Center for Discrete Mathematics and Theoretical Computer Science, Rutgers University, NJ, March 2004. `http://dimacs.rutgers.edu/TechnicalReports/`.

[5] H. Björklund, S. Sandberg, and S. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. In J. Fiala, V. Koubek, and J. Kratochvil, editors, *29th International Symposium on Mathematical Foundations of Computer Science (MFCS'2004)*, volume 3153 of *Lecture Notes in Computer Science*, pages 673–685. Springer, August 2004. Preliminary full version: TR DIMACS-2004-05 `http://dimacs.rutgers.edu/TechnicalReports/`.

[6] H. Björklund, S. Sandberg, and S. Vorobyov. Memoryless determinacy of parity and mean payoff games: A simple proof. *Theoretical Computer Science*, 310(1-3):365–378, 2004.

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

[8] A. Condon. On algorithms for simple stochastic games. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 13:51–71, 1993.

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

[10] J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer, 1996.

[11] V. A. Gurvich, A. V. Karzanov, and L. G. Khachiyan. Cyclic games and an algorithm to find minimax cycle means in directed graphs. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 28(5):85–91, 1988.

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

[13] G. Kalai. A subexponential randomized simplex algorithm. In *24th ACM STOC*, pages 475–482, 1992.

[14] W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and Computation*, 117:151–155, 1995.

[15] J. Matoušek, M. Sharir, and M. Welzl. A subexponential bound for linear programming. In *8th ACM Symp. on Computational Geometry*, pages 1–8, 1992.

[16] J. Matoušek, M. Sharir, and M. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16:498–516, 1996.

[17] H. Moulin. Extensions of two person zero sum games. *J. Math. Analysis and Applic.*, 55:490–508, 1976.

[18] V. Petersson and S. Vorobyov. Interior-point approach to parity games. In *IEEE Annual Symposium on Logic in Computer Science (LICS'2001)*, Boston, Massachusetts, June 2001. Short communication.

[19] V. Petersson and S. Vorobyov. Parity games: interior-point approach. Technical Report 008, Uppsala University / Information Technology, May 2001. `http://www.its.uu.se/research/reports/`.

[20] N. Pisaruk. Mean cost cyclical games. *Mathematics of Operations Research*, 24(4):817–828, 1999.

[21] L. S. Shapley. Stochastic games. *Proc. Nat. Acad. Sci. USA*, 39:1095–110, 1953.

[22] U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.*, 158:343–359, 1996.