Available online at www.sciencedirect.com

SCIENCE DIRECT®

Theoretical Computer Science 349 (2005) 347–360

Theoretical Computer Science

www.elsevier.com/locate/tcs

Combinatorial structure and randomized subexponential algorithms for infinite games[☆]

Henrik Björklund, Sergei Vorobyov[*]

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

Received 1 April 2004; received in revised form 27 April 2005; accepted 24 July 2005

Communicated by A.S. Fraenkel

Abstract

The complexity of solving infinite games, including parity, mean payoff, and simple stochastic, is an important open problem in verification, automata, and complexity theory. In this paper, we develop an abstract setting for studying and solving such games, based on function optimization over certain discrete structures. We introduce new classes of *recursively local-global* (RLG) and *partial recursively local-global* (PRLG) functions, and show that strategy evaluation functions for simple stochastic, mean payoff, and parity games belong to these classes.

In this setting, we suggest randomized subexponential algorithms appropriate for RLG- and PRLG-function optimization. We show that the subexponential algorithms for combinatorial linear programming, due to Kalai and Matoušek, Sharir, Welzl, can be adapted for optimizing the RLG- and PRLG-functions.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Mean payoff games; Simple stochastic games; Parity games; Pseudo-Boolean optimization; Completely unimodal functions; Combinatorial linear programming; Randomized algorithms

1. Introduction

The theory of infinite two-person adversary full information games is a well-established framework for modeling interaction between a system and its environment. A correct system can be interpreted as a player possessing a winning strategy against any strategy of the malevolent environment. In this setting, the verification process can be considered as a proof that the system possesses 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 in fitting diverse approaches to computer-aided verification into the game-theoretic paradigm and in developing efficient algorithms for solving games, i.e., determining the winner and the winning strategy [16,11,19,25,39,3]. A natural approach to solving games [23,13,33,39,3] consists in taking an initial strategy of the system and gradually “improving”

[☆] Supported by Swedish Research Council (VR) grants “*Infinite Games: Algorithms and Complexity*”, “*Interior-Point Methods for Infinite Games*”, “*Infinite Games for Program Verification: Combinatorial Optimization Approach*”, and by a grant from the Swedish Foundation for International Cooperation in Research and Higher Education (STINT).

[*] Corresponding author. Tel.: +46 18 471 10 55; fax: +46 18 55 02 25.
E-mail address: Sergei.Vorobyov@it.uu.se (S. Vorobyov).

0304-3975/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2005.07.041

it, since a nonoptimal strategy is outperformed by some counterstrategy of the environment. In this paper, we address the complexity of such approaches in an abstract model based on optimizing functions on sets of game strategies.

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*. One of the major 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 to solving real large-scale games emerging from practical verification of complex reactive systems [31].

Our primary motivation consists in creating a unified theory of abstract optimization appropriate for solving different classes of infinite games, including parity, mean payoff, and simple stochastic games (SSG). It turns out that despite differences, these games possess common structural properties that allow us to abstract away irrelevant features when studying iterative improvement algorithms. This leads to a simple, general, and powerful optimization theory. It resulted in the first strongly subexponential randomized algorithms for parity [3] and mean payoff games (MPGs) [9], as well as in a randomized subexponential algorithm for SSGs of arbitrary outdegree [5]. [1] Recent improvements and refinements of these results are reported in [7,36,9].

1. *Parity games* (PGs) are infinite games played on finite directed leafless graphs, with vertices colored by nonnegative integers. Two players move a pebble along edges of the graph forming an infinite path. Vertices are partitioned between the players, and the owner of the vertex currently visited by the pebble decides where to move it next. The goal of one player is to ensure that the largest color visited by the pebble infinitely often is *even*, whereas the other player tries to make it *odd*. Deciding the winner in PGs is equivalent to the Rabin chain tree automata nonemptiness, as well as to the μ-calculus model checking [17,16]. μ-calculus is one of the most expressive temporal logics of programs [16], expressively subsuming virtually all known such logics. The problem is therefore significant in verification, automata theory, and also from the complexity-theoretic point of view, since it belongs to the complexity class NP $\cap$ coNP, but is not known to belong to P.

2. *Mean payoff games* (MPGs) [15,20,43] are another important motivation for studying iterative improvement. They are similar to PGs, but instead of colored vertices have integer-weighted edges, and the players try to maximize, respectively minimize, the average edge weight in the limit. PGs reduce to MPGs, and similar to PGs, the decision problem "whether a vertex value is above a given threshold" for MPGs belongs to NP $\cap$ coNP. MPGs can be used to design and analyze algorithms for job scheduling, finite-window online string matching, and selection with limited storage [43].

3. *Simple stochastic games*: Stochastic games, an important generalization of Markov decision processes, relevant for the analysis and verification of adversary probabilistic systems, were introduced by Shapley [34] and many variants have been extensively studied since then [23,14,18]. In this paper, we restrict to a subclass of *simple* stochastic games (SSGs), in which players select moves in turns rather than simultaneously. SSGs were introduced and studied by Condon [12,13] (see also [28,43]), motivated by complexity-theoretic analysis of randomized space-bounded alternating Turing machines. Both PGs and MPGs polynomially reduce to SSGs and thus can be solved by any SSG algorithms. However, the direct MPG and PG decision algorithms we suggest in this paper provide for a better complexity [9]. The decision problem, whether one of the players in an SSG can secure a given probability of reaching the distinguished vertex, also belongs to NP $\cap$ coNP.

The basic concept underlying all algorithms discussed in this paper consists in assigning values to strategies of one of the players, and then searching the strategy space guided by these values and making iterative strategy improvements. This approach relies on the fundamental *positional determinacy* property. It turns out that all the games mentioned are determined and solvable in *pure positional* strategies. This was first proved for MPGs by Ehrenfeucht and Mycielski [15], and for PGs by Emerson and Jutla [17]. (Simple) stochastic games have the same property; see, e.g., [34,24,12,18]. Positional determinacy means that each vertex of every such game has a value, and that both players have pure (nonrandom) positional (memoryless) strategies that secure this value, whenever the game starts in the vertex; moreover, such strategies are uniform, independent of the initial vertex. Solving the game amounts to finding these values and strategies. We can thus limit our search to the *finite* set of all pure positional strategies.

A pure positional strategy of a player maps each of the player's vertices to one of its successors. Using the positional strategy the player always selects this successor, independently of the history of the play. The set of all positional

[1] The algorithm of [28] is only subexponential for SSGs of *bounded* outdegree.

strategies of a player is a Cartesian product of finite sets (of successors, for each vertex), and solving a game amounts to finding a global maximum of the strategy evaluation function on this product. In particular, if the player has at most two choices in each vertex, the space of all positional strategies is a Boolean hypercube. The resulting functions resemble those studied in the field of pseudo-Boolean optimization [10,22], in which local search [1,38] is one of the dominating methods. In general, pseudo-Boolean functions cannot be efficiently optimized [38], but for some subclasses, in particular, for the *completely unimodal* (CU) functions [21,41,40,38], the situation is more promising. They can be optimized in subexponential time using an adaptation of Ludwig's algorithm for SSGs [2], based on ideas of Kalai [26] and Matoušek et al. [35,30,29]. A long-standing conjecture [41] is that CU-functions can be optimized in polynomial time.

In this paper, we generalize pseudo-Boolean functions and develop the abstract setting of *hyperstructure* optimization. The purpose is to facilitate the development and study of different algorithms for games. We introduce and study the classes of *total and partial recursively local-global functions*, and show that they are adequate to cover the above-mentioned games. In this abstract setting, we show that the subexponential, randomized algorithms for linear programming due to Kalai [26] and Matoušek et al. [29,30] can be adapted to optimize a wide class of functions on hyperstructures, including the strategy evaluation functions for PGs, MPGs, and SSGs.

There is a well-known function for assigning values to strategies in SSGs [23,13,28], with properties close to the CU-functions [2,4,5]. There is also a reduction from PGs via MPGs to SSGs [33,43]. Thus, PGs and MPGs can be solved by reduction followed by optimization of the SSG value function. This approach has a downside, however. Each function evaluation involves solving a linear program with high precision. This is costly, both with regard to running time and development (typical off-the-shelf linear programming modules do not provide for the necessary precision). The randomized algorithm for SSGs by Ludwig [28], is subexponential in the number of vertices of the game, only if the outdegree of vertices is *at most two* (or bounded by a constant). Unfortunately, reduction of arbitrary outdegree games to binary ones may increase the number of vertices quadratically, making the algorithm exponential.

Until recently, no convenient (discrete) strategy evaluation function for MPGs was known. In this paper, we present a new discrete strategy evaluation for MPGs, first discovered in [9], and based on computing shortest paths. Together with the algorithms from combinatorial LP-type optimization, this shows that MPGs can be solved in strongly subexponential expected time, without the need for high-precision arithmetic. The evaluation measure also provides, by reduction, a new and efficient strategy improvement scheme for PGs.

Paper outline: After preliminaries on optimization over discrete structures, we proceed to SSGs, show that they define recursively local-global (RLG) strategy evaluation functions, which can be optimized in expected subexponential time. Then we address MPGs and partially RLG strategy evaluation functions they induce. In the end, we cover the case of PGs.

2. Optimization on discrete structures

2.1. Optimization on Boolean hypercubes

A function from the Boolean hypercube $\mathcal{H} = \{0, 1\}^d$ to $\mathbb{R}$ is called *pseudo-Boolean*. Optimizing such a function is a problem of finding the vertex of the hypercube with the maximal (or minimal) function value. This problem has been extensively studied; see, e.g., [21,40,37,41,38,10,2]. In general, it requires exponential time in the dimension d of $\mathcal{H}$ [38], and the research typically focuses on special subclasses of functions, such as the CU functions (injective, with unique maximum on every subcube) [21,40,41,2]. One of the dominating approaches to pseudo-Boolean optimization is local search. The *standard local search improvement algorithm* starts in an arbitrary vertex of the hypercube and iteratively improves by selecting a next iterate with a better function value from a *polynomial* neighborhood (usually the set of vertices at Hamming distance one) of the current vertex. Specific instances of the standard algorithm are obtained by fixing a neighborhood structure on $\mathcal{H}$, together with a discipline of selecting the initial vertex and the next iterate. The two major local search improvement algorithms, the *Best Neighbor Algorithm* (greedily selecting one of the best in the neighborhood) and the *Random Better Neighbor Algorithm*, have previously been extensively investigated and used for pseudo-Boolean optimization [38]. These algorithms perform well in practice and there is a long-standing conjecture that the Random Better Neighbor is polynomial for CU-functions [41]. However, the best currently known worst-case bounds for these algorithms are still *exponential*.

2.2. Hyperstructure optimization

SSGs and MPGs dealt with in this paper are played by two adversary players MAX and MIN on finite directed graphs. The players control two disjoint subsets of vertices, and when a token (pebble) comes to a vertex controlled by a player, that player moves it to some successor vertex by selecting an outgoing edge. In this way the players construct sequences of edges or plays. The rules of selecting edges are called *strategies*. A remarkable property of SSGs and MPGs is that optimal strategies are pure (nonrandom) and positional (memoryless), i.e., selection of edges may be done disregarding the history of the play (memoryless determinacy).

Combinatorially, the set of all pure positional strategies of player MAX in a game is isomorphic to a product of finite sets, each representing the player's choices in a vertex he controls. The number of vertices of MAX corresponds to the dimension. Evaluating strategies in games motivates our study of functions defined on the *hyperstructures* (products of simplices). Hyperstructures directly generalize Boolean hypercubes $\mathcal{H} = \{0, 1\}^d$, corresponding to the particular case when the player's choice in every vertex is binary. In Definition 2.1, it is convenient to identify $\mathcal{P}_j$ with the set of outgoing edges from vertex j, i.e., with the choices available to the player in this vertex.

Definition 2.1. Let $\{\mathcal{P}_j\}_{j=1}^d$ be a collection of finite nonempty disjoint sets. Call $\mathcal{P} = \prod_{j=1}^d \mathcal{P}_j$ a d-dimensional *hyperstructure* (or *structure* for short). An element of $\mathcal{P}$ is called a *vertex*.[2] Two vertices of $\mathcal{P}$ are *neighbors* if they differ in only one coordinate.

A *substructure* of $\mathcal{P}$ 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. A substructure $\mathcal{P}'$ of $\mathcal{P}$ is called a *facet* of $\mathcal{P}$ if there is a $j \in \{1, \ldots, d\}$ such that $|\mathcal{P}_j| > 1$, $|\mathcal{P}'_j| = 1$, and $\mathcal{P}'_k = \mathcal{P}_k$ for all $k \neq j$.

In terms of games and strategies, a *substructure* of the strategy hyperstructure $\mathcal{P}$ corresponds to the set of MAX strategies in a subgame in which some edges leaving vertices controlled by MAX have been removed, without creating sinks. Moving from a vertex of $\mathcal{P}$ to a neighbor is the same as making a single strategy switch in the underlying game.

Recall that a *local maximum* of a function $f : \mathcal{P} \to \mathcal{D}$ is a vertex v such that $f(v) \geqslant f(u)$ for all neighbors u of v. If $f(v) \geqslant f(u)$ for all vertices $u \in \mathcal{P}$, then v is a *global maximum* of f on $\mathcal{P}$. If $\mathcal{P}'$ is a substructure of $\mathcal{P}$ then $f|_{\mathcal{P}'}$ denotes the restriction of f to $\mathcal{P}'$.

Here comes one of our central concepts.

Definition 2.2 (*RLG-Functions*). A function $f : \mathcal{P} \to \mathbb{R}$ is called RLG if on every substructure of $\mathcal{P}$ every local maximum of f is global. In this case, the pair $(\mathcal{P}, f)$ is called an *RLG-structure*.

RLG-functions clearly generalize CU-functions. We will show that SSGs induce RLG-functions and those may be optimized in subexponential time. Note that RLG-functions are more general than Kalai's *abstract objective functions* (AOFs) [27], because we do not stipulate uniqueness of local maxima. An even further generalization is achieved by partial RLG (PRLG-) functions in Section 8.

3. Simple stochastic games

We summarize main notions concerning SSGs, cf., [12,43].

Definition 3.1. An SSG is played on a finite directed edge-weighted graph $G = (V, E, w)$, where $w : E \to (0, 1]$, and the set of vertices $V = \{0, 1, 2, \ldots, n-1\}$ is partitioned into sets V_{MAX}, V_{MIN}, V_{AVG}, and $\{0, 1\}$, called, respectively, MAX-controlled vertices, MIN-controlled vertices, average (random) vertices, 0-sink, and 1-sink. Every vertex, except sinks, has at least one outgoing edge; sinks have none. In addition, $\sum_{(u,v)\in E} w(u, v) \leqslant 1$ for each $u \in V_{\text{AVG}}$.[3] Some vertex is distinguished as initial.

[2] Corresponds to a positional strategy, not to be confused with game vertices!

[3] It is convenient (for discounted SSGs; see below) to have probabilities assigned to edges leaving vertices from $V_{\text{MAX}} \cup V_{\text{MIN}}$ as well. To conform to the standard definition, assign to every such edge probability 1.

Starting in the initial vertex players MAX and MIN move a pebble along edges according to the following rules:

- When the pebble comes to a vertex $u \in V_{\mathrm{MAX}} \cup V_{\mathrm{MIN}}$, the player who controls u selects an outgoing edge (u, v), and the pebble moves to:
  - vertex v with probability $w(u, v)$.
  - the 0-sink with probability $1 - w(u, v)$.
- From a vertex $u \in V_{\mathrm{AVG}}$ the pebble goes to:
  - one of the successors v of u with probability $w(u, v)$.
  - the 0-sink with probability $1 - \sum_{(u,v)\in E} w(u, v)$.

A play terminates in a sink. Player MAX/MIN tries to maximize/minimize the probability of reaching the 1-sink.

The value $\mathrm{val}(v)$ of a vertex $v \in V$ is the probability that a play starting in v reaches the 1-sink, if both players act optimally.

The SSG decision problem is whether MAX can secure a given probability of reaching the 1-sink starting from the initial vertex.

A pure positional strategy of a player is a selection of exactly one outgoing edge from every vertex the player controls. For a strategy σ of a player in G denoted by G_σ the game obtained by deleting all edges of the player in G, except those used in σ. A switch is a change of a single selected edge in a strategy.

Say that an SSG is *discounted* if both:

(1) the probability assigned to every edge leaving a vertex in $V_{\mathrm{MAX}} \cup V_{\mathrm{MIN}}$ and
(2) the sum of probabilities assigned to edges leaving a vertex in V_{AVG}

are strictly smaller than 1.

SSGs are known to be solvable in pure positional strategies [34,12].

Theorem 3.2. *In every SSG every vertex has a value* $\mathrm{val}(v)$, *and the players possess pure positional strategies guaranteeing a value* $\geqslant \mathrm{val}(v)$ *and* $\leqslant \mathrm{val}(v)$, *respectively, against any strategy of the opponent, when the play starts in* v.

Condon shows [12] (corrected in [43]) that every SSG, with an arbitrary vertex outdegree, can be polynomially transformed into a discounted SSG with a negligibly different value, allowing for precisely determining the value of the original game (the result due to Shapley [34], modulo polynomiality of the reduction). Thus, concentrating on discounted SSGs is no loss of generality. We need several details of the reduction from an arbitrary SSG G to a discounted SSG Γ; see [12,43]. During the transformation, every edge (u, v) with assigned probability p in the original game G (here it is useful that edges of players MAX and MIN in G are initially given probability 1) splits 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 an appropriately selected exponentially small probability, negligible for determining the true value of a vertex in G. After that the edges of type (a) are discarded and the resulting SSG Γ becomes discounted. Determining vertex values of discounted SSGs, for fixed MAX strategies, can be done by solving linear programs, as explained below.

Definition 3.3. For a discounted SSG Γ the associated system of linear constraints S_Γ contains the following constraints:

(1) $v_i \leqslant w(i, j)v_j$, for each $i \in V_{\mathrm{MAX}}$ and edge $(i, j) \in E$, called MAX-constraints;
(2) $v_i \leqslant w(i, j)v_j$, for each $i \in V_{\mathrm{MIN}}$ and edge $(i, j) \in E$, called MIN-constraints;
(3) $v_i = \sum_{k=1}^{n_i} w(i, j_k)v_{j_k}$, for each $i \in V_{\mathrm{AVG}}$ with n_i outgoing edges $(i, j_k) \in E$, $k = 1, \ldots, n_i$, and $\sum_{k=1}^{n_i} w(i, j_k) < 1$, called *average* constraints;
(4) $v_0 = 0$ and $v_1 = 1$, called *sink* constraints.

For a pure positional strategy σ of MAX in Γ denote by $S_\Gamma(\sigma)$ the subsystem of S_Γ obtained by deleting all MAX-constraints in S_Γ, except those corresponding to edges used in σ.

The following theorem is a modification, to discounted SSGs of arbitrary outdegree, of the results due to Shapley [34] and Derman [14]; see [12, Lemma 7, Theorem 2]. It will be used to assign values to hyperstructure vertices.

Theorem 3.4. *For a discounted SSG Γ and a pure positional* MAX *strategy σ, the values of all vertices in Γ_σ (when* MAX *fixes his strategy σ) are equal to the components of the unique optimal solution to the linear program*

$$\textit{maximize} \sum_{v_i \in V} v_i \quad \textit{subject to } S_\Gamma(\sigma). \tag{1}$$

4. Discounted simple stochastic games generate RLG-functions

In this section we prove the following:

Theorem 4.1. *Every discounted SSG induces an RLG function.*

We first describe how a discounted SSG induces a function (valuation) on its hyperstructure of pure strategies, and then prove that this function is RLG. Assume, till the end of this section, that Γ is an arbitrary but fixed discounted SSG. Let $\mathcal{P} = \prod_{j=1}^{d} \mathcal{P}_j$ be the hyperstructure of pure positional strategies of MAX in Γ, with $\mathcal{P}_j$ corresponding to the set of edges leaving vertex $v_j \in V_{\text{MAX}}$. Define a (strategy evaluation) function $f_\Gamma : \mathcal{P} \to \mathbb{R}$ by associating to each strategy $\sigma \in \mathcal{P}$ the optimal value of the linear program (1) from Theorem 3.4.

The direct (inefficient) way to define the f_Γ-improving directions between neighbors on $\mathcal{P}$ would be to solve a linear program for every neighbor of the current strategy and to define switches to neighbors with bigger f_Γ-values as improving. We will proceed more efficiently, after introducing the important concepts of attractive and profitable switches.

Definition 4.2. Assume the current strategy is $\sigma \in \mathcal{P}$, the LP (1) is solved with the (unique) optimal solution v^* assigning finite values to all variables of $S_\Gamma(\sigma)$ and, respectively, to all vertices of Γ. Let the current edge e selected by σ in vertex v_i correspond to constraint $v_i \leqslant p v_j$ in $S_\Gamma(\sigma)$. Suppose, there is another edge e' (not currently selected by σ) corresponding to constraint $v_i \leqslant p' v_k$ such that $p v_j^* < p' v_k^*$. In this case, we say that a *switch* in σ of edge e to edge e', resulting in a new neighboring strategy σ', is *attractive*. In the corresponding LP $S_\Gamma(\sigma')$ the constraint $v_i \leqslant p v_j$ of $S_\Gamma(\sigma)$ is replaced with $v_i \leqslant p' v_k$. Say that a switch from σ to σ' is *profitable* if $\max(\sum_{v_i \in V} v_i | S_\Gamma(\sigma)) < \max(\sum_{v_i \in V} v_i | S_\Gamma(\sigma'))$.

The following nice property is crucial for an efficient monotonic iterative strategy improvement.

Lemma 4.3. *Every attractive switch is profitable.*

Proof. The proof is based on the fact that all constraints in S_Γ have a *monotonic* form $x \leqslant P(\bar{x}) + w$, where P is a homogeneous linear polynomial with *nonnegative* coefficients (actually, discountedness in not used). Let σ have an attractive switch to σ', resulting in the replacement of the constraint $v_i \leqslant p v_j$ with $v_i \leqslant p' v_k$. If vector v^* is a finite optimal solution to $\max(\sum_{v_i \in V} v_i | S_\Gamma(\sigma))$, then v^* satisfies $v_i \leqslant p v_j$ as equality; otherwise it is not optimal, since v_i^* can be increased. By monotonicity and attractiveness, v^* also satisfies $S_\Gamma(\sigma')$, but the new constraint $v_i \leqslant p' v_k$ is satisfied as $<$. Therefore, v_i^* can be increased in $S_\Gamma(\sigma')$ (this may only increase right-hand sides of other constraints; thus none of them will be violated), yielding a better solution, i.e., the switch is profitable. Moreover, none of the vertices decrease its value, and the vertex for which the switch was made strictly increases its value after the switch. $\square$

The next definition subsumes local optimality (see the proof of Theorem 4.1).

Definition 4.4 (*Stability*). A strategy without attractive switches is called *stable*.

Discountedness is essentially used in the proof of the following key lemma.

Lemma 4.5. *For a discounted SSG every stable strategy of* MAX *is globally optimal.*

Proof. Let σ be stable in Γ and v^* be the optimal solution to the LP (1). Select any MIN constraints in $S_\Gamma(\sigma)$ satisfied as equalities by v^*, one per MIN-vertex, and call this selection τ an *optimal counterstrategy* against σ. Discard all MIN constraints not used in τ and denote the resulting system $S_\Gamma(\sigma, \tau)$. Let σ' be an allegedly better than stable strategy σ. Denote by $S_\Gamma(\sigma', \tau)$ the constraint system obtained from $S_\Gamma(\sigma, \tau)$ by replacing constraints corresponding to σ with those corresponding to σ'. Both $S_\Gamma(\sigma, \tau)$ and $S_\Gamma(\sigma', \tau)$ can be written as linear equality systems $Av = b$ and $A'v = b'$, respectively, where $b = b'$ are the vectors with $b_1 = b'_1 = 1$ and zeros elsewhere (all right-hand sides are zeros, except for the 1-sink). Let v' be an optimal solution to $\max(\sum_{v_i \in V} v_i | A'v = b')$ and $v' = v^* + \Delta v$. Since v' satisfies the constraints, $A'(v^* + \Delta v) = A'v^* + A'\Delta v = b'$. But $A'v^* \geqslant b'$, because σ is stable, hence for every constraint $v_i \leqslant p'v_k$ not in σ one has $v_i^* \geqslant p'v_k^*$. Now, $A'v^* + A'\Delta v = b'$ and $A'v^* \geqslant b'$ imply that $A'\Delta v \leqslant 0$. The latter implies that $\Delta v \leqslant 0$. Indeed, assuming Δv has a positive coordinate, select the largest such coordinate Δv_i, and get a contradiction, because the corresponding constraint cannot be satisfied due to discountedness of S_Γ (the sum of probabilities on the right-hand side of constraint is < 1). But $\Delta v \leqslant 0$ means that $v' = v^* + \Delta v$ is no better than v^*, and we get a contradiction with the assumption that σ' improves a stable strategy σ. □

We are now ready to complete the proof of the main claim in this section.

Proof of Theorem 4.1. Let $\mathcal{P}'$ be a substructure of $\mathcal{P}$, corresponding to a subgame Γ' of Γ, in which some edges of player MAX are thrown away. The proof that every local maximum σ on $\mathcal{P}'$ is global on $\mathcal{P}'$ consists of two implications: σ is locally optimal $\Rightarrow$ σ is stable $\Rightarrow$ σ is globally optimal. For the first implication, assume σ is not stable; then it has an attractive switch to a neighboring strategy, which has a bigger value than σ by Lemma 4.3. Hence, σ is not a local maximum. The second implication is proved by Lemma 4.5. □

By an appropriate generalization (see, e.g., our work on controlled linear programming [8]), the technique developed in this and preceding sections extends to Shapley's stochastic games [34], in which players select moves simultaneously and accumulate payoffs throughout the course of play. These games also generate RLG-functions and are amenable to subexponential algorithms described in the next section.

5. Optimizing RLG functions in subexponential time

Theorem 5.1. *A global maximum of an RLG-function* $f : \prod_{j=1}^d \mathcal{P}_j \to \mathbb{R}$ *can be found in expected subexponential time* $e^{2\sqrt{d \ln(m/\sqrt{d})} + O(\sqrt{d} + \ln m)}$, *where* $m = \sum_{j=1}^d |\mathcal{P}_j|$. *When m is polynomial in d, then this time is* $2^{O(\sqrt{d \log d})}$.

Proof. The idea consists in modifying and adapting the randomized subexponential algorithm for linear programming due to Matoušek et al. [29,30] for the RLG-functions. Note that in contrast to linear programming, the RLG-functions induced by SSGs are *nonconvex*.

Algorithm 1 written in pseudocode is self-explanatory. It first checks (base case) whether the hyperstructure $\mathcal{P}$ has just one possible strategy σ_0 left. In line (4), it recurses on a substructure $\mathcal{P} \backslash F$ obtained after deleting the facet F from $\mathcal{P}$. In line (5), it checks attractiveness of a switch to the facet F, previously thrown away. The algorithm always terminates since the recursive calls in lines (4), (6) are made on strictly smaller substructures, and every switch in line (6) is profitable (monotonic improvement). It is correct since any vertex without better neighbors in an RLG-structure is globally optimal. Any MAX strategy σ_0 can be used in the initial call to the algorithm.

The subexponential expected running time follows from the analysis in [29,30]. It is based on solving a probabilistic recurrence for the expected decrease of the so-called *hidden dimension*. The facets F_i not containing vertex σ_0 define the total ordering by the largest value of the RLG-function f on $\mathcal{P} \backslash F_i$. Choosing uniformly at random such a facet and finding an optimum on $\mathcal{P} \backslash F_i$, the algorithm will never revisit any of the preceding substructures $\mathcal{P} \backslash F_j$ in the ordering. We refer the reader to [29,30] for details. □

Checking attractiveness in line (5) of Algorithm 1, for a discounted SSG Γ, can be done (as explained in Definition 4.2) by using the optimal solution of the LP (1) for $\sigma = \sigma_0$, which the algorithm may additionally return from line (2). If we want to avoid problems with nonstrongly polynomial LP-algorithms and high precision arithmetic (coefficients

```
Algorithm 1: MSW-Style Optimization Algorithm
MSW(RLG-structure $\mathcal{P} = \prod_{i=1}^{d} \mathcal{P}_i$, initial vertex $\sigma_0 \in \mathcal{P}$)
(1)  if $|\mathcal{P}_i| = 1$ for each $\mathcal{P}_i$, $i = 1, \ldots, d$
(2)      return $\sigma_0$
(3)  choose a random facet $F$ of $\mathcal{P}$, not containing $\sigma_0$
(4)  $\sigma^* \leftarrow$ MSW($\mathcal{P} \backslash F$, $\sigma_0$)
(5)  if neighbor $\sigma$ of $\sigma^*$ on $F$ is better than $\sigma^*$
(6)      return MSW($F$, $\sigma$)
(7)  else
(8)      return $\sigma^*$
```

for probabilities involved may be as close to 1 as $1 - 1/2^{poly(n)}$, where n is the number of game vertices), we can rely on combinatorial subexponential LP-algorithms of Kalai [26,27], or Matoušek et al. [29,30]. This will keep the same subexponential upper bound, but, as a disadvantage, will make the cost of just one switch *subexponential* rather than (nonstrongly) polynomial. An improvement showing how to avoid solving linear programs altogether in described in [36].

Note that Algorithm 1 generalizes Ludwig's $2^{O(\sqrt{d})}$ algorithm [28], applicable to binary SSGs in which every MAX vertex has outdegree at most two. Actually, the algorithm coincides with Ludwig's on hypercubes, and thus is also $2^{O(\sqrt{d})}$. Although an arbitrary outdegree SSG reduces to a binary SSG, this results, in general, in a quadratic blow-up in the number of vertices; hence, Ludwig's algorithm becomes exponential. Other LP-algorithms that can be adapted for subexponential RLG-optimization, with essentially the same subexponential bounds, are due to Kalai [26,27]. Actually our first subexponential algorithm for PGs [3] was based on [26].

6. Mean payoff games (MPGs)

MPGs [15,20,43,32,6,9] are full-information infinite adversary games played on finite directed leafless[4] edge-weighted graphs $G = (V, E, w)$, with $V = V_{\max} \cup V_{\min}$, $V_{\max} \cap V_{\min} = \emptyset$, and $w : E \to \{-W, \ldots, W\} \subset \mathbb{Z}$.[5] Starting in an initial vertex v_0 players MAX and MIN select edges (if $v_i \in V_{\max}$ then MAX selects an edge from v_i and moves to its destination vertex v_{i+1}; otherwise MIN makes a choice) and as a result construct an infinite sequence of edges (v_i, v_{i+1}), for $i \geqslant 0$. Player MAX/MIN wants to maximize/minimize, respectively, the quantities

$$\liminf_{k\to\infty} \frac{1}{k} \sum_{i=0}^{k-1} w(v_i, v_{i+1}) \quad \text{and} \quad \limsup_{k\to\infty} \frac{1}{k} \sum_{i=0}^{k-1} w(v_i, v_{i+1}).$$

It turns out that MPGs are solvable in positional strategies for both players: every vertex has a value, which both players can secure by applying these strategies, independently of the initial vertex [15,20,6]. When a player fixes his positional strategy, an optimal counterstrategy of the adversary is polynomial time computable. Consequently, the problem whether this value is above/below a certain threshold is in NP ∩ coNP [43].

The previous known algorithms for solving MPGs (finding strategies and values) are as follows: (1) An exponential in the number of vertices algorithm implicit in [15]. (2) A pseudopolynomial (in the largest absolute edge weight) based on potential transformations [20,32]. (3) A pseudopolynomial based on dynamic programming and another one based on fixpoint iteration of a contractive mapping [43]. (4) A recent one, strongly subexponential in the number of vertices and simultaneously pseudopolynomial (in the largest edge weight), based on the new *Longest–Shortest Paths* (LSP) problem and randomized combinatorial linear programming [9] will be described here. (5) Another recent subexponential LCP-based algorithm is described in [7]. LCP stands for the *Linear Complementarity Problem*: find $w, z \geqslant 0$ satisfying $w = Mz + q$ and $w^{T} \cdot z = 0$.

Zero-mean partitioning for MPGs: It turns out that to find exact values of all vertices in an MPG, it is enough to determine their signs. This allows us to concentrate on a particularly simple problem [9].

[4] No sinks, i.e., every vertex has an outgoing edge.
[5] It can be assumed, without loss of generality, that in an MPG every cycle contains a vertex of player MAX.

Definition 6.1 (*0-Mean partitioning*). The MPG 0-*mean partition* problem consists in partitioning the vertices of a game G into subsets $V_G^{\leqslant 0}$ and $V_G^{>0}$ such that MAX (resp. MIN) can secure a positive (resp. nonpositive) value for every vertex of $V_G^{>0}$ (resp. $V_G^{\leqslant 0}$).

Other MPG-related problems, including finding exact values, ergodic partitioning, optimal strategies, are all reducible to 0-mean partitioning, using dichotomy, edge reweighing, and approximation [20,43,9].

Theorem 6.2. *Finding exact vertex values and optimal strategies in MPGs are polynomial time reducible to the* 0-*mean partitioning*.

7. Longest-shortest paths: reduction from MPGs

We reduce the 0-mean partition problem to the following new *controlled optimization problem* [9].

Definition 7.1 (*Longest-shortest paths (LSP)*). Given a directed edge-weighted graph $\Gamma = (V \cup \{t\}, E)$ with unique sink t and subset $C \subseteq V$ of *controlled* vertices, make a selection σ of *exactly one* edge leaving each vertex from C in such a way that in the resulting graph Γ_σ (obtained by deleting all edges not in σ leaving vertices in C) the shortest paths distances from all vertices to the sink are as large as possible.

We may consider the LSP problem as a two-stage game: MAX selects a strategy σ in controlled vertices C, and MIN responds by computing the shortest paths in the resulting graph. The payoff is the vector of the shortest distances from each vertex to the sink, which MAX wants to maximize.

Remark. For a fixed selection σ the shortest distances in Γ_σ to t may be computed in polynomial time by the Bellman–Ford algorithm. As usual, negative-weight cycles count as $-\infty$ and positive-edge ones count as $+\infty$. Zero-weight cycles may be counted as either $+\infty$ or 0. To avoid complications, we may assume, without loss of generality, that the graphs we are interested in do not have 0-weight cycles; see Proposition 7.2.

The absence of 0-weight cycles is not a restriction for our purposes.

Proposition 7.2. *Every MPG can be transformed into a game without* 0-*weight cycles with the same* 0-*mean partition*.

Proof. For an MPG on n vertices, multiply the weight of each edge by $n + 1$ and subtract 1. This preserves the signs of all negative and positive cycles, but transforms all previously 0-weight cycles into *negative* ones. Therefore, this transformation does not change the 0-mean partition. □

The reduction of MPGs to LSP instances is easy to describe.

Definition 7.3 (*MPG to LSP reduction*). To solve the 0-mean partition problem for an MPG G, turn it into an instance of the LSP problem Γ by adding to the graph of G the sink t, and *retreat edges* of weight 0 from every MAX vertex to sink t.

The correctness of this reduction is provided by the following:

Theorem 7.4. *For an MPG G the positive mean partition $V_G^{>0}$ coincides with the set of vertices in the associated LSP Γ for which the optimal distance is $+\infty$, provided G has no* 0-*weight cycles* (*see Proposition* 7.2).

Proof. Let MAX secure the $+\infty$ distance for v in Γ. Then he may use the same positional strategy σ in the MPG G (which never uses a retreat edge to the sink). By construction, this corresponds to a positive-weight cycle in G, and $v \in V_G^{>0}$. Conversely, a positional MPG strategy in G securing a positive mean value secures a $+\infty$ distance in the LSP instance Γ. □

Remark. This proof relies on the fact that MPGs are solvable in positional strategies (note that all strategies in the LSP problem are positional by definition). It is also important that an MPG has no 0-weight cycles. Otherwise, a 0-weight cycle enforced by MAX and cutting MIN from the sink gives distance $+\infty$, but does not yield a positive mean.

8. Partial RLG-functions

In general, the LSP problem instances do not induce RLG-functions, because their strategy evaluation functions are not defined for every strategy (vertex of the associated hyperstructure). We need to extend RLG-functions to cover this case as well. Our extension to PRLG-functions simultaneously allows for:
(1) Partially defined functions.
(2) Partially ordered codomains.
(3) Infinite values.

Definition 8.1 (*PRLG-functions*). Let $\mathcal{D}$ be a partially ordered set. A partial function $f : \mathcal{P} \rightharpoonup \mathcal{D}$ with domain $X \subseteq \mathcal{P}$ is called PRLG if, for every substructure $\mathcal{P}'$ of $\mathcal{P}$, every local maximum of $f|_{\mathcal{P}' \cap X}$ is also global on $\mathcal{P}' \cap X$.

We start by demonstrating that LSP instances induce PRLG-functions. Afterward, we show that every PRLG-function can be maximized in randomized expected subexponential time. This can be done abstractly (generalizing Theorem 5.1), without involving any MPG- or LSP-related details.

9. LSP instances induce partial RLG-functions

Let us show that MPG-generated (Definition 7.3) LSP instances induce PRLG-functions. Let G be an MPG with n vertices, d of which belong to MAX, and Γ be the corresponding LSP instance. The function f_Γ is a partial function on the hyperstructure $\mathcal{P} = \prod_{i=1}^{d} \mathcal{P}_i$, where $\mathcal{P}_i$ is a set of edges in Γ (including retreat edges) leaving the ith vertex of MAX. The codomain of f_Γ is $\mathcal{D} = (\mathbb{R} \cup \{+\infty\})^n$, the set of n-vectors over reals $\mathbb{R}$ extended with $+\infty$, needed to give values to positive infinite cycles. We assume that $x < +\infty$ for every $x \in \mathbb{R}$ and denote by $<$ the partial order defined by extending the total order $<$ on $\mathbb{R} \cup \{+\infty\}$ coordinatewise: $(x_1, \dots, x_n) < (y_1, \dots, y_n)$ iff $x_i \leqslant y_j$ for all i, and at least for one i the inequality is strict. Whenever the function f_Γ is defined on $\sigma \in \mathcal{P}$, the ith component of the value vector $f_\Gamma(\sigma)$ contains the *shortest path* distance from the ith vertex to the sink in the graph Γ_σ.

First, we define *admissible strategies* and their valuations.

Definition 9.1 (*Strategy admissibility and valuation*). Say that a strategy $\sigma \in \mathcal{P}$ is *admissible* if for every vertex v of Γ_σ the length of the shortest path $d_\sigma(v)$ in Γ_σ from v to the sink is either finite or $+\infty$.

For an admissible strategy $\sigma \in \mathcal{P}$ the value function is defined as

$$f_\Gamma(\sigma) = (d_\sigma(v_1), \dots, d_\sigma(v_n)). \tag{2}$$

For an inadmissible strategy the value function is undefined.

Corollary 9.5 shows that f_Γ thus defined is a PRLG-function. Note that f_Γ is partially defined, in general, and has a partially ordered codomain. Admissibility is polynomial time verifiable. Use the Bellman–Ford (or any available) algorithm to compute the single sink shortest distances. It will automatically detect negative cycles, if any, thus determining inadmissibility. Otherwise, it will produce the valuation (2). Note that in the MPG-generated LSP instances there is always an admissible strategy in which MAX selects moving to the sink from every vertex.

Second, we formalize the concept of attractive switch.

Definition 9.2 (*Attractiveness*). Let σ be an admissible strategy of value $f_\Gamma(\sigma)$ defined by (2), v_i be a MAX vertex, $(v_i, v_j) \in \sigma$, and v'_j be another successor of v_i. A switch in σ, changing the successor of v_i from v_j to v'_j, resulting in a new strategy $\sigma' \ni (v_i, v'_j)$ (a neighbor of σ in $\mathcal{P}$) is *attractive* if

$$d_\sigma(v_i) < w(i, j') + d_\sigma(v_{j'}). \tag{3}$$

Note that deciding attractiveness is easy: only the known shortest distances with respect to the current strategy σ are used in (3).

Third, we show that making attractive switches guarantees monotonic increases in strategy evaluation.

Theorem 9.3 (*Attractiveness is profitable*). *Every attractive switch $\sigma \to \sigma'$ preserves admissibility and guarantees profitability, i.e., $f_\Gamma(\sigma) < f_\Gamma(\sigma')$.*

Proof. Consider an attractive switch in vertex v_i, from successor v_j (in an admissible strategy σ) to $v_{j'}$ (in σ'). Recall that the (v_i, t)-shortest path problem is expressible by the linear program

$$\begin{array}{rl} \text{maximize} & v_i \\ \text{subject to} & t = 0, \ \text{for sink } t, \\ & v_i \leqslant w(i,j) + v_j, \ \text{for each edge } (v_i, v_j) \text{ of weight } w(i,j). \end{array}$$

Suppose $d_\sigma = (d_\sigma(v_1), \ldots, d_\sigma(v_n))$ is the vector of shortest distances from all vertices to the sink in Γ_σ. This vector is a feasible solution to the LP above. Let $v_i \leqslant w(i,j) + v_j$ be the constraint corresponding to the edge (v_i, v_j) currently selected by σ, and $v_i \leqslant w(i,j') + v_{j'}$ be the constraint corresponding to an alternative edge $(v_i, v_{j'})$ selected by σ'. In order to speak about attractiveness, $d_\sigma(v_i)$ needs to be finite. Note that the constraint $v_i \leqslant w(i,j) + v_j$ corresponding to the current choice is satisfied by d_σ as equality; otherwise, $d_\sigma(v_i)$ is not the shortest path distance. By attractiveness, $d_\sigma(v_i) < w(i,j') + d_\sigma(v_{j'})$. Thus, replacing (switching) the old constraint $v_i \leqslant w(i,j) + v_i$ with the new $v_i \leqslant w(i,j') + v_{j'}$ results in a new LP, which keeps the old solution d_σ as feasible (thus we get admissibility), but also allows for *increasing* the value of v_i (thus we get profitability), since the new constraint is not tight for v_i in d_σ and increasing v_i may only increase the right-hand sides of other constraints. Note that this argument is based on the special monotone form of the linear constraints for the shortest path problem. □

Remark. By the same argument, making simultaneously *several* (rather than just one at a time) attractive switches is profitable, but we do not need it in this paper.

Fourth, we prove that a stable strategy (without attractive switches) is optimal.

Theorem 9.4 (*Stability implies optimality*). *Every stable strategy in an LSP-instance Γ is optimal, provided Γ has no zero-weight cycles.*

Proof. By Proposition 7.2, assuming the absence of 0-weight cycles is no loss of generality. Suppose σ is a stable strategy, and let d_σ be the vector of shortest distances in Γ_σ to the sink (one component per vertex). Suppose, toward a contradiction, that there is a better than σ strategy σ', providing for a longer distance $d_{\sigma'}(v_0) > d_\sigma(v_0)$, in $\Gamma_{\sigma'}$, for some vertex v_0. Note that $d_\sigma(v_0)$ should be finite and the shortest path in Γ_σ from v_0 leads to the sink.

For every edge $u \overset{w}{\to} v$ in Γ_σ the relation $d_\sigma(u) \leqslant d_\sigma(v) + w$ holds, because d_σ are shortest distances. Recall that in Γ_σ MAX has one edge per vertex. Therefore, $d_\sigma(u) = d_\sigma(v) + w$ for every edge $u \overset{w}{\to} v$ of MAX in Γ_σ (tightness). Moreover, since σ is stable, $d_\sigma(u) \geqslant d_\sigma(v') + w'$ for every other edge $u \overset{w'}{\to} v'$ of MAX not in σ (no attractive switches).[6]

Let Γ_σ^f and $\Gamma_\sigma^{+\infty}$ be the partition of Γ_σ into the sets of vertices with finite and infinite shortest distances d_σ. Thus, v_0 is in Γ_σ^f. Note that in Γ (full graph, not just Γ_σ) MAX has no edges from Γ_σ^f to $\Gamma_\sigma^{+\infty}$. Otherwise, σ has an attractive switch to this edge (from a finite to infinite value), contradicting the stability assumption. Although MIN may have edges from Γ_σ^f to $\Gamma_\sigma^{+\infty}$, they are not parts of the shortest paths under σ, and we may delete them. Let us also throw away the part $\Gamma_\sigma^{+\infty}$, and in Γ_σ^f for each MIN vertex leave just one outgoing edge $u \overset{w}{\to} v$, for which $d_\sigma(u) = d_\sigma(v) + w$ (such edges exist!). On the contrary, we return back all edges of MAX to the partition Γ_σ^f (as explained above, they are not leaving Γ_σ^f). Denote the resulting graph Γ'.

[6] It is convenient to assume that $+\infty \bowtie +\infty + w$, for $\bowtie \in \{\leqslant, =, \geqslant\}$ and finite w.

In Γ', MAX has all strategies available, while MIN has just one. Nevertheless, we will show that in Γ' no strategy σ' of MAX can improve over $d_\sigma(v_0)$. Suppose the contrary, and make a *potential transformation* of Γ', by changing the weight of each edge according to the rule $w'_{u,v} = w_{uv} - d_\sigma(u) + d_\sigma(v)$. After this transformation: (1) all edges in σ become 0-weight (by tightness), and all other MAX edges become nonpositive (by stability, as explained above); (2) all edges of MIN become 0-weight; (3) weights of all cycles remain *unchanged* (by telescoping); (4) weights of finite paths $v_0, \ldots, v_l$ change by a constant $-d_\sigma(v_0) + d_\sigma(v_l)$ (by telescoping), hence the relation $(<, =, >)$ between costs of two paths to the sink remains unchanged.

Let MAX use an allegedly better than σ strategy σ' from v_0 in Γ' undergone the potential transformation above. Since MIN has just one available strategy, σ' defines a unique play, in which every edge traversed is *nonpositive* by the explanation above. This play may be either finite, or infinite.

Suppose it is *finite*, terminating in the sink. Let π and π' be the paths from v_0 to the sink t in Γ' determined by σ and σ', respectively, and functions c, c' give paths costs before and after the potential transformation. We have $c'(\pi') = c(\pi') - d_\sigma(v_0) + d_\sigma(t) = c(\pi') - d_\sigma(v_0) + 0 = c(\pi') - d_\sigma(v_0) = c(\pi') - c(\pi)$. Hence, $c(\pi) + c'(\pi') = c(\pi')$, and $c(\pi) \geqslant c(\pi')$ whenever $c'(\pi') \leqslant 0$. But $c'(\pi') \leqslant 0$, because each edge traversed in π' after the potential transformation is nonpositive. Therefore, $d_\sigma(v_0) = c(\pi) \geqslant c(\pi') = d_{\sigma'}(v_0)$, and σ' provides no improvement over σ, a contradiction.

If the play is *infinite*, then the value for v_0 under σ' must be $-\infty$. Indeed, by assumption (w.l.o.g.), the graph does not contain 0-weight cycles, and this property is preserved during any potential transformation. All edges used by MAX in σ' and by MIN have nonpositive weights after the potential transformation above. Since there are no zero-weight cycles, the only possible cycles are *negative*. Therefore, σ' gives no improvement in this case neither. This contradiction shows that σ is indeed an optimal strategy. □

By Theorems 9.3 and 9.4 we immediately obtain the desired.

Corollary 9.5. *Every MPG-generated LSP instance induces a PRLG-function.*

10. Optimizing partial RLG functions in subexponential time

Theorem 10.1. *A global maximum of every PRLG-function can be found in expected subexponential time, given a vertex of the hyperstructure in which the function is defined.*

Algorithm 1 (with RLG- replaced with PRLG-) and the subexponential bound remain the same as in Section 5. Note that the availability of an initial vertex in which the function is defined is essential. Indeed, consider a PRLG-function defined exactly in one vertex of the hyperstructure. Then finding this vertex cannot be done without full (exponential) vertex enumeration. Luckily, the MPG-generated LSP instances always have an admissible MAX strategy "go to the sink everywhere" available. Note that after finding an optimal strategy σ^* on $\mathcal{P}\backslash F$ in line (4) the algorithm has the shortest paths distances from all vertices to the sink (which may be returned from line (2), in addition to σ_0). Thus, it avoids solving linear programs and is purely combinatorial.

Corollary 10.2. *Every MPG can be solved in expected subexponential time* $2^{\mathrm{O}(\sqrt{d\log(d)})}$ (*d the number of* MAX *vertices*) *and simultaneously pseudopolynomial time* $\mathrm{O}(poly(n)W)$ (*n total number of vertices*, *W maximal absolute edge weight*).

The second bound follows from the fact that the algorithm proceeds by monotonic (profitable) switches, each time increasing the distances in (2) by integral increments.

Although MPGs are known to be polynomially reducible to SSGs [43] (thus the subexponential algorithm from Section 5 applies), the LSP-based algorithm has a crucial advantage. It does not rely on solving linear programs, either by nonstrongly polynomial methods (and associated precision and stability issues), or by strongly subexponential methods (with worse hidden constants and subexponential time per iteration).

A recent paper [42] demonstrates that LSP instances with *nonnegative* edge weights can be optimized in polynomial time, by an extension of Dijkstra's algorithm. Note that the reduction from MPGs requires, however, negative weights.

11. Parity games

A scheme for assigning discrete values to strategies in PGs and an exponential iterative strategy improvement algorithm for PGs were described in [39]. In [3] we suggested a modification of the strategy evaluation function from [39], which provides for a better complexity analysis (including a subexponential bound). Now, by reusing our development for MPGs and the LSP problem, we obtain an even faster and much simpler iterative strategy improvement algorithm for PGs.

Given a PG P, transform it into an MPG M by assigning all outgoing edges from a vertex of color $c \in \mathbb{N}$ the weight $(-n)^c$, where n is the number of game vertices. It is immediate that in the original game P player Max secures an even color infinitely often iff Max in M secures a positive mean. Note that 0-weight cycles in M are impossible by construction. Therefore, solving PGs reduces to the 0-mean partitioning for MPGs, and our subexponential algorithm applies. In addition, since the algorithm makes only profitable switches with integral increments, the total number of iterations is also *pseudopolynomial* $\mathrm{O}(poly(n)W)$, where W is the largest edge weight $W = \mathrm{O}(n^k)$, where k is the maximal color. Note that W can be considerably decreased (thus improving the pseudopolynomial bound) by assigning weights w_i to edges leaving vertices of color i more sparingly: $w_0 = 1$ and $w_{i+1} =$ the total weight of all edges leaving vertices of colors of the opposite parity up to i; see [9] for details . Thus, the LSP-based algorithm described above inherits all advantages of the algorithms [39,3], is more general, provides for even better worst-case bounds, and is much more simple to explain and implement than both.

12. Conclusions

We introduced and investigated recursively local-global (RLG-) and partial recursively local-global (PRLG-) functions on Cartesian products of simplices and showed that these functions are abstract counterparts of strategy evaluation functions in infinite parity, mean payoff, and SSGs on finite graphs. Solving such a game amounts to finding a global maximum of a (P)RLG-function. These functions generalize CU pseudo-Boolean functions [21,40,41,38], as well as AOFs [27]. We demonstrated that RLG- and PRLG-functions, hence all the above-mentioned games, can be optimized/solved in randomized subexponential time. A further effort is needed to understand and extract additional combinatorial properties of games (in addition to already captured by RLG- and PRLG-functions) in order to improve on a subexponential analysis.

Acknowledgments

We are grateful to Leonid Khachiyan, Vladimir Gurvich, and Endre Boros for inspiring discussions and illuminating ideas. We thank DIMACS for providing a creative environment.

References

[1] E. Aarts, J.K. Lenstra (Eds.), Local Search in Combinatorial Optimization, Wiley, New York, 1997.

[2] H. Björklund, S. Sandberg, S. Vorobyov, Optimization on completely unimodal hypercubes, Technical Report 2002-018, Uppsala University/Information Technology, May 2002 <http://www.it.uu.se/research/reports/>.

[3] 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).

[4] H. Björklund, S. Sandberg, S. Vorobyov, Complexity of model checking by iterative improvement: the pseudo-Boolean framework, in: M. Broy, A. Zamulin (Eds.), Andrei Ershov Fifth Internat. Conf. "Perspectives of System Informatics", Lecture Notes in Computer Science, Vol. 2890, 2003, pp. 381–394.

[5] H. Björklund, S. Sandberg, 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/>.

[6] H. Björklund, S. Sandberg, S. Vorobyov, Memoryless determinacy of parity and mean payoff games: a simple proof, Theoret. Comput. Sci. 310 (1–3) (2004) 365–378.

[7] H. Björklund, O. Svensson, S. Vorobyov, Linear complementarity algorithms for mean payoff games, Technical Report DIMACS-2005-05, DIMACS: Center for Discrete Mathematics and Theoretical Computer Science, Rutgers University, NJ, February 2005 <http://dimacs.rutgers.edu/TechnicalReports/>.

[8] H. Björklund, O. Svensson, S. Vorobyov, Controlled linear programming for infinite games, Technical Report DIMACS-2005-13, DIMACS: Center for Discrete Mathematics and Theoretical Computer Science, Rutgers University, NJ, April 2005 <http://dimacs.rutgers.edu/TechnicalReports/>.
[9] H. Björklund, S. Vorobyov, A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games, Preliminary version in MFCS'04, Lecture Notes in Computer Science, Vol. 3153, Springer, Berlin, 2005, pp. 673–685, and DIMACS TR 2004-05, Discrete Appl. Math., to appear.
[10] E. Boros, P.L. Hammer, Pseudo-Boolean optimization, Technical Report RRR 48-2001, RUTCOR Rutger Center for Operations Research, 2001.
[11] E.M. Clarke, O. Grumberg, D. Peled, Model Checking, MIT Press, Cambridge, MA, 2000.
[12] A. Condon, The complexity of stochastic games, Inform. Comput. 96 (1992) 203–224.
[13] A. Condon, On algorithms for simple stochastic games, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 13, 1993, pp. 51–71.
[14] C. Derman, Finite-State Markovian Decision Processes, Academic Press, New York, 1970.
[15] A. Ehrenfeucht, J. Mycielski, Positional strategies for mean payoff games, Internat. J. Game Theory 8 (1979) 109–113.
[16] E.A. Emerson, Model checking and the Mu-calculus, in: N. Immerman, Ph.G. Kolaitis (Eds.), DIMACS Series in Discrete Mathematics, Vol. 31, 1997, pp. 185–214.
[17] E.A. Emerson, C.S. Jutla, Tree automata, μ-calculus and determinacy, in: Annu. IEEE Symp. on Foundations of Computer Science, 1991, pp. 368–377.
[18] J. Filar, K. Vrieze, Competitive Markov Decision Processes, Springer, Berlin, 1996.
[19] E. Grädel, W. Thomas, T. Wilke (Eds.), Automata Logics and Infinite Games. A Guide to Current Research, Lecture Notes in Computer Science, Vol. 2500, Springer, Berlin, 2003.
[20] V.A. Gurvich, A.V. Karzanov, L.G. Khachiyan, Cyclic games and an algorithm to find minimax cycle means in directed graphs, U.S.S.R. Comput. Math. Math. Phys. 28 (5) (1998) 85–91.
[21] P.L. Hammer, B. Simeone, Th.M. Liebling, D. De Werra, From linear separability to unimodality: a hierarchy of pseudo-Boolean functions, SIAM J. Discrete Math. 1 (2) (1988) 174–184.
[22] P. Hansen, B. Jaumard, V. Mathon, Constrained nonlinear 0–1 programming (state-of-the-art survey), ORSA J. Comput. 5 (2) (1993) 97–119.
[23] A.J. Hoffman, R.M. Karp, On nonterminating stochastic games, Management Sci. 12 (5) (1966) 359–370.
[24] R.A. Howard, Dynamic Programming and Markov Processes, MIT Press, Cambridge, MA, 1960.
[25] M. Jurdziński, Small progress measures for solving parity games, in: H. Reichel, S. Tison (Eds.), 17th STACS, Lectures Notes in Computer Science, Vol. 1770, Springer, Berlin, 2000, pp. 290–301.
[26] G. Kalai, A subexponential randomized simplex algorithm, in: 24th ACM STOC, 1992, pp. 475–482.
[27] G. Kalai, Linear programming, the simplex algorithm and simple polytopes, Math. Programming (Ser. B) 79 (1997) 217–234.
[28] W. Ludwig, A subexponential randomized algorithm for the simple stochastic game problem, Inform. Comput. 117 (1995) 151–155.
[29] J. Matoušek, M. Sharir, M. Welzl, A subexponential bound for linear programming, in: eighth ACM Symp. on Computational Geometry, 1992, pp. 1–8.
[30] J. Matoušek, M. Sharir, M. Welzl, A subexponential bound for linear programming, Algorithmica 16 (1996) 498–516.
[31] C. Papadimitriou, Algorithms, games, and the internet, in: ACM Annu. Symp. on Theory of Computing, ACM, New York, July 2001, pp. 749–753.
[32] N. Pisaruk, Mean cost cyclical games, Math. Oper. Res. 24 (4) (1999) 817–828.
[33] A. Puri, Theory of hybrid systems and discrete events systems, Ph.D. Thesis, EECS University of Berkeley, 1995.
[34] L.S. Shapley, Stochastic games, Proc. Natl. Acad. Sci. USA 39 (1953) 1095–1110.
[35] M. Sharir, E. Welzl, A combinatorial bound for linear programming and related problems, ninth Symp. on Theoretical Aspects of Computer Science (STACS), Lecture Notes in Computer Science, Vol. 577, Springer, Berlin, 1992, pp. 569–579.
[36] O. Svensson, S. Vorobyov, A subexponential algorithm for a subclass of P-matrix generalized linear complementarity problems, Technical Report DIMACS-2005-20, DIMACS: Center for Discrete Mathematics and Theoretical Computer Science, Rutgers University, NJ, June 2005.
[37] C.A. Tovey, Low order polynomial bounds on the expected performance of local improvement algorithms, Math. Programming 35 (1986) 193–224.
[38] C.A. Tovey, Local improvement on discrete structures, in: E. Aarts, J.K. Lenstra (Eds.), Local Search in Combinatorial Optimization, Wiley, New York, 1997, pp. 57–89.
[39] J. Vöge, M. Jurdziński, A discrete strategy improvement algorithm for solving parity games, in: E.A. Emerson, A.P. Sistla (Eds.), CAV'00: Computer-Aided Verification, Lecture Notes in Computer Science, Vol. 1855, Springer, Berlin, 2000, pp. 202–215, Extended version available as BRICS TR RS-00-48, 2000 <http://www.brics.dk/RS/00/48>.
[40] D. Wiedemann, Unimodal set-functions, Congr. Numer. 50 (1985) 165–169.
[41] K. Williamson Hoke, Completely unimodal numberings of a simple polytope, Discrete Appl. Math. 20 (1988) 69–81.
[42] J. Zhao, V. Gurvich, L. Khachiyan, Extending Dijkstra's algorithm to shortest paths with blocks, Technical Report DIMACS-2005-04, DIMACS: Center for Discrete Mathematics and Theoretical Computer Science, Rutgers University, NJ, February 2005 <http://dimacs.rutgers.edu/TechnicalReports/>.
[43] U. Zwick, M. Paterson, The complexity of mean payoff games on graphs, Theoret. Comput. Sci. 158 (1996) 343–359.