A Discrete Subexponential Algorithm for Parity Games ⋆ ⋆⋆

Henrik Björklund, Sven Sandberg, and Sergei Vorobyov

Information Technology Department, Uppsala University

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

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

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

1 Introduction

Parity games are infinite games played on finite directed leafless graphs, with vertices colored by integers. Two players move a pebble along edges of the graph. Vertices are partitioned between the players, and the owner of the vertex currenly visited by the pebble decides where to move it next by selecting an outgoing edge. The goal of Player 0 is to ensure that the biggest color visited by the pebble infinitely often is even, whereas Player 1 tries to make it odd. The complexity of determining a winner in parity games, equivalent to the Rabin chain tree automata non-emptiness, as well as to the μ-calculus model checking [10,8], is a fundamental open problem in complexity theory [19]. The problem belongs to NP∩coNP, but its PTIME-membership status remains widely open. All known algorithms for the problem are exponential, with an exception of [20] when the number of colors is large and games are binary.[1]

In this paper we present a new discrete, randomized, *subexponential* algorithm for parity games. It combines ideas of iterative strategy improvement, randomization techniques of Kalai [15] for Linear Programming and of Ludwig [16] for simple stochastic games, with discrete strategy evaluation similar to Vöge and Jurdziński [24]. Generally, algorithms for parity games are *exponential* in the number of colors k, which may be as big as the number n of vertices. For most, exponentially hard input instances are known [9,8,7,22,14]. Our algorithm is *subexponential* in n. Earlier we suggested a subexponential algorithm [20], similar to [16], but based on graph optimization rather than linear programming subroutines. Both algorithms [16,20] become exponential in n for graphs with *unbounded* vertex outdegree. The present paper eliminates this drawback. There is a well-known reduction from parity to mean payoff games, but the best known algorithms for the latter [13,26,21] are known to be exponential (pseudopolynomial).

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

⋆⋆ This is an extended version of [4].

[1] More precisely, when the total number of edges for one of the player is subquadratic in the number of his vertices.

Reducing parity to simple stochastic games [26] leads to manipulating high-precision arithmetic and to algorithms invariably subexponential in the number of vertices, which is worse than an exponential dependence on colors when colors are few.

A recent iterative strategy improvement algorithm by Vöge and Jurdziński [24] uses a discrete strategy evaluation involving game graph characteristics like colors, sets of vertices, and path lengths. Despite a reportedly good practical behavior, the only known worst-case bound for this algorithm is exponential in the number of vertices n, independently of the number of colors. This is no better than searching the full strategy space.

Our new algorithm avoids any reductions and applies directly to parity games of *arbitrary* outdegree. We use a discrete strategy evaluation measure similar to, but more economical than the one used in [24]. The longest possible improving sequence of strategies with our measure is exponential in the number of colors, shorter in comparison with exponential in the number of vertices in [24]. Combined with Kalai's and Ludwig's randomization schemes this limits the number of improvement steps to be: 1) at most subexponential in the number of vertices n and, simultaneously, 2) at most exponential in the number of colors k. The first bound is an advantage over preceding algorithms, when there are many colors, $k = \Omega(n^{1/2+\varepsilon})$. The second is better than the first when colors are few.

Recently [6,1], we discovered that another subexponential randomization scheme for linear programming suggested by Matoušek-Sharir-Welzl [17,18] at the same time as Kalai [15], can be adapted to parity games, and also gives an algorithm with similar bounds. Similarly, we show in [1] that Gärtner's approach [11] with abstract optimization problems yields yet another algorithm for parity games with similar bounds. A different subexponential algorithm for parity games is described in [5].

Outline. After preliminaries in Section 2 on parity games, Section 3 presents a simple, Ludwig-style randomized algorithm, relying upon an abstract 'quality' measure on strategies with certain desirable properties. Section 4 gives all the details of the measure definition. The important properties of the measure are proved in Section 5 and 6. Section 5 provides a crucial characterization of local improvement in the measure. Section 6 shows that a strategy that cannot improve locally is optimal. Section 7 sets an upper bound on the number of strategy improvement steps, and Section 8 describes a polynomial time algorithm to find optimal strategies in a *one-player* game, an important step in computing strategy values. Section 9 adapts Kalai's randomization scheme for linear programming to the parity game setting, providing a better subexponential bound for games on graphs with unbounded outdegree. Section 10 concludes and sketches directions for future research.

2 Parity Games

We use a conventional definition of infinite duration parity games on *finite* graphs. We only consider finite explicitly represented graphs, because we are concerned with the complexity of decision algorithms.

Definition 1 (Parity Games).
A parity game *is an infinite game played on a finite directed leafless graph* $G = (V_0, V_1, E, k, c)$*, where:*

- V_0*,* V_1 *form a partition of the graph vertices;*

– $E \subseteq (V_0 \cup V_1)^2$ *is the set of edges, and every vertex has at least one outgoing edge (there are no* leaves *or* sinks*);*
– $k \in \mathbb{N}$*, and* $c : V_0 \cup V_1 \to \{1, \ldots, k\}$ *is a coloring function.*

Starting from a vertex, the players construct an infinite path called a play*. Player* i *moves from a vertex in* V_i *by selecting one of its successors. The largest vertex color* j *occurring infinitely often in a play determines the winner: Player 0 wins if* j *is* even*; Player 1 wins if* j *is* odd*.* □

Convention 1 *Throughout the paper we systematically use notation introduced in Definition 1. We also use* $n_0 = |V_0|$*,* $n_1 = |V_1|$*, and* $n = n_0 + n_1$*.*

Positional strategies are fundamental for parity games.

Definition 2 (Positional Strategy). *A* positional strategy *for Player 0 is a function* $\sigma : V_0 \to V_0 \cup V_1$*, such that if* $\sigma(v) = v'$*, then* $(v, v') \in E$*. Saying that Player 0* fixes *his positional strategy means that he deterministically chooses the successor* $\sigma(v)$ *each time the play comes to* v*, independently of the history of the play. Positional strategies for Player 1 are defined symmetrically.* □

Parity games are known to be *determined* and solvable in positional strategies. This means that the vertices of any parity game can be partitioned into winning sets W_0, W_1 of Players 0 and 1 such that each player has a positional strategy that wins any play that starts from a vertex in his winning set [10]. Therefore, the players can restrict themselves to positional strategies, and it is no disadvantage to reveal the strategy to the opponent in advance. Consequently, in the sequel by 'strategy' we will always mean 'positional strategy'.

The problem we are addressing in this paper is to find the winning sets and optimal strategies of both players. Given an optimal strategy of one player, the optimal strategy of the other player and the winning sets can be computed in polynomial time.

All algorithms described in this paper are *iterative strategy improvement algorithms*, proceeding from a current trial positional strategy to another one that is 'better', until an optimal strategy is reached. They are based on the simplest possible, *single* switches, when a positional strategy is changed in one vertex at a time.

Definition 3 (Single Switch). *A* single switch *in a positional strategy of Player 0 is a change of successor assignment in exactly one vertex.* □

We use the adjective *single* to make a distinction with a *multiple* switch, when a strategy is simultaneously changed in several vertices. We consider multiple switch algorithms in [3,6].

In Section 4 we introduce an appropriate evaluation function on positional strategies, and in Sections 5 – 8 demonstrate that iteratively improving a strategy by single switches yields an optimal strategy.

Definition 4 (Subgames). *A* subgame *is obtained from a parity game by throwing away some edges, without creating sinks.*

Let G *be a parity game and* σ *a positional strategy. The subgame* G_σ *is obtained from* G *by deleting all edges going out from vertices in* V_0 *not selected by* σ*.* □

We will first present an algorithm for binary parity games.

Definition 5 (Binary Parity Game). *A* binary *parity game is a game where the vertex outdegree is at most two.* □

3 Ludwig-Style Algorithm with a Well-Behaved Measure

To motivate further developments we start with the conceptually easier case of binary games, describe an abstract version of the algorithm, and state the requirements needed for its correctness and efficiency. Subsequent sections implement technical machinery to meet these requirements.

Every positional strategy of Player 0 in a binary parity game can be associated with a corner of the n_0-dimensional Boolean hypercube $\mathcal{H} = \{0,1\}^{n_0}$. Suppose there is a way of assigning 'values' to strategies allowing for comparing neighbor strategies in $\mathcal{H}$ from the viewpoint of Player 0. Then we can run any variant of the local search iterative improvement algorithm, e.g., randomly moving to one of the better neighbors at each iteration. Unfortunately, no subexponential upper bounds are currently known for this algorithm [23,3,2,25].

Algorithm 1 below presents a different, randomized, *subexponential* iterative improvement algorithm to find the best strategy in a binary parity game. It is similar to the one suggested by Ludwig [16] for simple stochastic games. The extension to non-binary games, based on a related randomization scheme due to Kalai [15] is described in Section 9. Other randomized algorithms for parity games are described in [6,1,5].

Algorithm 1: Ludwig-Style Algorithm for Parity Games

```
LUDWIG(Hypercube H, initial vertex v_0 ∈ H)
(1)   if dimension(H) = 0
(2)       return v_0
(3)   choose a random facet F of H containing v_0
(4)   v* ← Ludwig(F, v_0)
(5)   if neighbor u of v* on H \ F (opposite facet) is better than v*
(6)       return Ludwig(H \ F, u)                    //switch to opposite facet
(7)   else
(8)       return v*
```

To guarantee correctness and subexponential running time of Algorithm 1, the assignment of values to strategies cannot be completely unstructured. In subsequent sections we present a strategy evaluation function that given a strategy returns a value. The function meets the following criteria, where the $\prec$ relation is a partial order on strategies.

Local optima are global. A strategy is a *local optimum* if changing its choice in any single vertex (i.e., moving along edges of $\mathcal{H}$) does not give an improvement of the value. Every locally optimal strategy should also be globally optimal, i.e., at least as good as any other strategy, with respect to $\prec$. This property is necessary to guarantee that whenever the algorithm terminates, the resulting strategy has optimal value.

Optimal corresponds to winning. Any strategy that is globally $\prec$-optimal wins in all vertices of the winning set of Player 0. This property ensures that the strategy returned by the algorithm actually solves the game.

Values in subgames are identical. If σ is a strategy in a subgame G' of G where *only choices of Player 0* are removed, then the value of σ is the same in G and G'. This property holds once choices of Player 0 not taken by the current strategy do not affect the value.

Evaluating strategies is costly, however, so it should be avoided if possible. Imagine, we were to evaluate all the neighbors (different in one vertex) of a current strategy at every iteration step. This would lead to a considerable computational overhead. The approach we use is more efficient. Our 'strategy quality' measure is a tuple consisting of one value per vertex. Vertex values are linearly ordered, and strategy values are partially ordered by comparing the vertex values componentwise. This allows us to reason about *attractive switches*. A *switch* (changing the strategy in a single vertex, by selecting an alternative successor of this vertex) is *attractive* if it selects a successor with a better value with respect to *the current strategy*. The following properties of attractive switches guide the iterative improvement, guarantee correctness, and allow us to save on the number of strategy evaluations.

Profitability of attractive switches. Let σ be a strategy and v a vertex. If the value under σ of the successor $\sigma(v)$ of v is worse than the value of v's other successor (the switch to this successor is *attractive*), then changing σ in v to this other successor results in a better strategy (the switch is *profitable*). This property (attractive implies profitable) ensures that the test in line (5) of Algorithm 1 can be performed without additional strategy evaluations, and is proved for our evaluation function as Theorem 3.

Optimality of stable strategies. If a strategy is *stable*, i.e., has no attractive switches, then it is globally optimal and also winning. Consequently, the algorithm can terminate as soon as it finds a stable strategy, without evaluating its neighbors. For our evaluation function, optimality of stable strategies is shown in Theorem 4 and Corollary 1.

Together, these two properties imply that we can let the iterative improvement be guided by attractiveness of switches. As long as there are attractive switches, we can make them, knowing that they are profitable. As soon as there are no attractive switches, we know that the current strategy is globally optimal. Actually, we show in Sections 5 and 6 that with our measure, switches are attractive if and only if they are profitable. It follows that any strategy without profitable switches is also stable, and thus globally optimal, and we get the property that all local optima are global. Additionally, relying on attractiveness, rather than investigating neighbors explicitly, does not change the behavior of our algorithms, only makes them more efficient. The Ludwig-style Algorithm 1 together with an evaluation function satisfying the above properties applies to solving binary parity games.

Ludwig [16] shows that his algorithm for simple stochastic games has a $2^{O(\sqrt{n_0})}$ upper bound on the expected number of improvement steps. With only minor modifications, the same proof shows that the Ludwig-style algorithm together with our strategy evaluation function has the same bound for parity games [3].

As shown in Section 7, the value space of the strategy evaluation function we use allows for at most $O(n^3 \cdot (n/k+1)^k)$ improvement steps. Since the algorithm only makes improving switches, the upper bound on the number of switches of the combined approach is

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

This is better than any previous algorithms (exponential in k) for binary parity games whenever $k = \Omega(n_0^{1/2+\varepsilon})$ for $\varepsilon > 0$. When k is small, the first term in the bound makes the algorithm comparable with the previous algorithms exponential in k. Note that the algorithms in [20] and [24] have the bounds $2^{O(\sqrt{n_0})}$ and $poly \cdot 2^{n_0}$ respectively.

Any parity game reduces to a binary one. This allows for a subexponential algorithm for games with *subquadratic* total vertex outdegree. For arbitrary games the reduction may give a *quadratic blow-up* in the number of vertices and the Ludwig-style algorithm becomes *exponential* in n_0. In Section 9 we achieve a subexponential bound by employing a more involved randomization scheme due to Kalai [15].

4 Values, Comparisons, Attractive Switches

In this section we define the 'value' of a strategy — the target iteratively improved by our algorithms, until a local optimum is found (which will also be proved global). Given a strategy σ of Player 0, its value is a vector of values of all vertices of the game. The value of each vertex is computed with respect to the pair of strategies (σ, τ) of Players 0 and 1, respectively, where τ is an '*optimal*' response counterstrategy against σ. The optimality of τ is essential for guiding Player 0 in improving σ. We delay the issue of constructing optimal counterstrategies until Section 8, assuming for now that Player 1 always responds with an optimal counterstrategy.

4.1 Vertex Values

For technical reasons to be explained shortly, each vertex is assigned a unique value, called a *tint*.

Definition 6 (Tints). *A bijection* $\mathbf{t} : V \to \{1, \ldots, n\}$ *such that*

$$c(u) \leq c(v) \Rightarrow \mathbf{t}(u) \leq \mathbf{t}(v)$$

assigns tints to vertices. The color of a tint $s \in \{1, \ldots, n\}$ *equals* $c(\mathbf{t}^{-1}(s))$. □

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

Definition 7 (Winning and Losing Colors and Tints). *Even colors are called* winning for Player 0 *and odd colors are* winning for Player 1. *Tint* t *is* winning for Player i *if its color* $c(t)$ *is winning for Player* i. *A color or tint is* losing *for a player if it is winning for his adversary.* □

Note that tints of different colors are ordered as these colors. In Section 4.2 we will define that within the same winning (resp. losing) color the bigger (resp. smaller) tint is better for Player 0.

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

Notation 2 Denote by V^i the set of vertices of *color* i and by $V^{>t}$ the set of vertices with *tints* numerically bigger than t. □

Definition 8 (Trace, Loop Major). *Suppose the players fix positional strategies σ and τ, respectively (not necessarily optimal). Then from every vertex u_0 the* trace of the play *is the sequence of vertices visited, up to the first repetition. It takes a δ-shape form: an initial simple path (of length $q \geq 0$) ending in a loop of length $s - q$:*

$$u_0, \dots u_q, \dots u_s = u_q, \tag{1}$$

where all vertices u_i are distinct, except $u_q = u_s$. The vertex of maximal tint on the loop is called the loop major. □

Definition 9 (Vertex Values and Path Values). *Consider the trace of a play $u_0, \dots, u_q, \dots, u_s = u_q$ determined by a pair of positional strategies σ and τ from vertex u_0 as in Definition 8. Let u_r, with $r \in \{q, \dots, s-1\}$, be the loop major. The* value *$\nu_{\sigma,\tau}(u_0)$ of u_0 with respect to a pair of positional strategies σ and τ has the form (t, P, p) and consists of:*

Loop Value (Tint) *t equal to the tint $\mathbf{t}(u_r)$ of the loop major.*
Path Color Hit Record Relative to t *defined as a vector*

$$P = (m_k, m_{k-1}, \dots, m_j, \underbrace{0, \dots, 0}_{j-1 \text{ times}}),$$

where $j = c(t)$ is the color of the major tint t, and

$$m_i = \left|\{u_0, u_1, \dots, u_r\} \cap V^i \cap V^{>t}\right|$$

is the number of vertices of color $i \geq j$ on the path beginning in u_0 and ending the first time it hits the the loop major (except that for the color l of the major we account only for the vertices with tints bigger than t.)
Path Length $p = r$.

Call the pair (P, p) the path value. □

Note that the path value is $(\overline{0}, 0)$ for the loop major and that the color hit record is $\overline{0}$ for all vertices on the loop.

The reason of the complexity of this definition is to meet the criteria enumerated in Section 3 and simultaneously obtain the 'tightest possible' bound on the number of iterative improvements. Theorem 5 settles such a bound for the measure from Definition 9, namely $O(n_0 n^2 \cdot (n/k+1)^k)$.

Vöge and Jurdziński [24] assign similar values to vertices with respect to a pair of strategies. The difference lies in the definition of path color hit records. The evaluation in [24] records the set of all vertices with bigger tints than the loop major, instead of recording the number of vertices of each color. The benefit of our modification is that instead of 2^n possible path color hit records, we get at most $(n/k+1)^k$; see Theorem 5.

4.2 Value Comparison

Now we proceed to comparison of vertex and path values. Later this allows us to define what it means for a strategy to be 'better' than another, with respect to the measure. In the sequel, when saying 'attractive', 'better', 'worse', 'profitable', etc., we consistently take the viewpoint of Player 0.

Definition 10 (Preference Orders). *The* preference order *on colors (as seen by Player 0) is defined as follows:*

$$c \prec c' \text{ iff } (-1)^c \cdot c < (-1)^{c'} \cdot c'.$$

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

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

□

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

Definition 11 ('Lexicographic' Ordering). *Given two vectors (indexed in descending order from the maximal color k to some $l \geq 1$)*

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

define $P \prec P'$ if the vector

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

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

This ordering is consistent with what we consider 'better' for Player 0.

Definition 12 (Path Value Comparison). *For two vertex values (t, P_1, p_1) and (t, P_2, p_2), where t is a tint, $j = c(t)$ is its color, and*

$$P_1 = (m_k, m_{k-1}, \ldots, m_{j+1}, m_j, \ldots, m_1),$$
$$P_2 = (m'_k, m'_{k-1}, \ldots, m'_{j+1}, m'_j, \ldots, m'_1),$$

say that $(P_1, p_1) \prec_t (P_2, p_2)$ (pronounced: for Player 0 path value (P_2, p_2) is more attractive modulo t *than the path value (P_1, p_1)), if:*

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

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

Remark 1. Note that (2) means that shorter (resp. longer) paths are better for a player when the loop tint t is winning (resp. losing) for him.

Definition 13 (Vertex Value Comparison). *For two vertex values define $(t_1, P_1, p_1) \prec (t_2, P_2, p_2)$ if*

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

Definition 14 (Vertex Values). *The value $\nu_\sigma(v)$ of a vertex v with respect to a strategy σ of Player 0 is the minimum of the values $\nu_{\sigma,\tau}(v)$, taken over all strategies τ of Player 1.* □

In Section 8 we show that the 'minimum' in this definition can be achieved in all vertices simultaneously by a positional strategy τ of Player 1.

Definition 15 (Strategy Value). *The* value of a strategy σ *of Player 0 is a vector of values of all vertices with respect to the pair of strategies (σ, τ), where τ is an optimal response counterstrategy of Player 1 against σ; see Section 8.* □

It turns out, however, that two strategies may have equal values in this sense, but one is 'better' than the other. We therefore define the order on strategies in a slightly more refined way as follows.

Definition 16 (Strategy Comparison). *A strategy σ'* improves *σ, symbolically $\sigma \prec \sigma'$, if:*

1. *$\nu_\sigma(v) \preceq \nu_{\sigma'}(v)$ for all vertices v, and*
2. *either a) there is at least one vertex u with $\nu_\sigma(u) \prec \nu_{\sigma'}(u)$ or b) there is at least one vertex that improves the value of a successor, i.e., $\nu_\sigma(\sigma(u)) \prec \nu_{\sigma'}(\sigma'(u))$.* □

We conclude this section with a simple but important property.

Proposition 1 (Transitivity). *The relations $\prec$ on colors, tints, vertex values, strategies, and $\prec_t$ on path values (for each t) are transitive.* □

4.3 Attractive Switches

Our algorithms proceed by single attractive switches only.

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

Remark 2. Note that deciding whether a switch is attractive (when comparing values of its successors) we do not directly account for the color/tint of the current vertex. However, this color/tint may be included in the values of successors possibly dependent on the current vertex.

From the definition of attractiveness it is not obvious that making attractice switches always provides for a value improvement and guarantees termination. Indeed, after making an attractive switch the strategy changes and the values of vertices are recomputed with respect to this new strategy and an optimal Player 1 counterstrategy. However, in Section 5 we introduce the notion of profitability, guaranteeing that the new strategy is better in the sense of Definition 16 and show profitability of attractive switches (actually, equivalence of attractiveness and profitability). In Section 7 we bound the maximal possible number of attactive/profitable switches.

The algorithm terminates once attaining a strategy without attractive switches.

Definition 18 (Stable Strategy). *A strategy σ of Player 0 is* stable *if it does not have attractive switches with respect to $\tau(\sigma)$, an optimal counterstrategy of Player 1.* □

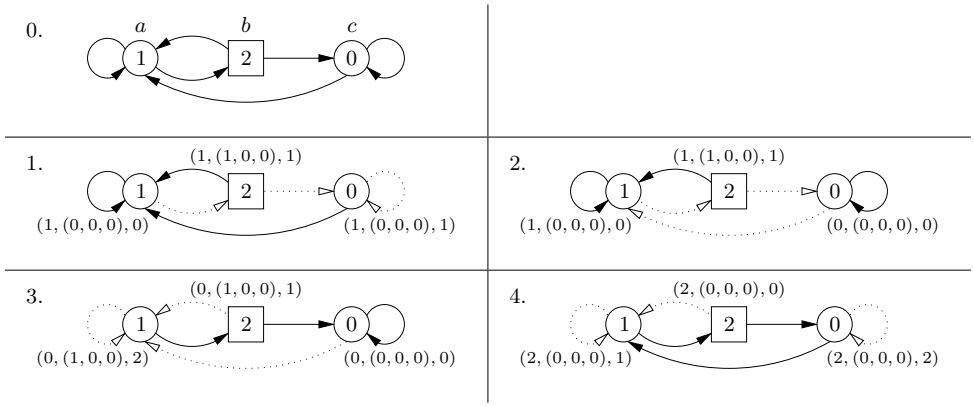

Fig. 1. An example run of a strategy improvement algorithm.

In Section 8 we show that all optimal counterstrategies of Player 1 provide for the *same* values. Thus stability of σ in the previous definition may be checked after computing any optimal counterstrategy $\tau(\sigma)$.

Stability is another term to characterize a locally optimal strategy (with respect to attractiveness).

Example 1. To illustrate the computation of the measure and the general flavor of strategy improvement algorithms, Figure 1 shows an example parity game with a sequence of improving strategies.

0. The parity game. Player 0 owns vertices a and c, and Player 1 owns vertex b. In this particular game, there are three colors and three tints, and they happen to coincide. We will now show a run of a strategy improvement algorithm using the measure defined earlier in this section. Steps 1–4 show the strategies in each step. Dotted arrows are those *not* selected by the current strategy of Player 0 and the corrseponding optimal counterstrategy of Player 1.
1. In the initial strategy, Player 0 goes from a to a and from c to a. Player 1's optimal counterstrategy goes left in b, since that choice provides for a shorter path length. The switch in a is attractive since b has a better value than a (the good color 2 is in the color hit record) and the switch in c is attractive since the path length is longer and the loop is losing.
2. Player 0 could do any one of the two attractive switches, but say he switches in b. The optimal counterstrategy of Player 1 is still to go left, since it gives a better loop value in b. As guaranteed by Theorem 3, the new strategy gave a better value even after Player 1 recomputed his counterstrategy. Now the switch in c is non-attractive, but the switch in a is still attractive.
3. Player 0 has now switched in a. The optimal counterstrategy of Player 1 has now changed: going left would give a loop over a and b, with tint 2, but going right provides the 1-better loop tint 0. The vertex values of both a and b have improved. The switch in c has become attractive, because the good color 2 appears in the color hit record.
4. After switching in c, Player 0 has now reached a strategy with no attractive switches. As guaranteed by Theorem 4, this means that the stategy is optimal. Note that *both* Player 1's two possible counterstrategies are equally good at this point: even if the

loop length is greater if he chooses to go right, this is not reflected in any vertex value (the path length of the loop major is 0).

4.4 Are More Shallow Measures Possible?

One could imagine even tighter measures than given by Definition 9. For instance, instead of our color hit record, one could record only one bit for every color $> c(t)$, indicating whether a vertex of this color appears on the path. However, this definition would violate the *optimality* criterion of Section 3, as demonstrated by Figure 2. Round and square vertices in Figure 2 belong to Player 0 and 1 respectively. The value of the left successor of vertex v colored 2 would be better than the value of the right successor, since the odd color 15 is part of the path from the vertex colored 15 to the loop. However, switching right in v (with a worse value) would let Player 0 win in more vertices. We thus have a stable strategy that is not optimal.

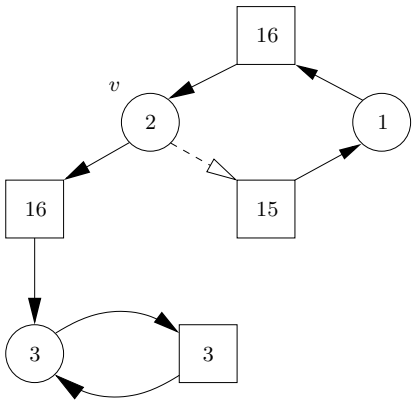

Fig. 2. Inattractive switch in v allows Player 0 to win in more vertices.

Currently we do not know any evaluation functions on strategies that would have all the required properties and possess a more shallow value space. The existence of such measures is not excluded.

4.5 Path Comparison Properties

Proofs in the subsequent sections rely on the following technical lemmas.

Lemma 1. *If* $(P_0, p_0) \succeq_t (P_1, p_1)$*, then* $(Q + P_0, q + p_0) \succeq_t (Q + P_1, q + p_1)$*, where* $Q + P_i$ *is vector addition.*

Intuitively, by extending the paths P_0 and P_1 with the same path Q, we do not change their initial order.

Proof. The first position in which the vectors $Q + P_0$ and $Q + P_1$ differ is the same as the first position in which P_0 and P_1 differ. If this position is one of $k, \ldots, c(t)$, then $(Q + P_0, q + p_0) \succ_t (Q + P_1, q + p_1)$, since $(P_0, p_0) \succeq_t (P_1, p_1)$. If, on the other hand, P_0 and P_1 coincide in all positions $k, \ldots, c(t)$, then $(-1)^{c(t)} \cdot (q + p_0) \leq (-1)^{c(t)} \cdot (q + p_1)$ iff $(-1)^{c(t)} \cdot p_0 \leq (-1)^{c(t)} \cdot p_1$, and the conclusion follows.

Lemma 2. *Let $p \geq 0$, $q > 0$ be integers and P, Q be path color hit records relative to a tint t, where Q corresponds to a path with the largest tint $t' \succ t$. Then*

$$(P, p) \prec_t (P + Q, p + q).$$

Intuitively, adding a good path Q to P gives an improvement.

Proof. Assume $t \prec t'$.

If $t < t'$ (numeric comparison) then t' is 0-winning and the first position (counting from the largest color k in decreasing order) where P and $P + Q$ are different is $c(t')$, and $P + Q$ is numerically greater at this position. Hence $(P, p) \prec_t (P + Q, p + q)$.

If $t' < t$ then t is 0-losing and P and $P+Q$ coincide in all positions $k, \ldots, c(t)$. Since t is 0-losing, the larger path length determines that $(P, p) \prec_t (P + Q, p + q)$.

5 Profitability of Attractive Switches

Our algorithms proceed by making attractive single switches. Attractiveness is established locally, by comparing values of the successors of a vertex with respect to the current strategy; see Definition 17. In this section we prove that making an attractive switch actually leads to an improvement, captured by the notion of profitability.

Definition 19 (Profitability). *Say that a single switch in a vertex v from σ to σ' is* profitable *if:*

1. *$\nu_\sigma(w) \preceq \nu_{\sigma'}(w)$, for all vertices w, and*
2. *either a) $\nu_\sigma(v) \prec \nu_{\sigma'}(v)$, or b) $\nu_\sigma(\sigma(v)) \prec \nu_{\sigma'}(\sigma'(v))$.* □

Condition 1 requires that all vertices (non-strictly) improve their values after a switch, while 2 stipulates a strict value improvement either for the switch vertex itself (2.a), or for its successor (2.b).

Proceeding by profitable switches guarantees termination, since vertices can only improve their values, or get better successors, a finite number of times.

Proposition 2. *Every sequence of profitable switches terminates.* □

Section 7 establishes an upper bound on the number of profitable switches.

Profitability of attractive switches proved in this section (Theorem 3) is a consequence of the the preceding complicated definitions of values and value comparison. On the one hand, profitability of attractive switches guarantees termination of any algorithms based on attractive switches. On the other hand, it improves the efficiency of every improvement step, since it allows our algorithms to find better neighbors by looking only at attractive switches rather than computing the values of all neighbors, as explained in Sections 3 and 9.

Theorem 3 (Profitability). *Every attractive switch is profitable.*

(Corollary 2 will show the converse, i.e., the notions of profitable and attractive are actually *equivalent*.)

Proof. Consider an attractive (say left-to-right) switch in vertex v: from vertex v_1 with value (t_1, P_1, p_1) to vertex v_2 with value (t_2, P_2, p_2). Let σ and σ' be the strategies before and after the switch, respectively. Let τ and τ' be optimal counterstrategies against σ and σ', respectively.

We start by proving that the second property of Definition 19 holds for all attractive switches. There are two major cases.

Case 1. In his optimal response after the switch in v, Player 1 *does not revisit* v starting from v_2. Assume $\nu_{\sigma'}(v_2) \prec \nu_\sigma(v_2)$. This contradicts the fact that τ is optimal, since the trace taken by τ' not going through v was available also before the switch. So $\nu_{\sigma'}(v_2) \succeq \nu_\sigma(v_2)$. If $t_1 \prec t_2$ the loop value at v will improve after the switch. If $t_1 = t_2$, Lemma 1 implies $\nu_\sigma(v) \prec \nu_{\sigma'}(v)$. This finishes the proof of Case 1.

Case 2. In his optimal response after the switch in v, Player 1 *does revisit* v starting from v_2. Consequently, we get a loop through v and v_2. We should prove that the value of v on this loop is better for Player 0 than the value before the switch, or remains the same while v_2 has a better value after the switch than v_1 had before the switch.

In this case, the loop values $t_1 = t_2$ *coincide*. Indeed, Player 1 can reach v from v_2, for which the loop value before the switch is t_1, so $t_2 \preceq t_1$. Also, attractiveness means that $t_2 \succeq t_1$. Denote $t := t_1 = t_2$.

Any path from v_2 to v could have been taken by Player 1 *before the switch*. Respectively, we analyze cases when such a path was or was not used by Player 1 before the switch.

Subcase 2.1. Suppose such a path was actually taken by Player 1 before the switch. By definition, (P_2, p_2) is the $\prec_t$-smallest path value of any paths from v_2 to t.

1 If $P_1 \neq P_2$, then the largest color in $P_2 - P_1$ is even, since the switch is attractive. So the largest tint t' on the path from v_2 to v is $\prec$-better than t. Since τ is optimal, this is the worst possible path from v_2 to v. Hence the new loop over v and v_2 has t' as the $<$-largest tint, so the loop tint has improved.
2 If $P_1 = P_2$, we first consider the case when v was on the loop over t before the switch. This implies $P_1 = P_2 = \overline{0}$. Thus there is no path from v_2 to t or v with an odd-colored highest tint bigger than t. This means that the loop after the switch will have a major $t' \leq t$. If $c(t)$ is odd, then $p_1 < p_2$, and either $t' < t$ or $t' = t$. If $t' < t$ then we are done. If $t' = t$, the path length in v_2 will improve, since the path from v_2 to t achieving (P_2, p_2) is 1-optimal. Thus the path length in v improves, unless v is the loop major; in any case the value of the successor of v improves.
If, on the other hand, $c(t)$ is even, then $p_2 < p_1$, so no 1-optimal path from v_2 to t passes through v (unless $t = v$ and the path ends up in v). Hence, any 1-optimal path from v_2 to v passes through t, and such a path will be taken after the switch. Thus the loop tint in v_2 after the switch is t and the path length is p_2. If $v \neq t$ then the path length in v improves; if $v = t$ then v gets a better successor.
Finally, if v was not on the loop over t before the switch, then $p_1 < p_2$ so $c(t)$ is odd. Also, by attractiveness there can be no path from v_2 to v with a highest tint t' such that $c(t')$ is odd and $t' > t$. Therefore, the loop tint after the switch is smaller than or equal to t. If it is smaller, then it is also better, and we are done. If it is equal, then the path value in v_2 after the switch will be (P_2, p_2) or better; any worse path from v_2 to t would have been taken before the switch. Thus by Lemma 1 the value will improve in v.

Subcase 2.2. Suppose such a path P_2'' from v_2 to v *was not taken* by Player 1 in his optimal response before the switch (alternatively to Subcase 2.1), and Player 1 preferred P_2 instead, as shown in the Figure 3 (where the round vertex belongs to Player 0, square vertices may belong to any player, P_2' is a path from v_2 through v to the loop with major tint t).

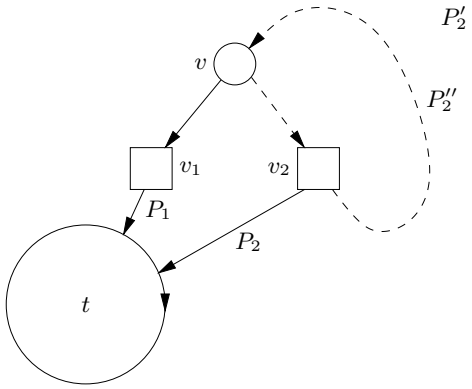

Fig. 3. Illustration to case 2.2 of the proof that attractiveness implies profitability.

In the sequel, we abuse notation and identify the paths P_1, P_2, P_2', P_2'' with their color hit records. From attractiveness and optimality conditions we derive:

$$(t, P_1, p_1) \prec (t, P_2, p_2) \preceq (t, P_2', p_2') \equiv (t, P_1 + P_2'', p_1 + p_2''),$$

where (P_2'', p_2'') is the path from v_2 to v. The first inequality above follows from attractiveness, the second says that Player 1 did not take the path through v before the switch.

By transitivity, $(t, P_1, p_1) \prec (t, P_1 + P_2'', p_1 + p_2'')$. By definition of $\prec$ it follows that:

- either P_2'' contains a major color from $k, ..., c(t)$ winning for Player 0 and corresponding to a 0-better tint than t,
- or P_2'' does not contain a color from $k, \ldots, c(t)$ (note that $c(t)$ is odd in this case, since $p_2'' > 0$).

In both cases, the new loop through v_2 and v improves the loop value of v after the switch. This finishes the proof of the first claim.

Proof for property 1 of Definition 19. Consider any vertex $w \neq v$, and a 1-optimal trace from w under σ'. If the trace does not contain v it would be possible under σ as well, and thus $\nu_\sigma(w) \preceq \nu_{\sigma'}(w)$. Assume the trace contains v.

Let t and t' be the loop values of w under σ and σ', respectively. We first prove that $t' \succeq t$. Since w reaches v after the switch, they have the same loop value, which is better than or equal to the loop value of v before the switch (as shown above), which is better than or equal to t (since v was reachable from w before the switch).

We now prove that the path value of w does not decrease with the switch. If $t' = \mathbf{t}(w)$ then the path value is constant $(\overline{0}, 0)$ so either it did not change with the switch or the loop tint changed, hence improved, with the switch. Assume $t' \neq \mathbf{t}(w)$ and consider the path P taken in the optimal counterstrategy of Player 1 from w to t against σ'. If P does not contain v then it was possible under σ as well, so the 1-optimal path before the switch was at least as bad as P. If P contains v, consider the initial segment of

P leading up to v. This segment was possible under σ, so Lemma 1 implies that the 1-optimal path before the switch was at least as bad as P.

It follows that $\nu_\sigma(w) \preceq \nu_{\sigma'}(w)$.

Example 2. We conclude this section with a small example explaining why we need profitability as introduced by Definition 19. Consider the game in Figure 4.

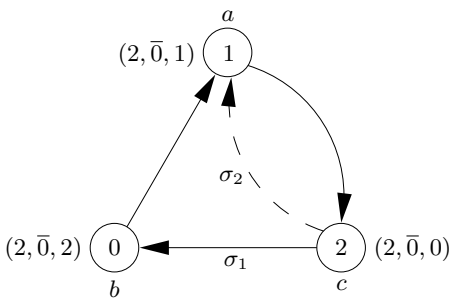

Fig. 4. Example showing the need for the complicated definition of profitability

The switch in vertex c from the solid to the dashed arrow is attractive, because a seems better than b (shorter path). However, the values of the strategies σ_1 before and σ_2 after the switch are *the same*, namely (the three components are the values of vertices a, b, c):

$$((2, \bar{0}, 1), (2, \bar{0}, 2), (2, \bar{0}, 1))$$

This example shows that comparing just values of strategies as in Definition 15 would not provide for an improvement (profitability). According to our Definition 19, the switch from σ_1 to σ_2 described above is profitable, because the new successor of c has a better value. In the proof, this type of situation is considered in the ends of the first and second paragraphs of Subcase 2.1.2. □

Remark 3. It would be desirable for the strategy value to reflect the improvement of attractive switches. We note briefly that this can be achieved by adding a fourth component to the measure, recording the loop length for any loop majors and 0 for non-major vertices.

The fourth components of two vertex values will be compared only if the other three are equal. In this case, similarly to the third component, longer (shorter) loop lengths are favored when the loop tint is odd (even). As shown in the proof of Theorem 5, any attractive switch that does not give a strict improvement in the switching vertex improves the path length in one of its successors, which for the loop major is equal to the loop length minus 1. Thus the modified measure has the property that a switch is attractive if and only if it strictly improves the value.

6 Stability Implies Global Optimality

In this section we show that iterative improvement can terminate once a stable strategy with no attractive switches is found (see Definition 18). In more general terms, every local optimum is global. This is one of the main motivations for the complex strategy evaluation definitions.

The theorem below states that any stable strategy has a value that is better than or equal to the values of all other strategies. This means that any stable strategy is *globally*

optimal, which is crucial in the subexponential complexity analysis. As a consequence of the theorem, we also derive that any stable strategy is *winning* from all vertices in the winning set of Player 0 (Corollary 1), which means that it is a solution to the parity game. This ensures correctness of the algorithms.

Theorem 4 (Optimality). *Let σ be a stable strategy of Player 0. Then for all strategies σ' of Player 0 and all vertices v of the game, $\nu_\sigma(v) \succeq \nu_{\sigma'}(v)$.*

Proof. Let σ be a strategy with no attractive switches with respect to $\tau \equiv \tau(\sigma)$, an optimal counterstrategy of Player 1. We will show that no strategy σ' of Player 0 can improve the value of any vertex, even if Player 1 fixes and continues to invariably use his strategy τ.

Suppose, towards a contradiction, that there is a strategy σ' and a vertex w_0 for which $\nu_{\sigma,\tau}(w_0) \prec \nu_{\sigma',\tau}(w_0)$. Figure 5 depicts this situation, where round vertices belong to Player 0 and square vertices may belong to any player, bold paths are parts of the traces determined by (σ, τ) where σ and σ' coincide, dashed edges are taken only by σ' and normal edges are taken only by σ. Also, (r_i, C_i, c_i) and (t, B_i, b_i) are values of vertices, whereas (D_i, d_i) and (P_i, p_i) are path values, all computed with respect to (σ, τ). In particular, the loop shown in Figure 5 is determined by the trace from w_0 under the pair of strategies (σ', τ). Denote this loop λ.

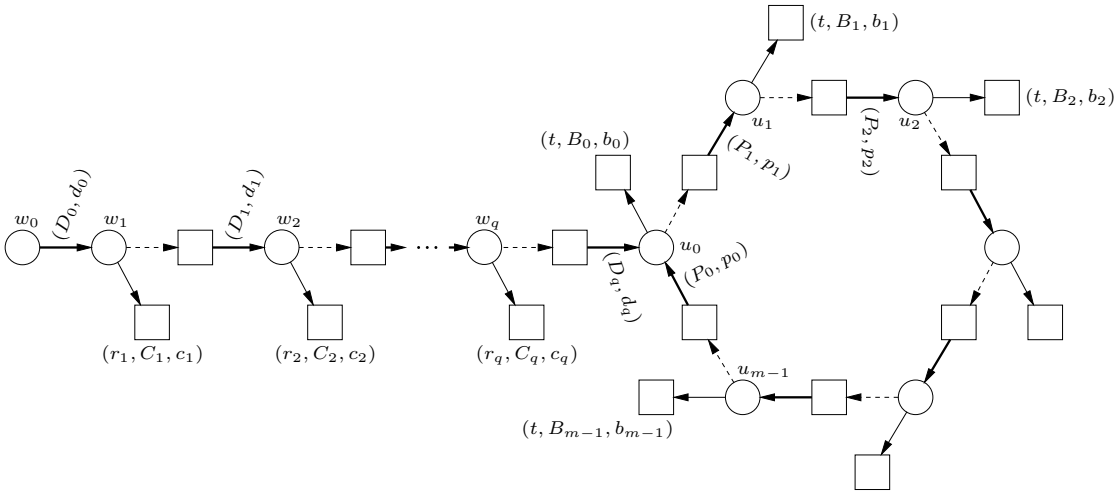

Fig. 5. Illustration to the proof that stable strategies are optimal.

We derive the desired contradiction in several steps.

1. First, we claim that all loop values of vertices u_i with respect to (σ, τ) *coincide* (call this value t, as shown in the picture). Indeed, suppose $u_{(j+1 \mod m)}$ has a 0-better loop value than u_j (such a pair should exist if not all loop values of u_i's are equal). Then switching from σ in u_j is attractive, contradicting the assumption that σ is stable with respect to τ.

2. Second, we show with a similar argument that $r_1 \succeq r_2 \succeq \cdots \succeq r_q \succeq r_{q+1} \equiv t$. Suppose not, and $r_i \prec r_{i+1}$ for some i. Then the switch in w_i is attractive, contradicting the assumption that σ is stable with respect to τ.

3. Now we prove that the vertex of tint t actually belongs to λ. Suppose not. Then the major tint t' on λ must satisfy $t' \succ t$: by assumption $\nu_{\sigma',\tau}(w_0) \succeq \nu_{\sigma,\tau}(w_0)$, so $t' \succeq r_1$, and we showed that $r_1 \succeq t$. Since none of the switches in u_i (from a normal to a dashed arrow) is attractive, we derive the following system of inequalities, one for each u_i:

$$\begin{aligned} (B_0, b_0) &\succeq_t (P_1 + B_1, p_1 + b_1) \\ (B_1, b_1) &\succeq_t (P_2 + B_2, p_2 + b_2) \\ &\cdots \\ (B_{m-1}, b_{m-1}) &\succeq_t (P_0 + B_0, p_0 + b_0) \end{aligned} \tag{3}$$

Lemma 1 applied to this system of inequalities implies

$$(B_0, b_0) \succeq_t (P_0 + P_1 + \cdots + P_{m-1} + B_0, p_0 + p_1 + \cdots + p_{m-1} + b_0) \tag{4}$$

By Lemma 2, (4) is a contradiction, because the largest tint represented in $P_0 + P_1 + \cdots + P_{m-1}$ is t', and $t' \succ t$. Thus t is on λ.

4. We now proceed to proving that $t = t'$, where t' again is the loop major on λ. Suppose not. Then we know that $t < t'$ (numerical comparison), since t' is the major tint on λ. This also implies that $c(t')$ is even since $t' \succ t$. Assume without loss of generality that t belongs to the segment of λ between the successor of u_{m-1} and u_0 (we do not lose generality because the argument does not deal with the initial path through w_i's). Let (P'_0, p'_0) be the path value in the vertex succeeding u_{m-1} on λ. We can now build a system of inequalities similar to (3):

$$\begin{aligned} (B_0, b_0) &\succeq_t (P_1 + B_1, p_1 + b_1) \\ (B_1, b_1) &\succeq_t (P_2 + B_2, p_2 + b_2) \\ &\cdots \\ (B_{m-1}, b_{m-1}) &\succeq_t (P'_0, p'_0) \end{aligned} \tag{5}$$

Again applying Lemma 1 gives

$$(B_0, b_0) \succeq_t (P_1 + \cdots + P_{m-1} + P'_0, p_1 + \cdots + p_{m-1} + p'_0) \tag{6}$$

The vertex succeeding u_0 under σ is part of the loop over t under σ and thus B_0 must be the all-zero vector. If t' belongs to the part of λ represented by $(P_1 + \cdots + P_{m-1} + P'_0, p_1 + \cdots + p_{m-1} + p'_0)$, then (6) is a contradiction, since t' would be the highest tint on this segment and $c(t')$ is even. Also, if t' instead lies between t and u_0 on λ, we get a contradiction since then t' and not t would be the major tint on the loop under σ. Thus $t = t'$.

5. We proceed to demonstrating that $t = t' = r_i$, for all i. Indeed, since $\nu_{\sigma,\tau}(w_0) \prec \nu_{\sigma',\tau}(w_0)$, we must have $t' \succeq r_1$. Together with $t = t'$, shown above, and the fact that the sequence $r_1, r_2, ..., r_{q+1} \equiv t$ is nonincreasing, we obtain $t = t' = r_i$, for all i.

6. Now we finally show the contradiction. We know that all paths from all w_i's and u_j's, both with respect to (σ, τ) and (σ', τ), lead to the *same* major vertex t on λ. Let u_{i-1} be the last vertex on the path from w_0 to t under (σ', τ) where σ and σ' diverge. Let (P'_i, p'_i) be the path value of its successor with respect to (σ, τ).

On the one hand, from the assumption that (σ', τ) gives a better value for w_0 we obtain:

$$(D_0 + C_1, d_0 + c_1) \prec_t (D_0 + D_1 + \cdots + D_q + P_1 + \cdots + P'_i, d_0 + d_1 \cdots + d_q + p_1 + \cdots + p'_i) \quad (7)$$

On the other hand, from non-attractiveness of switches, we have

$$\begin{aligned}
(C_1, c_1) &\succeq_t (D_1 + C_2, d_1 + c_2) \\
(C_2, c_2) &\succeq_t (D_2 + C_3, d_2 + c_3) \\
(C_3, c_3) &\succeq_t (D_3 + C_4, d_3 + c_4) \\
&\cdots \\
(C_q, c_q) &\succeq_t (D_q + B_0, d_q + b_0) \\
(B_0, b_0) &\succeq_t (P_1 + B_1, p_1 + b_1) \\
(B_1, b_1) &\succeq_t (P_2 + B_2, p_2 + b_2) \\
&\cdots \\
(B_{i-2}, b_{i-2}) &\succeq_t (P_{i-1} + B_{i-1}, p_{i-1} + b_{i-1}) \\
(B_{i-1}, b_{i-1}) &\succeq_t (P'_i, p'_i)
\end{aligned}$$

Applying Lemma 1 to this system, we derive:

$$(C_1, c_1) \succeq_t (D_1 + \cdots + D_q + P_1 + \cdots + P'_i, d_1 + \cdots + d_q + p_1 + \cdots + p'_i)$$

Applying Lemma 1 once again to the last inequality, we get:

$$(D_0 + C_1, d_0 + c_1) \succeq_t (D_0 + \cdots + D_q + P_1 + \cdots + P'_i, d_0 + \cdots + d_q + p_1 + \cdots + p'_i),$$

which contradicts (7). This finishes the proof.

As consequences of Theorem 4, the strategy evaluation function presented in Section 4 has the remaining desired property stated in Section 3:

Corollary 1 (Stable Strategies are Winning). *Any stable strategy of Player 0 is also* winning *from all vertices in the winning set of Player 0 and has a value that is at least as good as that of any other strategy.*

Proof. Let σ be a stable strategy. Theorem 4 states that it is at least as good as all other strategies. Moreover, all vertices with 0-winning loop values under σ are winning for Player 0, since the traces from them under σ and an optimal counterstrategy of Player 1 lead to 0-winning loops. Also, no vertex with a 1-winning loop value with respect to σ can be winning for Player 0, since none of his other strategies gives them a better value.

This means that as soon as a stable strategy is found, the iterative improvement algorithms may stop.

From Theorem 4, we can also derive the equivalence of profitability and attractiveness.

Corollary 2. *Any profitable switch is also attractive.*

Proof. Suppose that a switch in vertex v from σ to σ' is profitable but not attractive. Remove from G all edges leaving vertices of Player 0 except those used by σ and σ'. In the resulting graph, σ is stable, and thus by Theorem 4, the other strategy σ' cannot have a better value. Therefore, the only possibility is that the value in v remains the same and $\nu_\sigma(\sigma(v)) \prec \nu_{\sigma'}(\sigma'(v))$. But this is a contradiction, since $\nu_\sigma(\sigma(v)) \succeq \nu_\sigma(\sigma'(v))$ (the switch is not attractive) and $\nu_\sigma(\sigma'(v)) \succeq \nu_{\sigma'}(\sigma'(v))$ (σ is stable, hence optimal).

This result shows that only looking at attractive switches is no restriction compared to actually evaluating all neighboring strategies (all strategies obtainable by a single switch from the current strategy).

7 Complexity: Depth of the Measure Space

In this section we provide an upper bound on the maximal number of improvement steps of any iterative improvement algorithm using our definition of strategy values.

We need the following technical lemma.

Lemma 3. *If a profitable switch in v from σ to σ' does not change the value in v, then the loop value in v under both strategies is $\mathbf{t}(v)$, and for all vertices w of the game, $\nu_\sigma(w) = \nu_{\sigma'}(w)$.*

Proof. Assume the switch in v from σ to σ' does not change the value in v. Definition 19 implies that $\nu_\sigma(\sigma(v)) \prec \nu_{\sigma'}(\sigma'(v))$. If v was not a loop major, its path value would depend on its successor. Thus Lemma 1 applies and gives $\nu_\sigma(v) \prec \nu_{\sigma'}(v)$, a contradiction.

Now consider any other vertex w. If the trace from w with respect to σ and an optimal counterstrategy τ did not go though v, the value in w does not change. Indeed, the same trace is still available to Player 1, so $\nu_\sigma(w) \succeq \nu_{\sigma'}(w)$, and on the other hand $\nu_\sigma(w) \preceq \nu_{\sigma'}(w)$ by Theorem 4, since the switch was attractive.

If the trace from w under (σ, τ) did pass through v, the loop value t of w remains the same. Also, the same paths from w to v are available to Player 1 under both strategies, so the path value also cannot improve.

Theorem 5. *The strategy evaluation function allows for at most $O(n_0 \cdot n^2 \cdot (n/k+1)^k)$ profitable switches.*

Proof. For each vertex there are $\prod_{i=1}^{k}(|V^i|+1)$ possible color hit records. This expression takes its maximum when there are equally many vertices of each color. Thus at most $(n/k+1)^k$ color hit records are possible. There are also n possible loop values and n possible path lengths. Thus each vertex can improve its value at most $n^2(n/k+1)^k$ times. Profitable switches that do not improve values are only possible in vertices v with current loop value $\mathbf{t}(v)$, by Lemma 3. The successor after such a switch always has an empty color hit record, since it must be part of the loop over v. Therefore, it can only be the path length of the new successor that is better than the path length of the old one. This can only happen at most n times per vertex, namely at most once per possible loop length when the value of v is $(\mathbf{t}(v), \overline{0}, 0)$. Thus only $n + n^2(n/k+1)^k = O(n^2(n/k+1)^k)$ switches are possible in each vertex of Player 0, and we get an upper bound of $O(n_0 n^2 (n/k+1)^k)$ on the total number of switches.

8 Computing Optimal Counterstrategies

In this section we present an algorithm for computing the values of strategies. Since values of strategies of Player 0 are defined as minima over all counterstrategies of Player 1, the algorithm implicitly computes optimal counterstrategies. A counterstrategy τ is *optimal* against σ if for every vertex v in the game and every strategy τ' of Player 1, $\nu_{\sigma,\tau}(v) \preceq \nu_{\sigma,\tau'}(v)$. The correctness proof for the algorithm shows that there is at least one counterstrategy that achieves the best possible value from every vertex. We also show that the algorithm has polynomial time complexity.

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

In a class L_t there may be cycles that do not contain t. As the following property shows, such loops must, however, have majors that are 0-better than t. This is important for the correctness of the algorithm.

Property 1. In every class L_t the following holds. If a tint t' is the biggest tint on a cycle, then $t \prec t'$. □

Once the partition into classes L_t has been accomplished, the loop value of each vertex is known and we can proceed to computing the path values. For each class L_t, Algorithm 2 below uses dynamic programming to calculate the values of 1-optimal paths of different lengths from each vertex to t. For each vertex, the algorithm first computes the optimal color hit record (abbreviated *chr* in the algorithm) over all paths of length 0 to the loop major (∞ for each vertex except t). Then it calculates the color hit record of optimal paths of length one, length two, and so forth. It uses the values from previous steps in each step except the initial one.

Algorithm 2: Computing path values within a class L_t.

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

The function ADD-COLOR takes a tint, a color hit record, and a second tint. If the second tint is bigger than the first one, then ADD-COLOR increases the position in the

vector representing the color of the second tint. The function always returns ∞ when the second argument has value ∞.

Lemma 4 (Properties of the Algorithm). *Algorithm 2 has the following properties:*

1. *It correctly computes values of 1-optimal paths.*
2. *Optimal paths are simple.*
3. *The values computed are consistent with an actual positional strategy that guarantees loop value t.* □

Proof. First we prove clause 2 of the Lemma. Observe that following a path to t with a cycle can never yield a worse value than following the same path, but without traversing the cycle. This is a consequence of Lemma 1. If all tints t' on the cycle satisfy $t' < t$, then $c(t)$ must be odd, and since traversing the loop contributes nothing to the color hit record the shorter path will be better for Player 1. If on the other hand the major tint t' on the cycle satisfies $t < t'$, then $c(t')$ must be even and traversing the cycle will give a better color hit record. This proves that 1-optimal paths are simple.

Now we move on to proving clause 1. Consider the class C of all paths p from all vertices in L_t to the major vertex t. We prove by induction on the length of such paths, that for each vertex v the algorithm actually computes the values of the 1-optimal paths of each length from v to t. Note that these paths may be non-simple. Cycles can be traversed a finite number of times (since the algorithm makes at most $|L_t| - 1$ iterations). This is not a contradiction to clause 2, since the compluted paths are not necessarily 1-optimal, only the best ones of a particular length.

Base: Path length = 0. There is just one path of length 0 in C (from t to itself) and obviously the algorithm correctly computes the optimal value in this case.

Inductive Case. Assume that the values computed for paths of all lengths $j < i$ are the values of the 1-optimal length j paths from each vertex to the major t. Consider any vertex v. If no successor of v has a value other than ∞ for paths of length $i - 1$ there cannot be a path of length i from v to the loop major, and the value $v.chr[i]$ is correctly set to ∞.

If, on the other hand, at least one successor of v has a value for paths of length $i-1$, then consider any successor v' of v. By Lemma 1 any optimal path from v directly via v' coincides after the first step with the optimal path from v'. Thus the minimum calculated on line (7) is the correct value for $v.chr[i]$.

This finishes the proof of algorithm correctness. Finally, to prove clause 3 we first observe that the algorithm maintains the following property:

> *If a value (P, p) has been computed for a vertex v other than t, then v has a successor v' for which a value $(P', p-1)$ has been computed, such that $P - P'$ is either the vector with all zeros, or it has all zeros except a one in the position for color $c(v)$.*

Thus from every vertex except t we can select one outgoing edge (positional strategy) providing for the optimal value.

It remains to select one edge leaving t to complete the description of a positional strategy providing for optimal path values and loop value t. From at least one of the successors of the major t, there must be a path to t on which no tint t' such that $t < t'$ appears. Furthermore, by Property 1, no successor of t has a path to t with biggest tint

t' such that $t' \prec t$ and $t < t'$. Thus there is at least one successor v of t such that the 1-optimal path from v to t contains no even color bigger than $c(t)$. By going from t to the successor with the worst value Player 1 can therefore ensure that the loop tint for all vertices in L_t will be t. Thus there is a positional strategy corresponding to the path values computed by the algorithm.

Lemma 5 (Algorithm Complexity). *The algorithm for computing an optimal counterstrategy runs in time $O(|V| \cdot |E| \cdot k)$, where $|V|$ is the number of vertices of the graph, $|E|$ is the number of edges, and k is the number of colors.*

Proof. Calculating the classes L_t is straightforwardly done in time $O(|V| \cdot |E|)$. The bottleneck in each call to Path-Values is the double loop on lines (4) to (7). Taking the $\prec_t$-minimum over all successors of v requires one $\prec_t$-comparison per edge leaving v, and each $\prec_t$-comparison requires $O(k)$ integer comparisons. Thus the loop on line (6) needs $O(|E| \cdot k)$ comparisons. It is executed $\sum_t (|L_t| - 1) = O(|V|)$ times altogether (counting all calls to Path-Values), so the total running time is $O(|V| \cdot |E| \cdot k)$.

9 Kalai-Style Randomization for Games with Unbounded Outdegree

As discussed in Section 3, any non-binary parity game reduces to a binary one, and the Ludwig-style algorithm applies. However, the resulting complexity gets worse and may become exponential (rather than subexponential) due to a possibly quadratic blow-up in the number of vertices. In this section we describe a different approach relying on the randomization scheme of Kalai [15,12] used for Linear Programming. This results in a subexponential randomized algorithm directly applicable to parity games of arbitrary outdegree, without any preliminary translations. When compared with reducing to the binary case combined with the Ludwig-style algorithm of Section 3, the algorithm of this section provides for a better complexity when the total number of edges in the game graph is roughly $\Omega(n \log n)$.

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

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

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

The Algorithm takes a game $G \in \mathcal{G}(d, m)$ and an initial strategy σ_0 as inputs. It uses the subroutine COLLECT-IMPROVING-FACETS, described later, that collects a set of pairs (F, σ) of σ_0-improving facets F and corresponding witness strategies $\sigma \succ \sigma_0$.

Algorithm 3: Kalai-Style Optimization Algorithm

```
KALAI(game G, initial strategy σ_0)
(1)   if no vertex of Player 0 in G has more than one successor
(2)       return σ_0
(3)   M ← COLLECT-IMPROVING-FACETS(G, σ_0)
(4)   choose a random pair (F, σ_1) ∈ M
(5)   σ* ← KALAI(F, σ_1)
(6)   if σ* has an attractive switch to some edge e in G
(7)       return KALAI(G, σ*[e])
(8)   else
(9)       return σ*
```

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

How to Find Many Improving Facets. Now we describe the subroutine COLLECT-IMPROVING-FACETS. The goal is to find r facets that are σ_0-improving, where r is a parameter (Kalai uses $r = \max(d, m/2)$ to get the best complexity). To this end we construct a sequence $(G^0, G^1, \ldots, G^{r-d})$ of subgames of G, with $G^i \in \mathcal{G}(d, d+i)$ and G^i is a subgame of G^{i+1}. All the $d+i$ facets of G^i are σ_0-improving; we simultaneously determine the corresponding witness strategies σ^j optimal in G^j. The subroutine returns r facets of G, each one obtained by fixing one of the r edges in $G^{r-d} \in \mathcal{G}(d, r)$. All these are σ_0-improving by construction.

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

To construct G^{i+1} from G^i, let e be the target edge of an attractive switch from σ^i in G. (Note that σ^i is optimal in G^i but not necessarily in the full game G. If it is, we terminate.) Let G^{i+1} be the game G^i with e added. Recursively apply the algorithm to find the optimal strategy σ^{i+1} in G^{i+1}. Note that fixing the choice in G to any of the $d+i$ choices in G^i defines a σ_0-improving facet. Therefore, the final G^{r-d} has r σ_0-improving facets facets.

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

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

The first term represents the work of finding r different σ_0-improving facets in COLLECT-IMPROVING-FACETS. The second term comes from the first recursive call on

line 5.[2] The last term accounts for the recursive call on line 7 and can be understood as an average over the r equiprobable choices made on line 4, as follows. All facets of G are partially ordered by the values of their optimal strategies (although this order is unknown to the algorithm). Since the algorithm only visits improving strategies, it will never again visit facets possessing optimal strategies that have values worse, equal, or incomparable to σ^*. In the *worst* case, the algorithm selects the r worst possible facets on line 3. Thus, in the worst case, the second recursive call solves a game in $\mathcal{G}(d, m-i)$ for $i = 1, \ldots, r$, with probability $1/r$. This justifies the last term.

Kalai solves the recurrence for $r = \max(d, m/2)$. The result is subexponential, $m^{O(\sqrt{d/\log d})}$.

By symmetry, we can choose to optimize a strategy of the player possessing fewer vertices. Let n_i denote the number of vertices of player i. Since m is bounded above by the maximal number of edges, $(n_0+n_1)^2$, and $d \leq \min(n_0, n_1)$, we get

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

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

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

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

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

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

10 Conclusions and Further Work

We presented and justified two new discrete randomized algorithms for finding optimal strategies in parity games. The algorithms rely on evaluating strategies and iteratively improving their values, by making a change in one vertex at a time (single switch), until an non-improvable strategy is obtained. This strategy is optimal and allows for computing winning sets of both players.

The algorithms presented are the first algorithms with the two simultaneous worst-case bounds:

A. *subexponential* ($2^{O(\sqrt{n_0})}$ or $2^{O(\sqrt{n_0\log(n_0)})}$) in the number of vertices n_0 of Player 0,
B. *exponential* $O(poly\cdot(n/k+1)^k)$ in the number of colors.

Bound **A** is stronger than any currently known upper bound for other parity games algorithms when the number of colors is unbounded (except the algorithm from [20] that has the same bound **A**, but not **B**). Bound **B** makes our algorithms comparable

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

with other parity games algorithms when the number of colors is fixed and small. It should be noted that for all other such algorithms exponential, $\Omega((n/k+1)^{\Theta(k)})$, worst cases are known [7,14]. For our algorithms, which are randomized, currently there are no known examples of exponential behavior [3,2] (experiments of [24] report the same for their algorithm). We also believe that our bound **B** is 'loose', being based on a simple count of the total number of values in the measure space. Is seems possible to improve the bound **A**, by taking additional structure of parity games into account [6].

Both algorithms presented are discrete graph-theoretic, operating in terms of paths, loops, colors, and directly apply to parity games without any intermediate translations. The Ludwig-style algorithm applies to binary parity games and the Kalai-style algorithm applies to games of unbounded vertex outdegree. To our knowledge, there were no such subexponential algorithms for parity games before. The Kalai-style subexponential algorithm from Section 9 straightforwardly extends to cover the more general class of simple stochastic games with *arbitrary* vertex outdegree. Note that for such games the algorithm of Ludwig [16] becomes *exponential*, requiring a preliminary translation with a possibly quadratic increase in the number of vertices.

While presenting the algorithms, we tried to factor out abstract properties the strategy evaluation function must satisfy in order for our algorithms to work correctly and efficiently. Then we presented one such possible valuation providing for the bounds **A** and **B**. Our valuation is close in spirit to the one used in [24]. We note in passing that the latter measure can also be safely used with our algorithms, but it does not provide for bound **B**, giving just $O(2^n)$, which is no improvement over **A**. The advantage of our measure is that it is 'more tight', and takes only at most $O(n^3 \cdot (n/k+1)^k)$ different values. Since each step of the algorithms improves the value, this is an additional upper bound **B** on the number of iterations. This is an improvement over bound **A** when the number of colors is small.

The advantages over reducing parity games to simple stochastic games and solving them with the algorithm of [16] are: (1) we do not have to solve Linear Programming problems with high-precision arithmetic; (2) our small measure space allows for better complexity when the number of colors is small; (3) our method is more direct, in the sense that all work is done in terms of parity games without reducing to other problems. The algorithm of [20], although it does not rely on Linear Programming, optimizes graphs with weights represented as high-precision rational numbers. Recall that both algorithms of [16,20] become exponential for the games of unbounded outdegree.

Let us briefly summarize directions for possible further improvements of the algorithms and ideas presented in this paper.

1. For the algorithm of Section 9, we rely entirely on Kalai's solution of a recurrence. However, our recurrence is somewhat better than the one for the Linear Programming case. Also, our measure space is partially rather than linearly ordered. It is plausible that a closer analysis of our recurrence, maybe taking more details of the measure into account, could give a better asymptotic bound.
2. In this paper we only showed that making *one* attractive switch at a time is profitable. Vöge and Jurdziński's measure allows for making *many* attractive switches and still get a strategy improvement. Showing the same (*Profitability of Multiple Switches*) for our measure (already proved in [6]) allows us to use it with many other iterative improvement algorithms, including:

   All Profitable Switches Algorithm simultaneously making all possible profitable switches,

Random Multiple Switches Algorithm simultaneously making a random subset of such switches.

The properties of our measure guarantee that these are iterative improvement algorithms [6]. Note that they are *not local-search type*, because the neigborhood size considered in one step is not polynomially bounded in the dimension n (number of vertices). Intuitively, such algorithms provide for 'more agressive' improvement compared with the ones based on single switches. For a description, theoretical and practical analysis of these algorithms see [3,2]. While no subexponential bounds have been proved for these algorithms so far, they are known to work extremely well in practice on problems with a similar structure (completely unimodal pseudoboolean functions) [2].

3. As a consequence, proving Profitability of Multiple Switches for our measure also improves the $O(2^n)$ upper bounds of Vöge and Jurdziński [24] to a better bound **B**; this is a substantial advantage when colors are few; recall that for many colors, our bound **A** is better. This also immediately sets the bound **B** for all algorithms in [2] applied to parity games, and allows to improve bounds of the form $O(2^{0.772n})$, $O(2^{0.46n})$ from [3], since the 'tighter' range of values can be exploited.
4. By investigating more exactly what properties of the measure are used in our proofs, it may be possible to invent new measures having even smaller value spaces.
5. The bound given by the measure space would also be improved if we could show that the value after a switch is 'often' improved 'much' in some vertex. For instance, it is not difficult to see that the total number of switches in vertices on a loop is polynomial, using the fact that the measure space for such vertices is polynomial in size. Therefore, one would like to show that a 'big' fraction of all switches are in vertices on a loop.
6. Our algorithms rely heavily on the subroutine for finding optimal response strategies of the adversary (Section 8). Since currently the dominating theoretical factor is the number of iterations, we are satisfied with a direct polynomial algorithm. However, in practice, when the number of iterations is usually small, the per-improvement-step efficiency may become dominating. It would be important to find a more efficient method to calculate optimal counterstrategies, possibly incremental.
7. Parity games are polynomial time equivalent to the μ-calculus Model Checking. Interpreting the new measure in the μ-calculus terms may allow for model checking the μ-calculus formulas directly by randomized subexponential iterative improvement algorithms.

Acknowledgements. The authors thank anonymous STACS'2003 referees for constructive remarks and suggestions towards improvement of the paper.

References

1. H. Björklund and S. Sandberg. Algorithms for combinatorial optimization and games adapted from linear programming. Technical Report 2003-015, Department of Information Technology, Uppsala University, 2003. `http://www.it.uu.se/research/reports/`.
2. H. Björklund, S. Sandberg, and S. Vorobyov. An experimental study of algorithms for completely unimodal optimization. Technical Report 2002-030, Department of Information Technology, Uppsala University, October 2002. `http://www.it.uu.se/research/reports/`.
3. H. Björklund, S. Sandberg, and S. Vorobyov. Optimization on completely unimodal hypercubes. Technical Report 2002-018, Uppsala University / Information Technology, May 2002. `http://www.it.uu.se/research/reports/`.

4. H. Björklund, S. Sandberg, and S. Vorobyov. A discrete subexponential algorithm for parity games. In H. Alt and M. Habib, editors, *20th International Symposium on Theoretical Aspects of Computer Science, STACS'2003*, volume 2607 of *Lecture Notes in Computer Science*, pages 663–674, Berlin, 2003. Springer-Verlag.
5. H. Björklund, S. Sandberg, and S. Vorobyov. An improved subexponential algorithm for parity games. Technical Report 2003-017, Department of Information Technology, Uppsala University, 2003. `http://www.it.uu.se/research/reports/`.
6. H. Björklund, S. Sandberg, and S. Vorobyov. On combinatorial structure and algorithms for parity games. Technical Report 2003-002, Department of Information Technology, Uppsala University, 2003. `http://www.it.uu.se/research/reports/`.
7. A. Browne, E. M. Clarke, S. Jha, D. E Long, and W Marrero. An improved algorithm for the evaluation of fixpoint expressions. *Theor. Comput. Sci.*, 178:237–255, 1997. Preliminary version in CAV'94, LNCS'818.
8. E. A. Emerson. Model checking and the Mu-calculus. In N. Immerman and Ph. G. Kolaitis, editors, *DIMACS Series in Discrete Mathematics*, volume 31, pages 185–214, 1997.
9. E. A. Emerson, C. Jutla, and A. P. Sistla. On model-checking for fragments of μ-calculus. In C. Courcoubetis, editor, *Computer Aided Verification, Proc. 5th Int. Conference*, volume 697, pages 385–396. Lect. Notes Comput. Sci., 1993.
10. E. A. Emerson and C. S. Jutla. Tree automata, μ-calculus and determinacy. In *Annual IEEE Symp. on Foundations of Computer Science*, pages 368–377, 1991.
11. B. Gärtner. A subexponential algorithm for abstract optimization problems. *SIAM Journal on Computing*, 24:1018–1035, 1995.
12. Goldwasser. A survey of linear programming in randomized subexponential time. *SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory)*, 26:96–104, 1995.
13. V. A. Gurvich, A. V. Karzanov, and L. G. Khachiyan. Cyclic games and an algorithm to find minimax cycle means in directed graphs. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 28(5):85–91, 1988.
14. M. Jurdziński. Small progress measures for solving parity games. In H. Reichel and S. Tison, editors, *17th STACS*, volume 1770 of *Lect. Notes Comput. Sci.*, pages 290–301. Springer-Verlag, 2000.
15. G. Kalai. A subexponential randomized simplex algorithm. In *24th ACM STOC*, pages 475–482, 1992.
16. W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and Computation*, 117:151–155, 1995.
17. J. Matoušek, M. Sharir, and M. Welzl. A subexponential bound for linear programming. In *8th ACM Symp. on Computational Geometry*, pages 1–8, 1992.
18. J. Matoušek, M. Sharir, and M. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16:498–516, 1996.
19. C. Papadimitriou. Algorithms, games, and the internet. In *ACM Annual Symposium on Theory of Computing*, pages 749–753. ACM, July 2001.
20. V. Petersson and S. Vorobyov. A randomized subexponential algorithm for parity games. *Nordic Journal of Computing*, 8:324–345, 2001.
21. N. Pisaruk. Mean cost cyclical games. *Mathematics of Operations Research*, 24(4):817–828, 1999.
22. H Seidl. Fast and simple nested fixpoints. *Information Processing Letters*, 59(3):303–308, 1996.
23. C. A. Tovey. Local improvement on discrete structures. In E. Aarts and Lenstra J. K., editors, *Local Search in Combinatorial Optimization*, pages 57–89. John Wiley & Sons, 1997.
24. J. Vöge and M. Jurdziński. A discrete strategy improvement algorithm for solving parity games. In E. A. Emerson and A. P. Sistla, editors, *CAV'00: Computer-Aided Verification*, volume 1855 of *Lect. Notes Comput. Sci.*, pages 202–215. Springer-Verlag, 2000.
25. K. Williamson Hoke. Completely unimodal numberings of a simple polytope. *Discrete Applied Mathematics*, 20:69–81, 1988.

26. U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.*, 158:343–359, 1996.