Parity Games: Interior-Point Approach *†‡

Viktor Petersson & Sergei Vorobyov
Information Technology/Computing Science Department
Uppsala University, Box 337
751 05 Uppsala, SWEDEN
Email: vorobyov@csd.uu.se

May 11, 2001

Abstract

We introduce and investigate continuous optimization techniques for solving Parity Games, based on the *interior-point paradigm*, combining barrier functions and quasi-Newton methods. These have been proven very successful for Linear and Convex Programming. The basic step is to leave the boundary (i.e. the stationary strategies in which such games can be solved) and to cut through the interior (i.e. probabilistic strategies) of the many-dimensional hypercube of strategies.

*Technical Report 2001-008, Department of Information Technology, Uppsala University, Sweden, http://www.it.uu.se/research/reports/2001-008/

†Supported by TFR Grant 221-2000-103 "Games for Computer-Aided Verification: Algorithms and Complexity"

‡Short communication in: IEEE Annual Symposium on Logic in Computer Science (LICS'2001), Boston, USA, June 2001

Contents

1 Introduction

In this paper we suggest and investigate a radically new approach to solving PARITY GAMES, or, equivalently, the MU-CALCULUS MODEL CHECKING problem, based on the *interior-point paradigm.* All previous approaches can be characterized as *purely combinatorial* or *iterative fixpoint*, and are based exclusively on analyzing *stationary or pure game strategies.* The novelty of our approach consists in:

1. Considering *probabilistic strategies* on the way to finding (by stepwise improvement) optimal stationary strategies.

2. Extending the cost functions associated to stationary strategies and corresponding to *vertices* of many-dimensional hypercubes to the *full interior of those hypercubes*[1].

3. Using powerful *continuous*, rather than *discrete*, methods of mathematical optimization, most notably *interior-point methods*, characterized by the combination of *barrier functions* and *quasi-Newton* unconditional optimization techniques.

Interior-point methods became popular and well known as giving the best, both theoretically and practically, polynomial-time algorithms for LINEAR PROGRAMMING (Karmarkar 1984). It seems natural to investigate similar techniques and approaches in the context of games, and in this paper we make the first step in this direction.

The precise computational complexity (upper and lower bounds) of solving PARITY GAMES is a fundamental open problem in complexity theory, automata theory, temporal logic, and computer-aided program verification. PARITY GAMES is the following decision problem about determining a winner in two-person adversary full-information *infinite* games on *finite* graphs. Given a finite directed bi-partite graph without sinks, and vertices painted in k colors, two players alternate moves along edges. Player 1 tries to make sure that in any infinite play, independently of what Player 2 does, the *maximal color* hit *infinitely often* is *even*, whereas Player 2 tries to make it *odd.*

The fundamental importance of the problem stems from the fact that it is inter-reducible, by linearly bounded polynomial reductions, with the so-called MU-CALCULUS MODEL CHECKING problem (given a Kripke structure M and a μ-calculus formula ϕ, decided whether $M \models \phi$?), and both problems belong to the complexity class $\mathrm{NP} \cap \mathrm{coNP}$ (Emerson, Jutla & Sistla 1993,

[1]Each interior point naturally corresponds to a *probabilistic strategy* – choose successors with some probabilities that sum up to 1.

Thm 4.1, 4.2, 4.3). The Mu-calculus is the 'ultimate' logic expressively subsuming virtually all known propositional temporal programming logics, so characterizing precisely its model checking complexity is a challenging problem.

Despite a long-standing commonly believed conjecture that both PARITY GAMES and MU-CALCULUS MODEL CHECKING are in PTIME, all the best previously known decision procedures were *exponential* in the number of colors. The original iterative fixpoint algorithm (Emerson & Lei 1986), (Emerson 1997, Section 5) runs in time $O(n^k)$ and space $O(n)$, where n is the number of vertices in game graph and k is the number of colors. The best previously known algorithm (Browne, Clarke, Jha, Long & Marrero 1997) takes exponential time and space $O(k^2 \cdot n \cdot (\sqrt{n})^k)$. Subsequently Seidl (1996), Jurdzinski (2000) proposed other algorithms of the same EXPONENTIAL complexity.

Recently we suggested a *randomized subexponential algorithm* when the number of colors is large, $k = \Omega(\sqrt{n})$, (Vorobyov 2001, Petersson & Vorobyov 2001).

In Section 2 we recall basic definitions concerning SIMPLE STOCHASTIC GAMES (SSGs) and introduce our basic optimization problem. Section 3 describes properties of the cost functions associated to SSGs, such as *continuity* and the *unimin*, favorable for mathematical optimization, and introduces the techniques of improvement directions.

Section 4 describes several classes of relevant optimization techniques, including Simplex-like methods in Section 4.1 (that quite surprisingly are simple particular cases of our interior-point approach). It also discusses problems and possible improvements (including barrier functions and quasi-Newton methods). In Section 5 we investigate further properties of SSG-associated cost functions.

2 Preliminaries

We assume known basic definitions concerning PARITY GAMES (PGs), *linear and nonlinear programming*; see, e.g., (Emerson & Lei 1986, Emerson et al. 1993, Emerson 1997) and (Nash & Sofer 1996). However, in this paper we forget straight away about PARITY GAMES and concentrate exclusively on the so-called SIMPLE STOCHASTIC GAMES (SSGs), using the following facts:

1. There exists a chain of polynomial-time reductions: 1) from PARITY GAMES to the so-called MEAN PAYOFF GAMES (Puri 1995, Jurdziński 1998, Vorobyov 2001), and then 2) from MEAN PAYOFF GAMES

to Simple Stochastic Games (Zwick & Paterson 1996) with the property: Player 1 wins a PG iff Player 1 has a probability $\geq 1/2$ of reaching the 1-sink in the corresponding SSG. (This is the key to a subexponential algorithm for PGs, when the number of colors is large, based on the randomized subexponential algorithm due to Ludwig (1995); see also (Vorobyov 2001, Petersson & Vorobyov 2001).)

2. SSGs seem more *appropriate* for mathematical optimization techniques than (discrete) PGs. Indeed, it is often virtually impossible to compare the quality of two strategies in a PG (unless one is winning), whereas it is easy in the case of SSGs. This naturally allows for stepwise strategy improvements.

We briefly recall needed definition concerning SSGs.

Definition 2.1 *A* simple stochastic game (SSG) *is a directed graph* $G = (V, E)$*, where:*

1. *The set of vertices* $V = \{1, \ldots, n\}$*, with* $n \geq 2$*, is partitioned into the sets*
$$Max \cup Min \cup Avg \cup \{n-1\} \cup \{n\}$$
of Maximum*,* Minimum*,* Average*,* 0-sink *and* 1-sink *vertices respectively.*

2. *We only consider SSGs where every vertex in* $\{1, \ldots, n-2\}$ *has* exactly two *outgoing edges and where vertices* $n-1$ *and* n *do not have outgoing edges. (Any SSG can be reduced to such in polynomial time.)*

3. *The outgoing edges of every vertex are ordered and called the* left *and the* right *edges and labeled 0 and 1 respectively. For each* $i \in Max \cup Min \cup Avg$ *by* $l(i)$ *and* $r(i)$ *we denote the left and right successor vertices of* i*.*

4. *The left and the right edges of each average vertex* $i \in Avg$ *are assigned rational probabilities* $a_i \in [0, 1]$ *and* $1 - a_i$ *respectively.* □

Definition 2.2 *A (stationary)* strategy for Player 1 *is a bit vector* σ *indicating his choice of exactly one edge leaving every* Max*-vertex, with the* i*-th component 0 or 1 meaning the label of a chosen edge originating from* $i \in Max$*. Similarly, a* strategy for Player 2 *is a bit vector* τ *with the* i*-th component 0 or 1 indicating the label of an edge originating from* $i \in Min$*.* □

Definition 2.3 *When both players fix their stationary strategies σ, τ, the graph $G_{\sigma,\tau}$ results from G by deleting all edges not selected in strategies σ, τ. Say that an SSG G* halts with probability 1 *iff for every σ, τ the graph $G_{\sigma,\tau}$ has a path from every vertex to a sink.* □

Proviso. In the sequel we assume without loss of generality (see (Ludwig 1995, Zwick & Paterson 1996)) that all SSGs are halting with probability 1. This condition is automatically guaranteed by the translation[2] from PGs into SSGs. In this translation each $i \in Max \cup Min$ has $l(i), r(i) \in Avg$ and each $i \in Avg$ has a sink as a successor with nonzero corresponding probability. □

Definition 2.4 (Values) *The* value $v_{\sigma,\tau}(i)$ *of a vertex i with respect to strategies σ, τ is the probability that Player 1 reaches the 1-sink (wins) when the game starts in i and the players follow the strategies σ, τ.* □

Definition 2.5 (Stability, Optimality) *A vertex $i \in Max$ with successors $l(i)$ and $r(i)$ is* stable *with respect to strategies σ, τ iff $v_{\sigma,\tau}(i) = \max\{v_{\sigma,\tau}(l(i)), v_{\sigma,\tau}(r(i))\}$. Analogously, a vertex $i \in Min$ with successors $l(i)$ and $r(i)$ is* stable *with respect to strategies σ, τ iff $v_{\sigma,\tau}(i) = \min\{v_{\sigma,\tau}(l(i)), v_{\sigma,\tau}(r(i))\}$.*

A strategy τ of Player 2 is optimal *against a strategy σ of Player 1 iff every vertex in Min is stable with respect to σ, τ. Let $\tau(\sigma)$ denote a strategy optimal against σ.* □

2.1 Derman's Theorem

The following result due to Derman (1970) is one of the cornerstones of our approach. It allows for finding an optimal strategy against a *trivial* player who fixes his stationary strategy in advance.

Theorem 2.6 *The values and an optimal strategy for Player 1 in an SSG with no Min vertices that halts with probability 1 can be found by solving the following* Linear Program*:*

$$\left\{ \begin{array}{ll} \textit{minimize} & \sum_{i=1}^{n} v(i) \\ \textit{subject to} & v(i) \geq v(j) \textit{ for } i \in Max,\ j \in \{l(i), r(j)\} \\ & v(i) = a_i \cdot v(l(i)) + (1 - a_i) \cdot v(r(i)) \textit{ for } i \in Avg \\ & 0 \leq v(i) \leq 1 \textit{ for } 1 \leq i \leq n-2 \\ & v(n-1) = 0,\ v(n) = 1 \end{array} \right. \quad (1)$$

[2]Call it Puri-Jerrum-Zwick-Paterson's translation, or PJZP-translation

Sketch of the Proof. Let us show that any optimal solution to the LP (1) gives an optimal strategy for Player 1 in the SSG, or, equivalently, every Max-vertex is stable (wrt a dummy Player 2). Suppose not, i.e., for some optimal solution v to (1) and some Max-vertex i both $v(i) > v(l(i))$ and $v(i) > v(r(i))$ hold as strict inequalities (they cannot be smaller since v is a solution).

In this case we can construct a better solution to (1) with a smaller value of the target function (than provided by v) just replacing the component $v(i)$ with a smaller value $v(i)' = \min\{v(l(i)), v(r(i))\}$. This gives a contradiction with the optimality of v. Note that $v(i)$ appears only twice on the left of $\geq$ in (1); thus replacing $v(i)$ with $v(i)'$ does not violate any constraints. □

2.2 Turning the Min Player into the Average Player

The next step of our approach is to look at the Min-player (Player 2) as a *probabilistic player*, who follows a *probabilistic* (rather than *stationary*) strategy picking left and right edges leaving the vertex $i \in Min$ with probabilities $1 - p_i$ and $0 \leq p_i \leq 1$.

Since this turns all Min-vertices into the Avg-vertices, Derman's Theorem 2.6 applies and we can find an optimal strategy (for Player 1 against the probabilistic Player 2) and corresponding values of all vertices by solving an LP similar to (1), as shows the following

Theorem 2.7 *The values and an optimal strategy for Player 1 in an SSG halting with probability 1 in which Player 2 follows a probabilistic strategy by selecting at each Min-vertex i its successors $l(i)$ and $r(i)$ with probabilities $1-p_i$ and $p_i \in [0,1]$ respectively, can be found by solving the following* Linear Program*:*

$$f(\bar{p}) = \begin{cases} \textit{minimize} \quad \sum_{i=1}^{n} v(i) \\ \\ \textit{subject to} \\ \quad v(i) \geq v(j) \quad \textit{for } i \in Max,\ j \in \{l(i), r(j)\} \\ \quad v(i) = a_i \cdot v(l(i)) + (1 - a_i) \cdot v(r(i)) \quad \textit{for } i \in Avg \\ \quad v(i) = (1 - p_i) \cdot v(l(i)) + p_i \cdot v(r(i)) \quad \textit{for } i \in Min \\ \quad 0 \leq v(i) \leq 1 \quad \textit{for } 1 \leq i \leq n-2 \\ \quad v(n-1) = 0,\ v(n) = 1 \end{cases} \tag{2}$$

where $\bar{p}$ is a vector of probabilities p_i for each $i \in Min$. □

Definition 2.8 *Given an SSG G, call the function $f(\bar{p})$ defined by* (2) *in Theorem 2.7 the* cost function *associated to G.* □

Now the goal of Player 2 in an SSG is to select the vector of probabilities $\bar{p}$ that gives a *global minimum* of the associated cost function $f(\bar{p})$ on the unit hypercube of probabilities $0 \le p_i \le 1$ for $i \in Min$, i.e.,

$$\begin{cases} \text{minimize} & f(\bar{p}) \\ \text{subject to} & 0 \le p_i \le 1 \quad \text{for } i \in Min \end{cases} \tag{3}$$

The function $f(\bar{p})$ is well defined and for any *fixed* probabilities vector $\bar{p}$ its value is computable in *polynomial time.* Therefore, to solve (3) we may use any (continuous) method for finding global minima of $f(\bar{p})$ in the unit hypercube $\mathcal{H}$. In the following sections we will investigate properties of the function $f(\bar{p})$ and show, in particular, that:

1. The function is continuous, and it does not have local minima that are not global.
2. The method is correct in the sense that solving (3) gives values of all vertices of an SSG with respect to the pair of *optimal stationary strategies.*

Indeed, both facts are not immediately obvious. In Sections 4 we consider appropriate methods of (un)conditional optimization.

3 Properties of the Cost Functions

3.1 Continuity

Lemma 3.1 *For any SSG the associated function $f(\bar{p})$ is* continuous*.*

Proof Sketch. The constraints of the linear program (2) are hyperplanes with normal vectors defined by probabilities p_i's. When the latter change slightly the corresponding convex polyhedron also changes slightly and so does the minimum value of the target function $\sum_{i=1}^{n} v(i)$. □

3.2 Improving Directions

The definition below formalizes a discrete analog of the *gradient* for the cost function (2) associated to an SSG.

Definition 3.2 *Let $f(\bar{p})$ be the cost function associated to an SSG. Suppose $\bar{p}$ is a vector of probabilities p_i for $i \in Min$ and $v(1), \ldots, v(n)$ is an optimal solution of the LP $f(\bar{p})$. The* vector $\bar{d}$ of improving directions *d_i (for $i \in Min$) at the point $\bar{p}$ for the function $f(\bar{p})$ is defined as follows:*

1. *if* $0 < p_i < 1$ *then* $d_i = v(l(i)) - v(r(i))$

   (note that d_i *is positive if* $v(l(i)) < v(r(i))$ *and negative if* $v(l(i)) > v(r(i))$*; this corresponds to increasing the probability of selecting the vertex with the smaller value)*

2. *if* $p_i = 0$ *then*

   (a) *if* $v(l(i)) \leq v(r(i))$ *then* $d_i = 0$

   (b) *else* $d_i = v(l(i)) - v(r(i))$

3. *if* $p_i = 1$ *then*

   (a) *if* $v(l(i)) \geq v(r(i))$ *then* $d_i = 0$

   (b) *else* $d_i = v(l(i)) - v(r(i))$ □

The advantage of the vector of improving directions, as compared to the (anti-)gradient, is that it does not require any additional computations after the value $f(\bar{p})$ is computed and the optimal solution is found. A simple additional computation as in the definition above suffices to find out the vector of improving directions. The important properties of the cost function (2) and the role of the vector of improving directions are summarized in the following;

Lemma 3.3 *Let* $\bar{d}$ *be the vector of improving directions at the point* $\bar{p}$ *for the cost function* f*.*

1. *If* $\bar{d}$ *is* zero *then the point* $\bar{p}$ *is the global minimum of* f*.*

2. *If* $\bar{d}$ *is* nonzero *then the point* $\bar{p}$ *is not a local minimum of* f*; moreover, for any* $\lambda > 0$ *such that the point point* $\bar{p}' = \bar{p} + \lambda \cdot \bar{d}$ *belongs to the unit hypercube* $\mathcal{H}$*, one has* $f(\bar{p}') < f(\bar{p})$*.*

3. *If* $\bar{d}$ *is* nonzero *and* $\hat{d}$ *is any vector such that* $\bar{d}^T \cdot \hat{d} > 0$ [3] *and the point* $\bar{p}' = \bar{p} + \hat{d}$ *belongs to the unit hypercube* $\mathcal{H}$*, one has* $f(\bar{p}') < f(\bar{p})$*. (In fact, the vector* $\bar{d}$ *defines a whole cone of certain improvement directions from* $\bar{p}$ *— all directions that from an angle smaller than* 90^o*).*

4. *If the point* $\bar{p}$ *is the global minimum of* f *but not* integral[4] *then there is an integral global minimum* $\bar{p}'$ *with* $f(\bar{p}) = f(\bar{p}')$ *and all* $p'_i \in \{0, 1\}$

[3] We assume known and use the standard notation for scalar products, transpositions, etc. For example, the scalar product above means that $\bar{d}$ and $\hat{d}$ form an angle smaller than 90^o.

[4] I.e., some $p_i \notin \{0, 1\}$

Proof Sketch. 1) Follows from the definition of stability. 2) Suppose $\bar{d} \neq 0$ is the vector of improving directions at the point $\bar{p}$ and let i be such that $d_i \neq 0$ and $0 < p_i < 1$. Changing slightly $p_i := p_i + \lambda d_i$ diminishes the values of the right-hand sides of equations for *Min*-vertices in (2), which allows to diminish correspondingly values in left- and right-hand sides sides of constraints in (2), thus diminishing the target value $\sum_{i=1}^{n} v_i$. Other proofs are similar and will appear in the full paper.[5] □

3.3 Unimin and No-Plateau Properties

Lemma 3.3 implies two key properties of the cost functions, rendering them appropriate for local improvement optimization techniques.

Corollary 3.4 *For any SSG the associated cost function f defined by* (2) *possesses*

1. *the* unimin property*, i.e., its every local minimum is also the global minimum;*
2. *the* no-plateau property*, i.e., whenever the value of a function is constant in some neighborhood of a point, this point is a global minimum.* □

Therefore, none of the descent methods (which improve their values at every step) can be trapped in a local minimum or on a plateau, failing to find the global one. These are very favorable and important properties for iterative improvement optimization.

4 Optimization

In this section we describe several classes of methods appropriate for optimizing SSG-related costs (3), (2), based on Lemma 3.3.

4.1 Simplex-Like Methods

We first describe a class of Simplex-like methods, which form a very particular (even degenerate) subclass of our general interior-point method discussed below. In this case the interior-point flavor is immediately lost, the methods leave at once the interior, jump to the boundary, and make improvements

[5] ?????

traveling (making profitable pivots) between vertices (=pure strategies). Despite extensive effort, up until now nobody succeeded to come up with a polynomial pivot rules for the simplex-like methods even in the case of LINEAR PROGRAMMING. The best currently known simplex pivot rule for LP gives a randomized subexponential expected running time (Kalai 1992). There exists an analogous randomized subexponential algorithm for SSGs (Ludwig 1995), which proceeds by *single switches*. We feel it is necessary to keep connection with this particular case of the general interior point scheme we discuss below as practical and easy to implement approach.

Simplex-Like Algorithm(s).

1. Let $i = 0$ and start in some vector $\bar{p}^0$, describing an initial (probabilistic or stationary) strategy for Player 2.
2. Solve the LP $f(\bar{p}^i)$ defined by (2) and find the corresponding vector of improving directions $\bar{d}^i$ at the point $\bar{p}^i$.
3. If the vector of improving directions $\bar{d}^i$ is zero, terminate, the optimal strategy $\bar{p}^i$ is found (it may be non-integral, but could be always turned into an integral one; see Lemma 3.3).
4. **(Select Switches)** Otherwise, select a nonempty subset $S \backslash \{j_1, \ldots, j_m\}$ of nonzero components of the vector of improving directions $\bar{d}^i$ and make the *switches*:
   - $p_j^{i+1} = 1$ if $j \in S$ and $d_j^i > 0$
   - $p_j^{i+1} = 0$ if $j \in S$ and $d_j^i < 0$

   Leave the components in $Min \backslash S$ unchanged: $p_j^{i+1} = p_j^i$ if $j \notin S$
5. Set $i := i + 1$ and repeat from Step 2. □

It is easy to see, using Lemma 3.3, that the above algorithms (there is a whole family, one for each particular heuristic of choosing the set of switches) are correct and terminating:

Proposition 4.1 *For any choice of the switches set S in the Step 4 the Simplex-Like Algorithm above: 1) is a descent algorithm decreasing the value of the cost function $f(\bar{p}_i)$ at every iteration; 2) terminates after at most $2^{|Min|}$ iterations.* □

The last upper bound is a trivial one, since Player 2 has that many strategies, and none of them is repeated (the algorithm is descending). It is an open problem as to whether for any **(Select Switches)** strategy the Simplex-like algorithm above becomes *polynomial*[6] or just *subexponential.* Of special interest is the following particular case of the Simplex-like algorithm above:

- The *Greedy* one, making all profitable switches at every **(Select Switches)** step, i.e., $S = \{$ indices of all nonzero components of the vector of improving directions$\}$.

One member of the family above is the Randomized *single switch* algorithm due to Ludwig (1995). It assumes an edge e in the strategy, recursively finds an optimal strategy containing this edge, and if the resulting strategy (together with e) turns out non-optimal, makes a single profitable switch exchanging e for the alternative edge. Up to now, this is the only known *subexponential* algorithm for the problem.

4.2 Interior Point: (Sub)Gradient Methods

We now describe another class of gradient-descent-like methods that improve the cost functions by staying inside (and sometimes moving on the border) of the unit hypercube $\mathcal{H}$. Intuitively, always staying inside the hypercube leaves more degrees of freedom, allowing to move in all improving directions, whereas a restriction to the simplex-like moves between vertices along edges or faces is more restrictive. Speaking in terms of games, staying inside the hypercube (or moving between probabilistic strategies) means always keeping some (and amplifying the) influence of the optimal strategy in each iteration. This is not the case for the simplex-like methods which operate solely with pure non-probabilistic strategies.

Interior-point (sub)gradient-like methods start in some internal point $\bar{p}_0$ of the unit hypercube $\mathcal{H}$ and iteratively compute, for $k = 0, 1, \ldots$ successive approximations to the minimum of the cost function f:

$$\bar{p}^{k+1} = \bar{p}^k + \alpha^k \cdot \mathbf{d}^k,$$

where $\mathbf{d}^k$ is an antigradient-related vector (direction of improvement) and $\alpha^k > 0$ is a scalar step selected at iteration k. Usually, for smooth functions the direction $\mathbf{d}^k$ coincides with the antigradient

$$-\nabla f(\bar{p}^k) = (\ldots, -\frac{\partial f}{\partial x_i}(\bar{p}^k), \ldots)^T$$

[6] Recall that this same question remains open for the genuine Simplex algorithms (with different pivot rules) even for LINEAR PROGRAMMING.

, possibly multiplied by some positive definite symmetric matrix D^k (which results in a vector making an angle less that 90^o with antigradient and guarantees decrease of value). The stepsize α^k is selected in a way to keep the next iterate $\bar{p}^{k+1}$ inside the unit hypercube.

In our case, the antigradient $-\nabla f(\bar{p}^k)$ is not available in analytic form (moreover, SSG cost functions are not differentiable, as we will see later), but we have the following alternatives:

1. Using the vector of improving directions $\bar{d}^k$ at the point $\bar{p}^k$ instead (as described earlier).

2. Computing finite-difference approximations by the *forward difference* formula
$$\frac{\partial f(\bar{p}^k)}{\partial x_i} \approx \frac{1}{h}(f(\bar{p}^k + he_i) - f(\bar{p}^k)) \qquad (4)$$
or by the more accurate *central difference* formula
$$\frac{\partial f(\bar{p}^k)}{\partial x_i} \approx \frac{1}{2h}(f(\bar{p}^k + he_i) - f(\bar{p}^k - he_i)), \qquad (5)$$
where h is a small positive scalar and e_i is the i-th unit vector.

Note that (5) is more time consuming compared with the vector of improving directions, since it requires solving many LP problems at points $f(\bar{p}^k \pm he_i)$. However, it is more accurate, does not lead to additional overheads when the Hessian (second-order partial derivatives matrix is computed), and guarantees the faster rate of convergence, as we discuss later.

Different heuristics of choosing the improvement direction and stepsize at every step are possible and well-studied. They may lead to different rates of convergence. It is known that under certain quite general conditions many such choices lead to *linear or superlinear* rates of convergence near the optimum. However, some experimentation is necessary to make adjustments of the powerful gradient-related methods for SSGs.

4.3 Interior Point: Barrier Functions + Newton-Like Methods

Interior points methods work best (converge fastest) when a Newton-like method for unconditional optimization is combined with the barrier functions. In particular, the best currently known polynomial-time algorithms for many optimization problems are of this kind. In this section we develop the corresponding method for the cost functions associated to SSGs defined by (2).

The idea behind the method is pretty straightforward. Instead of solving a *conditional* optimization problem (3) for $f(\bar{p})$ (which, with a gradient-related method requires careful choices of the descent directions and stepsizes so as to always remain in the interior of the domain determined by constraints), in the *barrier method* we simply add to the cost function $f(\bar{p})$ defined by (2) a *barrier function* $B(\bar{p})$ defined in the interior of the feasible domain, the unit hypercube $\mathcal{H}$ in our particular case. The barrier function is continuous, convex, and goes to $+\infty$ as $\bar{p}$ approaches the boundaries of the feasible domain from inside. After that we may apply the fastest unconditional optimization method, which in most cases is Newton-like, as discussed below.

The most common barrier functions used in practice are *logarithmic* and *inverse* defined by

$$B(\bar{p}) = -\sum_{j=1}^{m} \ln(-g_j(\bar{p})), \tag{6}$$

$$B(\bar{p}) = -\sum_{j=1}^{m} \frac{1}{g_j(\bar{p})}, \tag{7}$$

where $g_j(\bar{p}) < 0$, for $j = 1, \ldots, m$ are constraints defining the feasible domain.

A particular barrier method is also determined by the choice of a *parameter sequence* $\{\varepsilon^k\}$ such that $0 < \varepsilon^{k+1} < \varepsilon^k$ and $\varepsilon^k \to 0$ as $k \to \infty$.

Now the barrier method consists in finding the sequence $\{\bar{p}^k\}$ defined by

$$\bar{p}^k = \operatorname{argmin}\big(f(\bar{p}) + \varepsilon^k \cdot B(\bar{p})\big) \tag{8}$$

using any unconditional optimization method, such as Newton's or alike. Note that the barrier term $\varepsilon^k \cdot B(\bar{p})$ tends to 0 for all interior points of the feasible domain as $\varepsilon^k \to 0$. Thus the barrier term becomes less and less significant, progressively allowing $\bar{p}^k$ to approach the boundary of the feasible set. (We know that for SSGs the minimum is attainable in a vertex of the feasible hypercube of strategies, i.e., in a stationary strategy.) Therefore, combining with Lemma 3.3, we have the following convergence (correctness) result:

Lemma 4.2 *Every limit point of a sequence $\{\bar{p}^k\}$ generated by a barrier method is a global minimum of the original constrained problem* (2). $\square$

4.4 Unconditional Optimization Techniques: Towards a Polynomial-Time Algorithm

In this section we describe several unconditional optimization techniques (or rather, a combination of those) we use to minimize (8). The basic technique is a Newton-like scheme with several adjustments. A *quasi-Newton method* is a gradient method of unconditional optimization with iteration of the form

$$\bar{p}^{k+1} = p^k - \alpha^k \cdot \bar{\mathbf{d}}^k, \tag{9}$$

where the direction of improvement $\bar{\mathbf{d}}^k$ is selected in a way to approximate the Newton direction

$$(\nabla^2 f(\bar{p}^k))^{-1} \cdot \nabla f(\bar{p}^k), \tag{10}$$

which provides for the fastest rate of convergence, but: 1) requires the existence and invertibility of the second-order partial derivatives Hessian matrix $\nabla^2 f(\bar{p}^k)) = (\dots \frac{\partial^2 f(\bar{p}^k)}{\partial x_i \partial x_j} \dots)$ and 2) leads to a considerable computation overhead at each iteration. Instead, we can use successive iterates $\bar{p}^k$ and gradients $\nabla f(\bar{p}^k)$ to find accurate approximations to the Hessian $\nabla^2 f(\bar{p}^k))$ with less computations. We omit the tedious details known from the nonlinear optimization literature.

Generally, Newton's method is the best since it converges *linearly* (i.e., with the rate of *geometrical progression* $c \cdot q^k$ with $c > 0$ and $0 < q < 1$) near the optimum, or even *superlinearly* (i.e., as $c \cdot q^{p^k}$ with $c > 0$ and $0 < q < 1 < p$). However, to improve its convergence rate far from the minimum in the initial iterations from a randomly chosen zeroth iterate, the following adjustment (combination with the steepest descent) should be made.

Suppose when we iterate (9) at point $\bar{p}^k$ the gradient $\nabla f(\bar{p}^k)$ and the Hessian matrix $\nabla^2 f(\bar{p}^k))$ both exist and the latter is *invertible*. Also the direction vector $\mathbf{d}^k$ is chosen equal to Newton's direction (10). Then one has, for some $c_1, c_2 > 0$, $p_1 > 2$, $p_2 > 1$:

$$\begin{aligned} c_1 \cdot \|\nabla f(\bar{p}^k)\|^{p_1} &\leq \nabla f(\bar{p}^k)^T \cdot \bar{\mathbf{d}}^k, \\ \|\bar{\mathbf{d}}^k\|^{p_2} &\leq c_2 \cdot \|\nabla f(\bar{p}^k)\|, \end{aligned}$$

Otherwise, select $\bar{\mathbf{d}}^k = D \cdot \nabla f(\bar{p}^k)$ for some fixed positive definite matrix D, or even choose a subgradient instead of gradient if the latter does not exist (we will see shortly that the SSGs cost functions are piece-wise smooth and may fail to possess gradients in certain points).

It is possible to show that the sequence of vectors $\bar{\mathbf{d}}^k$ is gradient-related, and the rate of convergence is *superlinear*. This may lead to a polynomial-time interior-point algorithm to SSGs. Possible problems however, are certain non-smoothness conditions, the validation of the approximations of Hessians, and the existence of *singular* points. These are minima $\bar{p}^*$ and directions $\bar{\mathbf{d}}$ such that the slope of the target function f along $\bar{\mathbf{d}}$, which is

$\nabla f(\bar{p}^* - \alpha \cdot \bar{\mathbf{d}})^T \cdot \bar{\mathbf{d}}$, changes very slowly or very rapidly, i.e., respectively, either

$$\lim_{\alpha \to 0} \frac{\nabla f(\bar{p}^* - \alpha \cdot \bar{\mathbf{d}})^T \cdot \bar{\mathbf{d}} - \nabla f(\bar{p}^*)^T \cdot \bar{\mathbf{d}}}{\alpha} = 0, \qquad \text{or} \tag{11}$$

$$\lim_{\alpha \to 0} \frac{\nabla f(\bar{p}^* - \alpha \cdot \bar{\mathbf{d}})^T \cdot \bar{\mathbf{d}} - \nabla f(\bar{p}^*)^T \cdot \bar{\mathbf{d}}}{\alpha} = \infty. \tag{12}$$

The question of existence and significance of such singular points in the context of cost functions for SSGs is currently open. On the one hand, such problems are generally known to be difficult in classical optimization, and the rate of convergence near singular points may be slower than linear (this leads to the loss of efficiency). On the other hand, we state the following

Conjecture 4.3 *The translation of PGs into SSGs allows, for any instance of a game, to bound away (non-uniformly) the limits* (11) *and* (12) *from* 0 *and* ∞ *respectively.* $\square$

If this is the case, then the methods described above provide for a *superlinear* rate of convergence in all circumstances. The latter would imply, in principle, the polynomial time algorithm for SSGs. Note, however, that this might mean only a theoretical superlinear convergence with the rate of $c \cdot q^{p^k}$ with $0 < q < 1$ and $p > 1$, but both very close to 1. We will report on progress along these lines in subsequent publications.

5 Further Properties of the Cost Functions

In the remainder of the paper we discuss further properties of the cost functions associated to SSGs and defined by (2), with the purpose to clarify the intuitions and show that technically the problems remaining may be difficult to solve completely, precisely, and efficiently.

5.1 Non-Convexity

In general, an SSG-associated cost function $f(p)$ is non-convex. This is usually dangerous, since non-convex functions may have isolated local minima, which may trap Newton- or gradient-related methods. Quite luckily, Lemma 3.3 shows that our SSGs-associated cost functions are always unimin, and the traps do not exist. However, non-convexity raises other problems with non-positive definitiveness of Hessians.

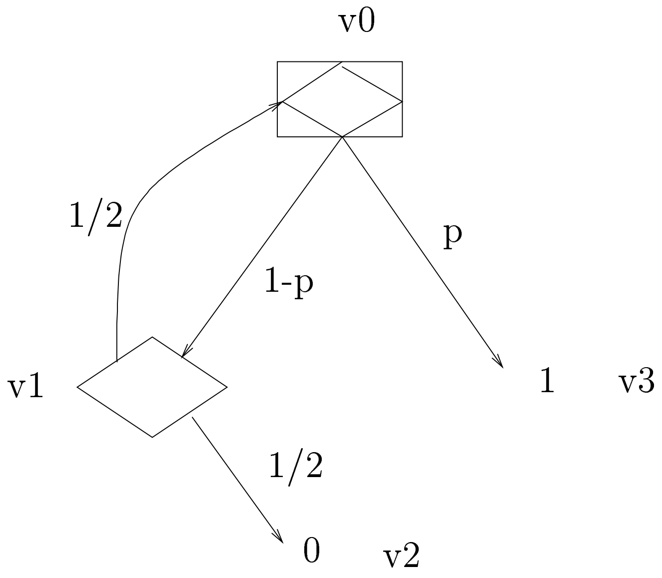

Figure 1: A game defining non-convex cost function

Consider the game in Figure 1 with a non-convex cost function defined, according to (2), as follows:

$$f(p) = \begin{cases} \min \; v_0 + v_1 + v_2 + v_3 & \\ \text{subject} \quad v_0 \;=\; p + (1-p)v_0 & \\ \qquad\qquad v_1 \;=\; 1/2v_0 & \\ \qquad\qquad v_2 \;=\; 0 & \\ \qquad\qquad v_3 \;=\; 1 & \end{cases} \tag{13}$$

Solving (13) explicitly for v_0, v_1 we get

$$\begin{aligned} v_0 &= 2p/(1+p) \\ v_1 &= p/(1+p) \\ f(p) &= 4 \;-\; 3/(p+1) \end{aligned}$$

From this solution and obviously from Figure 1, $f(p)$ is minimized by $p = 0$, which determines the optimal "go left" stationary strategy for the minimizing player (in the rectangular-diamond vertex v_0.) The graph of $f(p)$ given in Figure 2 shows that it is non-convex (it is the difference rather than a sum of two convex functions 4 and $3/(p+1)$.)

5.2 Non-Differentiable

In general, the target function $f(p)$ we associate to an SSG is not everywhere differentiable. As a drawback, we loose good properties guaranteed by

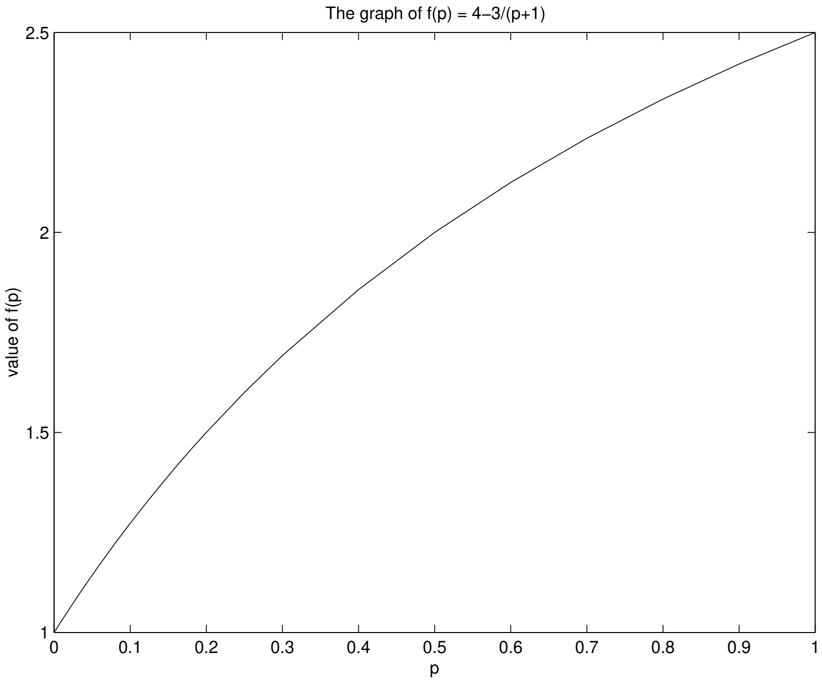

Figure 2: The graph for the non-convex target function of the game in Figure 1

smoothness, and have to adjust everything by using finite-difference approximations, as (4), (5) (and similarly for the Hessians). On the other hand, we do not have the cost function given in analytic form anyway, just as a polynomial-time solvable LP (2). Therefore, using difference approximations is quite adequate and does not raise practical problems. The precise rate of convergence analysis, however, becomes more complicated.

Consider the game in Figure 3 with a non-differential cost function (2) defined as follows:

$$f(p) = \begin{cases} \min \ v_0 + v_1 + v_2 + 1 & \\ \text{subject} \ \ v_0 \ \geq \ v_1 & \\ \qquad\qquad v_0 \ \geq \ v_2 & \\ \qquad\qquad v_1 \ = \ 1/2 & \\ \qquad\qquad v_2 \ = \ p & \end{cases} \tag{14}$$

Solving (14) explicitly for v_0, $f(p)$ we get

$$\begin{aligned} v_0 &= \begin{cases} p & , \ p > 1/2 \\ 1/2 & , \ p \leq 1/2 \end{cases} \\ f(p) &= \begin{cases} p + 1/2 + p + 1 = 2p + 3/2 & , \ p > 1/2 \\ 1/2 + 1/2 + p + 1 = p + 2 & , \ p \leq 1/2 \end{cases} \end{aligned}$$

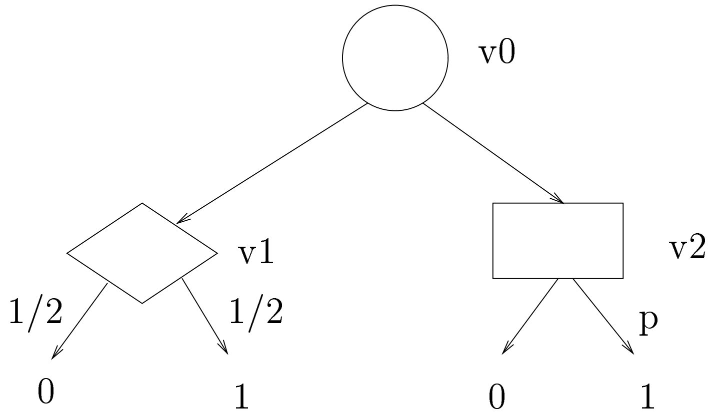

Figure 3: A game defining non-convex target function

From this solution and obviously from Figure 3, $f(p)$ is determined by the left switch of the round player when $p \leq 1/2$, and the right switch when $p > 1/2$, which determines a non-differentiable behavior of $f(p)$ in $p = 1/2$. The graph of $f(p)$ given in Figure 4 shows that it is non-differentiable, and the optimal strategy of the square-diamond player is determined by $p = 0$, i.e., "go left" in v_2.

5.3 Ill-Conditioned Behavior

Let us analyze the behavior of the cost function for the game presented in Figure 5 and described by the following LP:

$$\begin{aligned} \min \quad & v_0 + v_1 + v_2 + v_3 \\ v_0 &= p + (1-p)v_0 \\ v_1 &= qv_0 \\ v_2 &= 0 \\ v_3 &= 1 \end{aligned}$$

Let us define the following probabilities for a vertex v:

- $p_L(v)$ be the probability that the game from v returns to v.
- $p_0(v)$ be the probability that the game from v ends up in the 0-sink without revisiting v.
- $p_1(v)$ be the probability that the game from v ends up in the 1-sink without revisiting v.

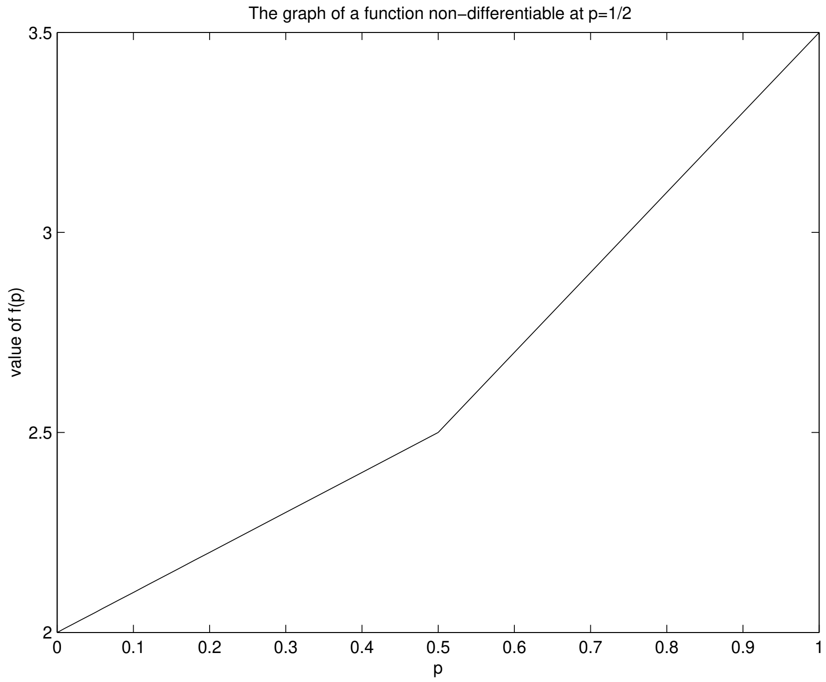

Figure 4: The graph for the non-differentiable target function of the game in Figure 3

We will omit parentheses if it is clear what vertex is referred to. Since one of three things must happen, and they are disjoint, we have $p_0 + p_1 + p_L = 1$. Now the value of a vertex v can be expressed in those probabilities as follows:

$$v = p_1 + p_L * p_1 + (p_L)^2 * p_1 + ... = p_1 * \sum_{i=0} (p_L)^i,$$

which simplifies to

$$v = p_1 * \sum_{i=0} (p_L)^i = \frac{p_1}{1 - p_L} = \frac{p_1}{p_0 + p_1}.$$

Now consider the value of the vertex v_0 in the game in Figure 5. We have $p_0 = (1-p)*(1-q)$, $p_1 = p$, and $p_L = (1-p)*q$, which gives:

$$v_0 = \frac{p}{(1-p)(1-q)+p}.$$

The ill-conditioned behavior of this function, depending on the probabilities p and q (when p is small and q is big) is shown in Figure 6. Here we have singularities of both kinds (11) and (12). Therefore, the adjustments described in Section 4.4 are really necessary in this case.

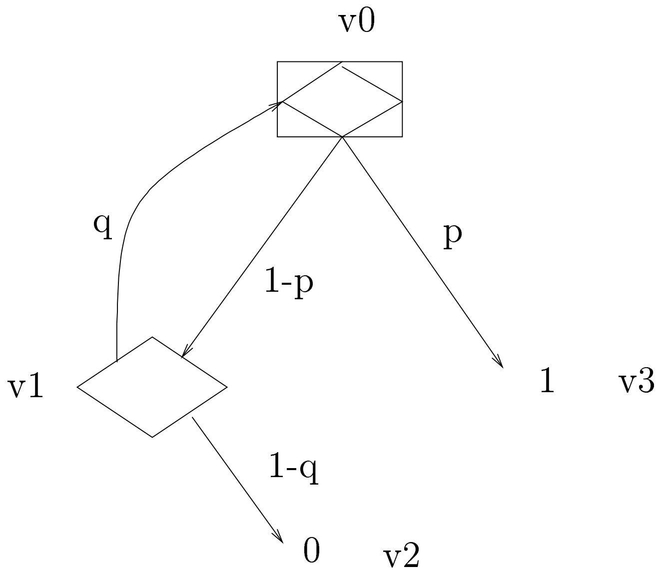

Figure 5: Game illustrating ill-conditioned behavior

6 Conclusions

In this paper we suggested, for the first time, to use continuous interior-point optimization techniques based on logarithmic barrier functions and quasi-Newton unconditional optimization methods for the Parity Games (PGs) and Simple Stochastic Games (SSGs) problems. Both problems are well-known and recognized as difficult, with NP$\cap$coNP-membership, open PTIME-status, and the best currently known algorithms being exponential, with the exception of (Ludwig 1995, Petersson & Vorobyov 2001), which are *subexponential*.

At the same time the interior-point methods are currently the best polynomial-time methods of Linear Programming (LP), and we hope that pursuing the interior-point ideas for the cases of PGs and SSGs will lead to practically best and (hopefully) theoretically polynomial algorithms.

The novelty of our approach consists in:

1. Extending the cost functions associated to SSGs to *probabilistic strategies* (in contrast to pure strategies exclusively considered before in the context of both PGs and SSGs), thus turning the problem into the one rendering itself to the interior-point optimization.

2. Expressing the cost functions associated to SSGs using a parameterized Linear Program (via an extension of the old Derman's theorem).

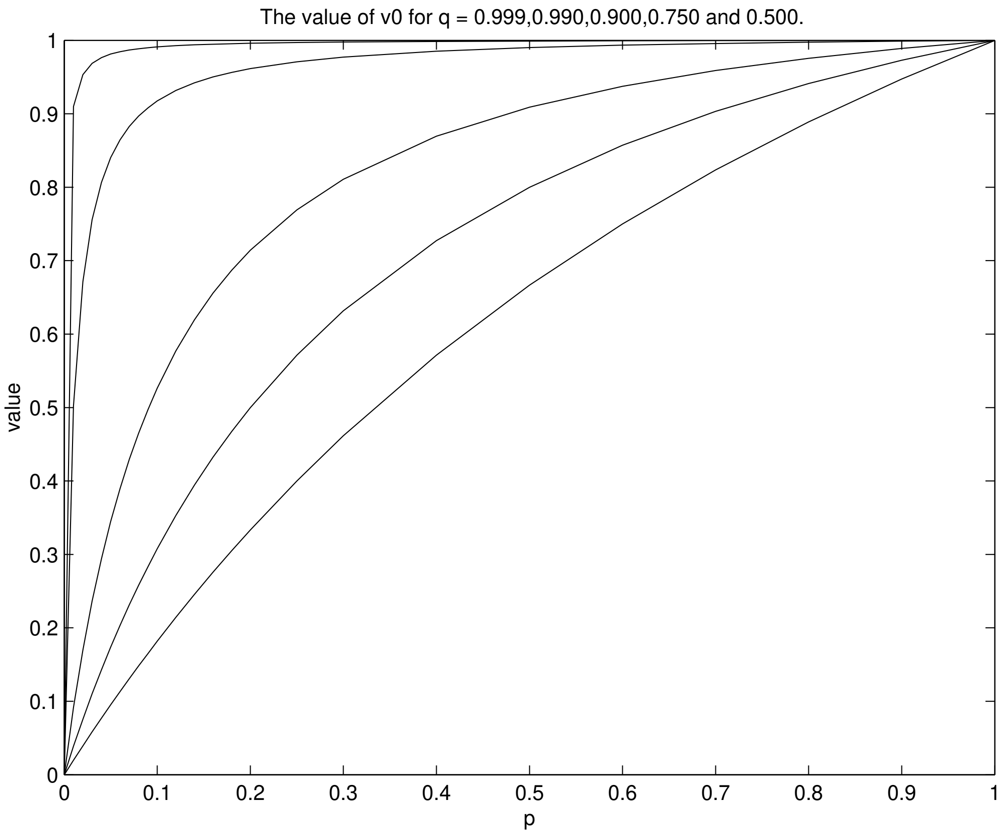

Figure 6: The graph for the ill-conditioned cost function of the game presented in Figure 5

3. Investigating favorable properties, as continuity and unimin of the cost functions, making them appropriate for global optimization techniques.

4. Suggesting feasible interior-point optimization schemes for those functions.

5. Making a preliminary rate of convergence analysis of the proposed techniques.

6. Analyzing potential technical difficulties with implementation and singularities.

The new framework is now settled and motivated. The problems for future research may be summarized as follows.

1. Collect practical experimental data, by running prototypes of the suggested algorithms for randomly generated or known hard instances of

PGs and SSGs. This may allow for selecting some techniques, different gradient- or quasi-Newton, in favor of others. It is well known that the interior-point techniques are superior in practice to ellipsoid-like, although in theory do not provide for the better worst-case bounds. We plan and hope to get similar evidence in favor of our interior-point techniques for PGs and SSGs, as compared with the previous traditional algorithms, and even *subexponential* ones. The major problem here is the absence of ready-to-use implementations of the previous algorithms and documented hard instances of problems.

2. Analyze the influence of non-smoothness of the target functions on the accuracy of approximations and the rate of convergence. It should also be pointed out that our methods are approximative (rather than precise), although getting a solution with some controlled tolerance is completely satisfactory for games; see, e.g., (Zwick & Paterson 1996).

3. Analyze possibilities and implications of existence of *singular* problems and investigate the related rates of convergence.

A considerable further effort is needed to succeed, and we will report on the progress along these lines in future publications.

References

Browne, A., Clarke, E. M., Jha, S., Long, D. E. & Marrero, W. (1997), 'An improved algorithm for the evaluation of fixpoint expressions', *Theor. Comput. Sci.* **178**, 237–255. Preliminary version in CAV'94, LNCS'818.

Derman, C. (1970), *Finite-State Markovian Decision Processes*, Academic Press.

Emerson, E. A. (1997), Model checking and the Mu-calculus, *in* N. Immerman & P. G. Kolaitis, eds, 'DIMACS Series in Discrete Mathematics', Vol. 31, pp. 185–214.

Emerson, E. A., Jutla, C. & Sistla, A. P. (1993), On model-checking for fragments of μ-calculus, *in* 'Computer Aided Verification, Proc. 5th Int. Conference', Vol. 697, Lect. Notes Comput. Sci., pp. 385–396.

Emerson, E. A. & Lei, C.-L. (1986), Efficient model checking in fragments of the propositional Mu-calculus, *in* '1st Annual IEEE Symp. on Logic in Computer Science LICS'86', pp. 267–278.

Jurdziński, M. (1998), 'Deciding the winner in parity games is in UP∩co-UP', *Information Processing Letters* **68**, 119–124.

Jurdzinski, M. (2000), Small progress measures for solving parity games, *in* '17th STACS', Vol. 1770 of *Lect. Notes Comput. Sci.*, Springer-Verlag, pp. 290–301.

Kalai, G. (1992), A subexponential randomized simplex algorithm, *in* '24th ACM STOC', pp. 475–482.

Karmarkar, N. K. (1984), 'A new polynomial-time algorithm for linear programming', *Combinatorica* **4**, 373–395.

Ludwig, W. (1995), 'A subexponential randomized algorithm for the simple stochastic game problem', *Information and Computation* **117**, 151–155.

Nash, S. & Sofer, A. (1996), *Linear and Nonlinear Programming*, McGraw Hill.

Petersson, V. & Vorobyov, S. (2001), 'A randomized subexponential algorithm for parity games', *Nordic Journal of Computing* . To appear. Preliminary version: Technical Report #1, 2001, Department of Information Technology, Uppsala Unitersity. Also: Proceedings of the 12th Nordic Workshop on Programming Theory.

Puri, A. (1995), Theory of hybrid systems and discrete events systems, PhD thesis, EECS Univ. Berkeley.

Seidl, H. (1996), 'Fast and simple nested fixpoints', *Information Processing Letters* **59**(3), 303–308.

Vorobyov, S. (2001), 'Better decision algorithms for parity games and the μ-calculus model checking', *Information Processing Letters* . To appear.

Zwick, U. & Paterson, M. (1996), 'The complexity of mean payoff games on graphs', *Theor. Comput. Sci.* **158**, 343–359.