A Discrete Strategy Improvement Algorithm for Solving Parity Games

Jens Vöge*
Lehrstuhl für Informatik VII
RWTH Aachen
Germany

Marcin Jurdziński†
BRICS‡
University of Aarhus
Denmark

February 2000

Abstract

A discrete strategy improvement algorithm is given for constructing winning strategies in parity games, thereby providing also a new solution of the model-checking problem for the modal μ-calculus. Known strategy improvement algorithms, as proposed for stochastic games by Hoffman and Karp in 1966 and for discounted payoff games and parity games by Puri in 1995, work with real numbers and require the solution of linear programming instances involving high precision arithmetic. In the present algorithm these difficulties are avoided, by means of a discrete vertex valuation in which information about the relevance of vertices and certain distances is coded. Another advantage of the present approach is that it provides a better conceptual understanding and easier analysis of strategy improvement algorithms for parity games. However, so far we could not settle the question whether the present algorithm works in polynomial time; thus the long standing problem whether parity games can be solved in polynomial time remains open.

We also provide some evidence for superiority of the strategy improvement algorithm over other known algorithms for parity games. In particular, we have verified that the algorithm needs only linear number of strategy improvement steps on some families of difficult examples, which make other algorithms work in exponential time.

1 Introduction

The study of the computational complexity of solving parity games has two main motivations. One is that the problem is polynomial time equivalent to the modal μ-calculus model checking [6, 14], and hence better algorithms for

*Address: RWTH Aachen, Lehrstuhl für Informatik VII, D-52056 Aachen, Germany. Email: voege@informatik.rwth-aachen.de. Supported by the Deutsche Forschungsgemeinschaft (DFG), project Th 352/5-3.

†Address: BRICS, Department of Computer Science, University of Aarhus, Ny Munkegade Bldg. 540, DK-8000 Aarhus C, Denmark. Email: mju@brics.dk.

‡Basic Research in Computer Science, Centre of Danish Science Foundation.

parity games may lead to better model checkers, which is a major objective in computer aided verification.

The other motivation is the intriguing status of the problem from the point of view of structural complexity theory. It is one of the few natural problems which is in $\mathbf{NP} \cap \mathbf{co\text{-}NP}$ [6] (and even in $\mathbf{UP} \cap \mathbf{co\text{-}UP}$ [8]), and is not known to have a polynomial time algorithm, despite substantial effort of the community (see [6, 1, 13, 9] and references therein). Other notable examples of such problems include simple stochastic games [2, 3], mean payoff games [4, 15], and discounted payoff games [15]. There are polynomial time reductions of parity games to mean payoff games [12, 8], mean payoff games to discounted payoff games [15], and discounted payoff games to simple stochastic games [15]. Parity games, as the simplest of them all, seem to be the most plausible candidate for trying to find a polynomial time algorithm.

A strategy improvement algorithm has been proposed for solving stochastic games by Hoffman and Karp [7] in 1966. Puri in his PhD thesis [12] has adapted the algorithm for discounted payoff games. Puri also provided a polynomial time reduction of parity games to mean payoff games, and advocated the use of the algorithm for solving parity games, and hence for the modal μ-calculus model checking.

In our opinion Puri's strategy improvement algorithm for parity games has two drawbacks.

- The algorithm uses high precision arithmetic, and needs to solve linear programming instances: both are typically costly operations. An implementation (by the first author) of Puri's algorithm, using a linear programming algorithm of Megiddo [10], proved to be prohibitively slow.

- Solving parity games is a discrete, graph theoretic problem, but the crux of the algorithm is manipulation of real numbers, and its analysis is crucially based on continuous methods, such as Banach's fixed point theorem.

The first one makes the algorithm inefficient in practice, the other one obscures understanding of the algorithm.

Our discrete strategy improvement algorithm remedies both above-mentioned shortcomings of Puri's algorithm, while preserving the overall structure of the generic strategy improvement algorithm. We introduce discrete values (such as tuples of vertices, sets of vertices and natural numbers denoting lengths of paths in the game graph) which are being manipulated by the algorithm, instead of their encodings into real numbers. (One can show a precise relationship between behaviour of Puri's and our algorithms; we will treat this issue elsewhere.)

The first advantage of our approach is that instead of solving linear programming instances involving high precision arithmetic, we only need to solve instances of a certain purely discrete problem. Moreover, we develop an algorithm exploiting the structure of instances occurring in this context, i.e., relevance of vertices and certain distance information. In this way we get an efficient implementation of a strategy improvement algorithm with $O(nm)$ running time for one strategy improvement step, where n is the number of vertices, and m is the number of edges in the game graph.

The other advantage is more subjective: we believe that it is easier to analyze the discrete data maintained by our algorithm, rather than its subtle encodings into real numbers involving infinite geometric series [12]. The classical continuous reasoning gives a relatively clean proof of correctness of the algorithm in a more general case of discounted payoff games [12], but we think that in the case of parity games it blurs an intuitive understanding of the underlying discrete structure. However, the long standing open question whether a strategy improvement algorithm works in polynomial time [3] remains unanswered. Nevertheless, we hope that our discrete analysis of the algorithm may help either to find a proof of polynomial time termination, or to come up with a family of examples on which the algorithm requires exponential number of steps. Any of those results would mark a substantial progress in understanding the computational complexity of parity games.

So far, for all families of examples we have considered the strategy improvement algorithm needs only linear number of strategy improvement steps. Notably, a linear number of strategy improvements suffices for several families of difficult examples for which other known algorithms need exponential time.

Acknowledgement

We are indebted to Wolfgang Thomas for his invaluable advice, support, and encouragement. The second author is grateful to Wolfgang Thomas and other members of his group at RWTH Aachen for their hospitality.

2 Preliminaries

A *parity game* is an infinite two-person game played on a finite vertex-colored graph $G = (V_0, V_1, E, c)$ where $V = V_0 \mathbin{\dot\cup} V_1$ is the set of vertices and $E \subseteq V_0 \times V_1 \;\cup\; V_1 \times V_0$ the set of edges with $\forall u \in V \;\exists v \in V : uEv$, and $c : V \to \{0, \ldots, k-1\}$ is a coloring of the vertices. The two players move, in alternation, a token along the graph's edges; player 0 moves the token from vertices in V_0, player 1 from vertices in V_1. A play is an infinite vertex sequence $v_0 v_1 v_2 \ldots$ arising in this way. The decision who wins refers to the coloring c of the game graph: if the largest color which occurs infinitely often in $c(v_0)c(v_1)c(v_2)\ldots$ is even then player 0 wins, otherwise player 1 wins. One says that player 0 (resp. player 1) has a winning strategy from $v \in V$ if starting a play in v, he can force a win for arbitrary choices of player 1 (resp. player 0). By the well-known determinacy of parity games, the vertex set V is divided into the two sets W_0, W_1, called the winning sets, where W_i contains the vertices from which player i has a winning strategy.

Moreover, it is known (see, e.g., [5, 11]) that on W_i player i has a memoryless winning strategy, which prescribes the respective next move by a unique edge leaving the vertex ($\in V_i$) under consideration.

In the following we fix the basic notations for games and strategies. Let $G = (V_0, V_1, E, c)$ be a game graph with the coloring $c : V \to \{0, \ldots k-1\}$, where $V = V_0 \mathbin{\dot\cup} V_1$. Let W_0, W_1 be the winning sets of G.

A strategy for player i ($i = 0, 1$) is a function $\rho : V_i \to V_{1-i}$ such that $vE\rho(v)$ for all $v \in V_i$. A strategy for player i is called a winning strategy on $W \subseteq V$ if player i wins every play starting in W. A *winning strategy* for player i is a winning strategy on the winning set W_i.

We set

$$V_+ = \{v \in V \mid c(v) \textit{ is even}\} \text{ and } V_- = \{v \in V \mid c(v) \textit{ is odd}\}$$

and we call vertices in V_+ positive and those in V_- negative.

We introduce two orderings of V, the relevance ordering $<$ and the reward ordering $\prec$. The relevance order is a total order extending the pre-order given by the coloring, i.e., such that $c(u) < c(v)$ implies $u < v$. (So higher colors indicate higher relevance.)

By $\inf(\pi)$ we denote the set of vertices occurring infinitely often in the play π. Referring to $<$ over V, we can reformulate the winning condition for player 0 in a play π as

$$\max\nolimits_<(\inf(\pi)) \in V_+.$$

Another ordering, the reward order $\prec$, indicates the value of a vertex as seen from player 0. The lowest reward originates from the vertex in V_- of highest relevance, the highest from the vertex in V_+ of highest relevance. Formally, we define:

$$u \prec v \iff (u < v \land v \in V_+) \lor (v < u \land u \in V_-).$$

We extend the reward order $\prec$ to an order on sets of vertices. If $P, Q \in 2^V$ are distinct, the vertex v of highest relevance in the symmetric difference $P \triangle Q$ decides whether $P \prec Q$ or $Q \prec P$: If $v \in V_+$ then the set containing v is taken as the higher one, if $v \in V_-$ then the set not containing v is taken as the higher one. Formally:

$$P \prec Q \iff P \neq Q \land \max\nolimits_<(P \triangle Q) \in Q \triangle V_-.$$

Note that we use the same symbol $\prec$ for vertices and for sets of vertices.

We also need a coarser version $\prec_w$ of $\prec$, using a reference vertex w and taking into account only P- and Q-vertices of relevance $\geq w$:

$$P \prec_w Q \iff P \cap \{x \in V | x \geq w\} \prec Q \cap \{x \in V | x \geq w\},$$

and the corresponding equivalence relation:

$$P \sim_w Q \iff P \preceq_w Q \land Q \preceq_w P.$$

3 Game graph valuations

In this section we present the terminology and main ideas underlying the approximative construction of winning strategies. The plays considered in the sequel are always induced by two given strategies σ and τ for player 0 and 1 respectively. Any such pair σ, τ determines from any vertex a play ending in a loop L. The first subsection 3.1 introduces certain values called play profiles for such plays. A play profile combines three data: the most relevant vertex of the loop L, the vertices occurring in the play that are more relevant than the most relevant vertex in the loop, and the number of all vertices that are visited before the most relevant vertex of the loop. The order $\prec$ above is extended to play profiles (so that $\prec$-higher play profiles are more advantageous for player 0). An arbitrary assignment of play profile values to the vertices is called a valuation.

Subsection 3.2 gives a certain condition when a valuation originates from a strategy pair (σ, τ) as explained. Such valuations are called locally progressive.

The algorithm will produce successively such locally progressive valuations. In each step, a valuation is constructed from a strategy σ of player 0 and an 'optimal' response strategy τ of player 1. In subsection 3.3 this notion of optimality is introduced. We show that a valuation which is both optimal for players 0 and 1 represents the desired solution of our problem: a pair of winning strategies for the two players.

The last subsection 3.4 will explain the nature of an approximation step, from a given valuation φ to a (next) 'response valuation' φ'. One should imagine that player 0 picks edges to vertices with $\prec$-maximal values given by φ, and that player 1 responds as explained above, by a locally progressive valuation φ'. The key lemma says that uniformly φ' will majorize φ (i.e., $\varphi(v) \preceq \varphi'(v)$ for all vertices $v \in V$). This will be the basis for the termination proof because equality $\varphi = \varphi'$ will imply that φ is already optimal for both players.

3.1 Play profiles

Let $G = (V_0, V_1, E, c)$ be a game graph. Let Π be the set of plays that can be played on this graph. We define the function $w : \Pi \to V$ which computes the most relevant vertex that is visited infinitely often in a play π, i.e., $w(\pi) = \max_{<} \inf(\pi)$. Furthermore, we define a function $\alpha : \Pi \to 2^V$ which computes the vertices that are visited before the first occurrence of $w(\pi)$:

$$\alpha(\pi) = \{u \in V \mid \exists i \in \mathbb{N}_0 : \pi_i = u \wedge \forall k \in \mathbb{N}_0 : k \leq i \Longrightarrow \pi_k \neq w(\pi)\}.$$

We are interested in three characteristic values of a play π:

1. The most relevant vertex u_π that is visited infinitely often: $u_\pi = w(\pi)$.

2. The set of vertices P_π that are more relevant than u_π and visited before the first visit of u_π: $P_\pi = \alpha(\pi) \cap \{v \in V \mid v > w(\pi)\}$.

3. The number of vertices visited before the first visit of u_π: $e_\pi = |\alpha(\pi)|$.

We call such a triple (u_π, P_π, e_π) a *play profile*. It induces an equivalence relation of plays on the given game graph. By definition each profile of a play induced by a pair of strategies belongs to the following set:

$$\mathcal{D} = \{(u, P, e) \in V \times 2^V \times \{0, \ldots, |V| - 1\} \mid \forall v \in P : u < v \ \wedge \ |P| \leq e\}.$$

The set $\mathcal{D}$ is divided into profiles of plays that are won by player 0 respectively 1:

$$\mathcal{D}_0 = \{(u, P, e) \in \mathcal{D} \mid u \in V_+\} \ \text{ and } \ \mathcal{D}_1 = \{(u, P, e) \in \mathcal{D} \mid u \in V_-\}.$$

We define a linear ordering $\prec$ on $\mathcal{D}$. A play profile is greater with respect to this ordering if the plays it describes are 'better' for player 0, respectively smaller if they are better for player 1. Let $(u,P,e),(v,Q,f) \in \mathcal{D}$:

$$(u,P,e) \prec (v,Q,f) \iff \begin{cases} & u \prec v \\ \vee & (u = v \wedge P \prec Q) \\ \vee & (u = v \wedge P = Q \wedge v \in V_- \wedge e < f) \\ \vee & (u = v \wedge P = Q \wedge v \in V_+ \wedge e > f). \end{cases}$$

The idea behind the last two clauses is that in case $u = v$ and $P = Q$ it is more advantageous for player 0 to have a shorter distance to the most relevant vertex v if $v \in V_+$ (case $f < e$), resp. a longer distance if $v \in V_-$ (case $f > e$).

For the subsequent lemmata we need a coarser ordering $\prec_w$ of play profiles:

$$(u,P,e) \prec_w (v,Q,f) \iff u \prec v \ \vee \ (u = v \ \wedge \ P \prec_w Q).$$

In this ordering neither the distance to the most relevant vertex of the loop nor vertices on the path that are less relevant then w are taken into account. The corresponding equivalence relation $\sim_w$ belonging to $\prec_w$ is:

$$(u,P,e) \sim_w (v,Q,f) \iff u = v \ \wedge \ P \sim_w Q.$$

3.2 Valuations

A *valuation* is a function $\varphi : V \to \mathcal{D}$ which labels each vertex with a play profile.

We are interested in valuations where all play profiles $\varphi(v)$ $(v \in V)$ are *induced* by the same pair of strategies. By definition, an initial vertex v and two strategies σ for player 0 and τ for player 1 determine a play $\pi_{v,\sigma,\tau}$. For a pair of strategies (σ,τ) we can define the *valuation induced by* (σ,τ) to be the function φ which maps $v \in V$ to the play profile of $\pi_{v,\sigma,\tau}$. This valuation φ assigns to each vertex $v \in V$ the play profile of the play starting in v played with respect to σ and τ. To refer to the components of a play profile $\varphi(v) = (u,P,e)$ we write φ_0, φ_1 and φ_2, where $\varphi_0(v) = u$, $\varphi_1(v) = P$ and $\varphi_2(v) = e$. We call u the most relevant vertex of play profile $\varphi(v)$ (or of v, if φ is clear from the context).

The play profiles in a valuation induced by a strategy pair are related as follows: Let σ,τ be strategies and φ their corresponding valuation. Let $x,y \in V$ be two vertices with $\sigma(x) = y$ or $\tau(x) = y$, i.e., in a play induced by σ and τ a move proceeds from x to y. It follows immediately that $\varphi_0(x) = \varphi_0(y)$. We can distinguish the following cases:

1. Case $x < \varphi_0(x)$: Then $\varphi_1(x) = \varphi_1(y)$ and $\varphi_2(x) = \varphi_2(y) + 1$.
2. Case $x = \varphi_0(x)$: By definition of φ we have: $\varphi_1(x) = \emptyset$ and $\varphi_2(x) = 0$. Furthermore $\varphi_1(y) = \emptyset$, because there are no vertices on the loop through x that are more relevant than x.
3. Case $x > \varphi_0(x)$: Then $\varphi_1(x) = \{x\} \mathbin{\dot\cup} \varphi_1(y)$ and $\varphi_2(x) = \varphi_2(y) + 1$.

These conditions define what we call the *φ-progress* relation $\lhd_\varphi$. If an edge leads from x to y then $x \lhd_\varphi y$ will mean that the φ-value is correctly updated when passing from x to y in a play. Formally we define for $x, y \in V$, assuming $\varphi(x) = (u, P, e)$, $\varphi(y) = (v, Q, f)$:

$$\begin{array}{lllllllll} x \lhd_\varphi y & \iff & u = v \wedge (& & (x = u & \wedge & P = Q = \emptyset & \wedge & e = 0) \\ & & & \vee & (x < u & \wedge & P = Q & \wedge & e = f + 1) \\ & & & \vee & (x > u & \wedge & P = Q \mathbin{\dot\cup} \{x\} & \wedge & e = f + 1)). \end{array}$$

The following lemma is obvious:

Lemma 1 *Let φ be a valuation and assume $v \lhd_\varphi \sigma(v)$ and $v \lhd_\varphi \tau(v)$ for all $v \in V$. Then (σ, τ) induces φ.*

Note that a valuation φ may be induced by several pairs (σ, τ) of strategies so that several plays starting in v with play profile $\varphi(v)$ may exist. We now characterize those valuations which are induced by some strategy pair (σ, τ). A valuation φ is called *locally progressive* if

$$\forall u \in V \; \exists v \in V : uEv \wedge u \lhd_\varphi v.$$

Lemma 2 *Let $G = (V_0, V_1, E, c)$ be a game graph. Let φ be a valuation for G. Then φ is a locally progressive valuation iff there exists a strategy σ for player 0 and a strategy τ for player 1 such that φ is a valuation induced by (σ, τ).*

Proof: Let $G = (V_0, V_1, E, c)$ be a game graph. Let φ be a valuation for G.

If φ is a locally progressive valuation then we can take a strategy σ for player 0 such that $v \lhd_\varphi \sigma(v)$ for $v \in V_0$ and a strategy τ for player 1 such that $v \lhd_\varphi \tau(v)$ for $v \in V_1$. Then φ is a valuation induced by (σ, τ).

If φ is a valuation induced by a strategy pair (σ, τ) it is clearly locally progressive. $\square$

We call a strategy $\rho : V_i \to V_{1-i}$ $(i \in \{0, 1\})$ *compatible* with the valuation φ if $\forall v \in V : v \lhd_\varphi \rho(v)$. From Lemma 2 it follows that for a locally progressive valuation φ at least one strategy for each player is compatible with φ.

3.3 Optimal valuations

A valuation φ is called optimal for player 0 if the φ-progress relation $x \lhd_\varphi y$ only applies to edges xEy where the value $\varphi(y)$ is $\preceq$-maximal among the E-successors of x (i.e., among the values $\varphi(z)$ with xEz). However a weaker requirement is made if x is the most relevant vertex associated to x via φ (i.e., $\varphi_0(x) = x$). In this case we discard the distance component φ_2; formally we replace $\preceq$ by $\preceq_x$. (Recall that $(u, P, e) \preceq_x (v, Q, f)$ holds if $u \prec v$ holds, or $u = v$ and $P \preceq_x Q$, i.e., e and f are not taken into account.) Formally φ is called *optimal for player* 0 if for all $x \in V_0$ and $y \in V_1$ with an edge xEy:

$$x \lhd_\varphi y \iff \forall z \in V_1 : xEz \Rightarrow \varphi(z) \preceq \varphi(y) \;\vee\; (\varphi_0(x) = x \;\wedge\; \varphi(z) \preceq_x \varphi(y)).$$

In the last case, the vertex y succeeds x in the loop of a play given by a strategy pair (σ, τ) which is compatible with φ; of course, there may be several such y with $x \lhd_\varphi y$. Similarly φ is called *optimal for player* 1 if for all $x \in V_1$ and $y \in V_0$ with an edge xEy:

$$x \lhd_\varphi y \iff \forall z \in V_0 : xEz \Rightarrow \varphi(y) \preceq \varphi(z) \ \vee\ (\varphi_0(x) = x \ \wedge\ \varphi(y) \preceq_x \varphi(z)).$$

A valuation that is optimal for both players is called *optimal valuation.*

It is useful to note the following fact: If φ is optimal for player 0, then (since $\preceq_x$ is a weakening of $\preceq$) $x \lhd_\varphi y$ implies $\varphi(z) \preceq_x \varphi(y)$ for xEz. Similarly if φ is optimal for player 1 then $x \lhd_\varphi y$ implies $\varphi(y) \preceq_x \varphi(z)$ for xEz.

The optimal valuations are closely related to the desired solution of a game, namely a pair of winning strategies. Consider a valuation φ which is optimal for player 0, and let W_1 be the set of vertices u whose φ-value $\varphi(u)$ is in $\mathcal{D}_1$ (i.e., the most relevant vertex $\varphi_0(u)$ associated to u is in V_-, signalling a win of player 1). Any strategy for player 1 compatible with φ turns out to be a winning strategy for him on W_1, whatever strategy player 0 chooses *independently* of φ. Applying this symmetrically for both players we shall obtain a pair of winning strategies.

Lemma 3 *Let $G = (V_0, V_1, E, c)$ be a game graph. Let $i \in \{0,1\}$ and φ be a locally progressive valuation of G which is optimal for player i. Let $W_{1-i} = \{v \in V \mid \varphi(v) \in \mathcal{D}_{1-i}\}$. Then the strategies for player $1-i$ compatible with φ are winning strategies on W_{1-i} (against an arbitrary strategy of player i).*

Proof: Let us consider the case of player $i = 0$. We are given a game graph $G = (V_0, V_1, E, c)$ and a locally progressive valuation φ which is optimal for player 0. Let $W_1 = \{v \in V \mid \varphi(v) \in \mathcal{D}_1\}$. Let τ be a strategy for player 1 compatible with φ and σ be an arbitrary strategy for player 0. Let π be a play starting in $v \in W_1$, played with respect to σ and τ. We have to prove that player 1 wins the play π.

Let $L = \inf(\pi)$, the loop of π, and $\nu = \sigma \cup \tau$. The function ν is the successor function on π. Let $w = max_< L$ and $x = \varphi_0(w)$. Note that $\varphi_0(w)$ may be different from w, since player 0 may not play compatible with φ (the loop L may be different from the ones coded in φ).

The valuation φ is optimal for player 0 and player 1 plays a strategy compatible with φ. We start with two auxiliary claims:

$$\varphi_0(\nu(u)) \preceq \varphi_0(u) \qquad \text{all for } u \in V \tag{1}$$

from which we get, by transitivity,

$$\varphi_0(u) = x \qquad \text{for all } u \in L;$$

and secondly

$$\varphi_1(\nu(u)) \preceq \varphi_1(u) \setminus \{u\} \qquad \text{for all } u \in L \tag{2}$$

To verify (1) in case $u \in V_0$ we use that one edge uEy exists with $u \lhd_\varphi y$ (φ is locally progressive), so that by optimality of φ for player 0, we have

$\varphi(\nu(u)) \preceq_u \varphi(y)$ (note that $\preceq$ implies $\preceq_u$). A fortiori, by definition of $\preceq_u$, we have $\varphi_0(\nu(u)) \preceq \varphi_0(y)$.
Also from $u \lhd_\varphi y$ we get $\varphi_0(u) = \varphi_0(y)$, which yields claim (1) in case of $u \in V_0$. If $u \in V_1$, apply the fact that player 1 plays compatible with φ, so $u \lhd_\varphi \nu(u)$, which implies $\varphi_0(u) = \varphi_0(\nu(u))$ and hence (1).

Claim (2) is clear in case $u \in V_1$, since player 1 plays compatible with φ (and because of $\varphi_0(\nu(u)) = \varphi_0(u)$ we have $\varphi_1(\nu(u)) = \varphi_1(u) \setminus \{u\}$). For $u \in V_0$ let $y \in V_1$ with $u \lhd_\varphi y$ as above for claim (1). Then we have $\varphi_0(y) = \varphi_0(u)$ and $\varphi_1(y) = \varphi_1(u) \setminus \{u\}$. Because φ is optimal for player 0 we have $\varphi(\nu(u)) \preceq_x \varphi(y)$. It follows $\varphi_1(\nu(u)) \preceq \varphi_1(u) \setminus \{u\}$, because from claim (1) we inferred on L that the φ_0–value is x and hence $\varphi_0(\nu(u)) = \varphi_0(u)$.

We show the claim of the lemma in two cases, by verifying $w \in V_-$:

Case $x < w$: From $w = \max_< L$ and (2) we obtain for all $u \in L$ different from w that $\varphi(\nu(u)) \preceq_w \varphi(u)$, and so $\varphi(w) \preceq_w \varphi(\nu(w))$. Let $y \in V$ with wEy and $w \lhd_\varphi y$. We claim:

$$\varphi(\nu(w)) \preceq_w \varphi(y) \tag{3}$$

from which, by transitivity, we will know $\varphi(w) \preceq_w \varphi(y)$. To verify (3) in case of $w \in V_0$ we use as for claim (1) that wEy and $w \lhd_\varphi y$, so that by optimality of φ for player 0, we have $\varphi(\nu(w)) \preceq_w \varphi(y)$. If $w \in V_1$, apply the fact that player 1 plays compatible with φ, so $w \lhd_\varphi \nu(w)$, which implies together with $w \lhd_\varphi y$ that $\varphi(\nu(w)) \sim_w \varphi(y)$.

As mentioned from claim (3) we know $\varphi(w) \preceq_w \varphi(y)$, and by our case $x < w$ and $x = \varphi_0(w)$, even $\varphi(w) \prec_w \varphi(y)$. From wEy and $w \lhd_\varphi y$ we obtain $\varphi_0(w) = \varphi_0(y)$ and hence must have $\varphi_1(w) \prec_w \varphi_1(y)$. Again by wEy and $w \lhd_\varphi y$, we can specify $\varphi_1(w)$ as $\varphi_1(y) \dot\cup \{w\}$, so we get even $\varphi_1(w) \prec \varphi_1(y)$, and the symmetric difference of the two sets is $\{w\}$. From the definition of $P \prec Q$ we obtain $w \in V_-$.

Case $w \leq x$: First we make a preliminary observation. Because $w = max_< L$ it follows that for all $u \in L$ we have $u \leq x$, so for all $u \in L : u \notin \varphi_1(u)$ and with (2) for all $u \in L$: $\varphi(\nu(u)) \preceq_x \varphi(u)$. By transitivity and because L is a loop we get for all $u, u' \in L$: $\varphi(u) \sim_x \varphi(u')$. In particular, the φ_0 and φ_1–components of u, u' agree.

Our aim is to show that x (the φ_0–value of the most relevant element w of the loop L) belongs to L, in order to get $x = w$. We do this by choosing an element $y \in L$ and showing $y = x$: Let $y \in L$ be a vertex with $\varphi_2(\nu(y)) = max_{u \in L} \varphi_2(u)$. For this y pick z with yEz and $y \lhd_\varphi z$.

We show $x = y$, first in case $\varphi_2(\nu(y)) > \varphi_2(z)$. By $y \leq w$ (w is most relevant on L) and our case $w \leq x$, we have $y \leq x$. We have $\varphi(\nu(y)) \sim_x \varphi(z)$. (To see this, note $\varphi_0(z) = \varphi_0(y) = \varphi_0(\nu(y))$ by $y \lhd_\varphi z$ and by (1). Also $\varphi_1(z) = \varphi_1(y) \sim_x \varphi_1(\nu(y))$: the equality follows from the first two clauses of the definition of $y \lhd_\varphi z$, using $\varphi_0(y) = \varphi_0(z)$ and $y \leq x$; the $\sim_x$ equivalence was noted above for elements of the loop L.) We have to show:

$$\varphi_2(\nu(y)) > \varphi_2(z) \Longrightarrow x = y.$$

In case of $y \in V_1$ we have $y \lhd_\varphi \nu(y)$, because player 1 plays compatible with φ. Together with $y \lhd_\varphi z$ and our assumption $\varphi_2(\nu(y)) > \varphi_2(z)$ we get by the definition of $\lhd_\varphi$ that $\varphi_0(y) = y$. So we have $x = \varphi_0(y) = y$.

Now assume $y \in V_0$ and remember that we consider play π starting in $v \in W_1$ i.e. $\varphi_0(v) \in \mathcal{D}_1$, i.e. $\varphi_0(v) \in V_-$. By claim (1) and transitivity we have: $\varphi_0(w) \preceq \varphi_0(v)$. So we have $x = \varphi_0(w) \in V_-$. With $x \in V_-$ and the assumption $\varphi_2(z) < \varphi_2(\nu(y))$ we get $\varphi(z) \prec \varphi(\nu(y))$, because the φ_0– and φ_1–values we have shown to agree. Using that φ is optimal for player 0 the definition of $\lhd_\varphi$ leads to $y = \varphi_0(y) = x$.

In the other case $\varphi_2(\nu(y)) \leq \varphi_2(z)$ we get, by maximality of $\varphi_2(\nu(y))$ on L, $\varphi_2(y) \leq \varphi_2(\nu(y)) \leq \varphi_2(z)$. So the distance from y to x in plays compatible with φ is at most the distance of z to x. Since $y \lhd_\varphi z$, we must have $y = x$.

So we have $y = x$ and hence $x \in L$. With $w \leq x$ and $w = max_< L$ we get $w = x \in V_-$.

The case $i = 1$ is dual. □

Applying this lemma symmetrically for both players leads to the following.

Theorem 4 *Let $G = (V_0, V_1, E, c)$ be a game graph. Let φ be an optimal locally progressive valuation of G. Then all strategies compatible with φ are winning strategies.*

3.4 Improving valuations

Given two locally progressive valuations φ and φ', we call φ' *improved for player* 0 *with respect to* φ if

$$\forall x \in V_0\ \exists y \in V : xEy \ \wedge\ x \lhd_{\varphi'} y \ \wedge\ \forall z \in V_1 : xEz \Rightarrow \varphi(z) \preceq \varphi(y).$$

A locally progressive valuation φ', which is improved for player 0 with respect to φ, can be constructed from a given locally progressive valuation φ by extracting a strategy $\sigma : V_0 \rightarrow V_1$ for player 0 that chooses maximal E-successors with respect to φ and constructing a locally progressive valuation φ' such that σ is compatible with φ'.

Lemma 5 *Let $G = (V_0, V_1, E, c)$ be a game graph. Let φ be a locally progressive valuation optimal for player* 1 *and φ' a locally progressive valuation that is improved for player* 0 *with respect to φ. Then for all $v \in V$: $\varphi(v) \preceq \varphi'(v)$.*

Proof: Let $G = (V_0, V_1, E, c)$ be a game graph. Let φ be a locally progressive valuation optimal for player 1 and φ' a locally progressive valuation that is improved for player 0 with respect to φ.

For the proof we shall start from a special strategy pair (σ, τ), that induces φ'. Then we shall show the claim by induction over the φ'_2–values (i.e. the distance component of the play profiles) of the vertices, referring to plays played according to (σ, τ).

Since φ' is improved for player 0 with respect to φ, we may choose a strategy $\sigma : V_0 \rightarrow V_1$ with

$$vE\sigma(v) \ \wedge\ v \lhd_{\varphi'} \sigma(v) \ \wedge\ \forall x \in V_1 : vEx \Rightarrow \varphi(x) \preceq \varphi(\sigma(v)),$$

Note that by this definition σ is compatible with φ'. There exists a strategy τ for player 1 that is compatible with φ', because φ' is locally progressive. In the following we combine the strategy pair (σ, τ) to a successor function $\nu = \sigma \cup \tau$.

First we prove an important property of ν. For all $v \in V$ and all $x \in V$ we have:

$$vEx \land v \lhd_\varphi x \implies \varphi(x) \preceq \varphi(\nu(v)) \ \lor \ (\varphi_0(v) = v \ \land \ \varphi(x) \preceq_v \varphi(\nu(v))). \quad (4)$$

If $v \in V_0$, we have $vEx \implies \varphi(x) \preceq \varphi(\nu(v))$ by definition of σ, which implies the claim. Case $v \in V_1$: We know, that φ is optimal for player 1, which means that for all $v \in V_1$ and $x \in V_0$ with an edge vEx:

$$v \lhd_\varphi x \iff \forall y \in V_0 : vEy \Rightarrow \varphi(x) \preceq \varphi(y) \ \lor \ (\varphi_0(v) = v \ \land \ \varphi(x) \preceq_v \varphi(y)).$$

Taking $\nu(v)$ for y, claim (4) is implied.

In the following we also need a weaker version of claim (4):

$$vEx \land v \lhd_\varphi x \implies \varphi(x) \preceq_v \varphi(\nu(v)) \qquad \text{for all } v, x \in V \quad (5)$$

This follows from (4), because $\preceq$ implies $\preceq_v$.

Now we prove the claim $\varphi(v) \preceq \varphi'(v)$ of the lemma by induction over $\varphi'_2(v)$. Let $v \in V$.

Induction base $\varphi'_2(v) = 0$:

Let π be a play played according to ν starting in v. Let $L = \inf(\pi)$ be the loop of this play. From $\varphi'_2(v) = 0$ we know by definition of a locally progressive valuation φ' that $\varphi'(v) = (v, \emptyset, 0)$. Because π is a play played according to a pair of strategies each compatible with φ', we know that v is the most relevant vertex in the loop of this play:

$$v = \max\nolimits_< L \quad (6)$$

We distinguish three cases for the value of v:

If $\varphi_0(v) = v$ then we have $\varphi(v) = (v, \emptyset, 0) = \varphi'(v)$, which implies $\varphi(v) \preceq \varphi'(v)$.

Case $\varphi_0(v) < v$: First we prove for all $u \in L$ distinct from v: $\varphi(u) \sim_v \varphi(\nu(u))$. With this we get $\varphi(\nu(v)) \preceq_v \varphi(v)$. From that we can derive $v \in V_+$, which will yield the claim of the lemma.

Now we prove for all $u \in L$ distinct from v:

$$\varphi(u) \sim_v \varphi(\nu(u)).$$

Let $u \in L \setminus \{v\}$. Because φ is locally progressive there exists a vertex $x \in V$ with uEx and $u \lhd_\varphi x$. From the definition of $u \lhd_\varphi x$ we get by (6), which implies $u < v$, that $\varphi(u) \sim_v \varphi(x)$. From claim (5) we get $\varphi(x) \preceq_u \varphi(\nu(u))$. So we have $\varphi(u) \sim_v \varphi(x) \preceq_u \varphi(\nu(u))$. By $u < v$ we know that $\preceq_u$ implies $\preceq_v$ and hence $\varphi(u) \preceq_v \varphi(\nu(u))$.

From this we get by transitivity: $\varphi(\nu(v)) \preceq_v \varphi(\nu^{|L|}(v))$, because the play played according to ν forms a loop with vertices set L, which means $\nu^i(v) \neq v$ for all i with $1 \leq i < |L|$. We also have $\nu^{|L|}(v) = v$. So we get:

$$\varphi(\nu(v)) \preceq_v \varphi(v). \tag{7}$$

Because φ is locally progressive there is a vertex $y \in V$ with vEy and $v \lhd_\varphi y$. With the assumption $\varphi_0(v) < v$ of the present case we get by definition of $v \lhd_\varphi y$ that $\varphi_0(v) = \varphi_0(y)$ and $\varphi_1(v) = \varphi_1(y) \dot\cup \{v\}$. From vEy and $v \lhd_\varphi y$ we get by (5) that $\varphi(y) \preceq_v \varphi(\nu(v))$, and by (7) (with transitivity), also $\varphi(y) \preceq_v \varphi(v)$. By $\varphi_0(v) = \varphi_0(y)$ we get $\varphi_1(y) \preceq_v \varphi_1(v)$. Together with $\varphi_1(v) = \varphi_1(y) \dot\cup \{v\}$ we get $\varphi_1(y) \preceq_v \varphi_1(y) \dot\cup \{v\}$. This implies, by definition of $\preceq_v$, that $v \in V_+$.

From $v \in V_+$ and the assumption $\varphi_0(v) < v$ and the definition of $\prec$ we have $\varphi_0(v) \prec v$. With the induction base assumption $\varphi'_2(v) = 0$, which implies (by definition of the locally progressive valuation φ') that $\varphi'_0(v) = v$, we get $\varphi_0(v) \prec v = \varphi'_0(v)$, which implies the claim of the lemma $\varphi(v) \preceq \varphi'(v)$.

Case $\varphi_0(v) > v$: First we prove that $\varphi_0(u) = \varphi_0(\nu(u))$ for all $u \in L$. We use it to show $u < \varphi_0(u)$ for all $u \in L$. By this and choosing a special vertex u in the loop L with a $\preceq$-smaller successor we shall derive $\varphi_0(v) \in V_-$, which leads directly to the lemma's claim.

First let us show the claim

$$\varphi_0(u) = \varphi_0(\nu(u)) \qquad \text{for all } u \in L.$$

Let $u \in L$. Because φ is locally progressive there is $x \in V$ with uEx and $u \lhd_\varphi x$. By (5) we know $\varphi(x) \preceq_u \varphi(\nu(u))$ which implies $\varphi_0(x) \preceq \varphi_0(\nu(u))$. By definition of $u \lhd_\varphi x$ we have $\varphi_0(u) = \varphi_0(x)$. Together with $\varphi_0(x) \preceq \varphi_0(\nu(u))$ we get the claim $\varphi_0(u) \preceq \varphi_0(\nu(u))$.

Because ν forms the loop with vertex set L we have $\nu^{|L|}(u) = u$ for all $u \in L$. By transitivity we get:

$$\forall u \in L : \varphi_0(u) = \varphi_0(\nu(u)).$$

and because $v \in L$ we get from the previous claim:

$$\forall u \in L : \varphi_0(u) = \varphi_0(v). \tag{8}$$

Now we show:

$$u < \varphi_0(u) \qquad \text{for all } u \in L \tag{9}$$

Let $u \in L$. We know $\varphi_0(u) = \varphi_0(v)$ from (8). We have $v < \varphi_0(v)$ by assumption of the present case, so $v < \varphi_0(u)$. By (6) we have $u \leq v$ and get finally: $u < \varphi_0(u)$.

Because ν forms the loop with vertices L there is $u \in L$ with: $\varphi(\nu(u)) \preceq \varphi(u)$ (the negation leads, by transitivity of $\succ$, immediately to a contradiction). Because φ is locally progressive we may pick $x \in V$ with uEx and $u \lhd_\varphi x$. By (4) we get

$$\varphi(x) \preceq \varphi(\nu(u)) \ \vee \ (\varphi_0(u) = u \ \wedge \ \varphi(x) \preceq_u \varphi(\nu(u))).$$

and by (9) we hence obtain

$$\varphi(x) \preceq \varphi(\nu(u)).$$

By assumption we have $\varphi(\nu(u)) \preceq \varphi(u)$, which leads to:

$$\varphi(x) \preceq \varphi(u).$$

Because $u \lhd_\varphi x$ we have $\varphi_0(x) = \varphi_0(u)$ and because $u < \varphi_0(u)$ we have $\varphi_1(x) = \varphi_1(u)$. So we get

$$\varphi_2(x) \preceq \varphi_2(u).$$

From $u \lhd_\varphi x$ we have $\varphi_2(u) = \varphi_2(x) + 1$, so $\varphi_2(x) < \varphi_2(u)$. Together with $\varphi_2(x) \preceq \varphi_2(u)$ this implies $\varphi_0(u) \in V_-$ and by (8) also $\varphi_0(v) \in V_-$. With $\varphi_0(v) \in V_-$ and $\varphi_0(v) > v$ we get $\varphi_0(v) \prec v$. So we have $\varphi_0(v) \prec v = \varphi'_0(v)$, and so we get the lemma's claim $\varphi(v) \preceq \varphi'(v)$.

Induction step:

Induction hypothesis: $\varphi(v) \preceq \varphi'(v)$ is true for all $v \in V$ with $\varphi'_2(v) = k$.

To show the induction claim, let $\varphi'_2(v) = k+1$: We choose a $\lhd_\varphi$-successor x and a $\lhd_{\varphi'}$-successor y of v: Because φ is locally progressive we may pick $x \in V$ with vEx and $v \lhd_\varphi x$. Let $y = \nu(v)$. So we have vEy and $v \lhd_{\varphi'} y$, because the strategies combined in ν are both compatible with φ'. In the following the profiles $\varphi(x)$ and $\varphi'(y)$ are compared, using (4) and induction hypothesis. Using $v \lhd_\varphi x$ and $v \lhd_{\varphi'} y$ the relation between $\varphi(x)$ and $\varphi'(y)$ will be transferred to $\varphi(v)$ and $\varphi'(v)$.

From vEx and $v \lhd_\varphi x$ and $y = \nu(v)$ we get by (5) that $\varphi_0(x) \preceq \varphi_0(y)$. By induction hypothesis we have $\varphi_0(y) \preceq \varphi'_0(y)$. So we get by transitivity:

$$\varphi_0(x) \preceq \varphi'_0(y).$$

First we consider the simpler case $\varphi_0(x) \prec \varphi'_0(y)$: From $v \lhd_\varphi x$ we have $\varphi_0(v) = \varphi_0(x)$ and from $v \lhd_{\varphi'} y$ we have $\varphi'_0(v) = \varphi'_0(y)$. So we get $\varphi_0(v) = \varphi_0(x) \prec \varphi'_0(y) = \varphi'_0(v)$ and this implies the claim of the lemma $\varphi(v) \preceq \varphi'(v)$.

It remains the case $\varphi_0(x) = \varphi'_0(y)$. We have $\varphi_0(v) = \varphi_0(x) = \varphi'_0(y) = \varphi'_0(v)$. By $\varphi'_2(v) = k+1$ we obtain $\varphi_0(v) \neq v$. So we get, by (4), that $\varphi(x) \preceq \varphi(y)$. By induction hypothesis we have $\varphi(y) \preceq \varphi'(y)$. So we get by transitivity:

$$\varphi(x) \preceq \varphi'(y). \tag{10}$$

In the following we transfer this relation to $\varphi(v)$ and $\varphi'(v)$. Our first step is to show:

$$\varphi_1(v) \preceq \varphi'_1(v).$$

From the assumption $\varphi'_2(v) = k+1$ of the induction step we know $\varphi_0(v) \neq v$. So the two cases $\varphi_0(v) > v$ and $\varphi_0(v) < v$ remain:
If $\varphi_0(v) > v$ we get by definition of $v \lhd_\varphi x$ that $\varphi_1(v) = \varphi_1(x)$ and by $v \lhd_{\varphi'} y$ that $\varphi'_1(v) = \varphi'_1(y)$. With (10) we have:

$$\varphi_1(v) = \varphi_1(x) \preceq \varphi'_1(y) = \varphi'_1(v).$$

If $\varphi_0(v) < v$ we get by definition of $v \lhd_\varphi x$ that $\varphi_1(v) = \varphi_1(x) \dot\cup \{v\}$ and by $v \lhd_{\varphi'} y$ that $\varphi'_1(v) = \varphi'_1(y) \dot\cup \{v\}$. So $v \notin \varphi_1(x)$ and $v \notin \varphi'_1(y)$. With this we get from (10) also $\varphi_1(x) \dot\cup \{v\} \preceq \varphi'_1(y) \dot\cup \{v\}$. Taking this together we have:

$$\varphi_1(v) = \varphi_1(x) \dot\cup \{v\} \preceq \varphi'_1(y) \dot\cup \{v\} = \varphi'_1(v).$$

Having verified $\varphi_1(v) \preceq \varphi'_1(v)$ (in the present case $\varphi_0(x) = \varphi'_0(y)$) we consider two subcases. If $\varphi_1(v) \prec \varphi'_1(v)$ the claim $\varphi(v) \preceq \varphi'(v)$ of the lemma follows, because we are in case $\varphi_0(x) = \varphi'_0(y)$, which means $\varphi_0(v) = \varphi'_0(v)$.

It remains subcase $\varphi_1(v) = \varphi'_1(v)$. By definition of $v \lhd_\varphi x$ we have $\varphi_1(x) = \varphi_1(v) \setminus \{v\}$ and by $v \lhd_{\varphi'} y$ we have $\varphi'_1(v) \setminus \{v\} = \varphi'_1(y)$. So $\varphi_1(x) = \varphi'_1(y)$. Considering that we are in case $\varphi_0(x) = \varphi'_0(y)$ we get from the definition of (10): If $\varphi_0(v) \in V_-$ then $\varphi_2(x) \leq \varphi'_2(y)$ and we get

$$\varphi_2(v) = \varphi_2(x) + 1 \leq \varphi'_2(y) + 1 = \varphi'_2(v).$$

If $\varphi_0(v) \in V_+$ then $\varphi_2(x) \geq \varphi'_2(y)$ and we get the converse relation

$$\varphi_2(v) = \varphi_2(x) + 1 \geq \varphi'_2(y) + 1 = \varphi'_2(v).$$

Keeping in mind our case $\varphi_0(v) = \varphi'_0(v)$ and subcase $\varphi_1(v) = \varphi'_1(v)$, we get, by definition of $\preceq$ (distinguishing the options $\varphi_0(v) \in V_-$ and $\varphi_0(v) \in V_+$) the claim of the lemma: $\varphi(v) \preceq \varphi'(v)$. □

4 The Algorithm

In this section we give an algorithm for constructing an optimal locally progressive valuation for a given parity game graph. This will lead to winning strategies for the game. The algorithm is split into three functions: *main()*, *valuation()*, and *subvaluation()*.

In the function *main()* a sequence of strategies for player 0 is generated and for each of these strategies a locally progressive valuation is computed by calling *valuation()*. This valuation is constructed such that the strategy of player 0 is compatible with it and that it is optimal for player 1. The first strategy for player 0 is chosen randomly. Subsequent strategies for player 0 are chosen such that they are optimal with respect to the previous valuation. The main loop terminates if the strategy chosen for player 0 is the same as in the previous iteration. Finally a strategy for player 1 is extracted from the last computed valuation.

In the functions *valuation()* and *subvaluation()*, we use the functions *reach*(G, u), *minimal_distances*(G, u), *maximal_distances*(G, u). The functions work on the graph G and perform a backward search on the graph starting in vertex u.

- The function *reach*(G, u) produces the set of all vertices from which the vertex u can be reached in G (done by a backward depth first search).
- The function *minimal_distances*(G, u) computes a vector $\delta : V_G \to \{0, \dots, |V_G| - 1\}$ where $\delta(v)$ is the length of the shortest path from v to u (done by a backward breadth first search starting from u).

main(G):

for each $v \in V_0$ $\{$Select initial strategy for player 0.$\}$
  select $\sigma(v) \in V_1$ **with** $v\, E\, \sigma(v)$
repeat
  $G_\sigma = (V_0, V_1, E', c)$, where f.a. $u, v \in V$:
    $uE'v \iff (\sigma(u) = v \wedge u \in V_0) \ \vee \ (uEv \wedge u \in V_1)$
  $\varphi = valuation(G_\sigma)$
  $\sigma' = \sigma$ $\{$Store σ under name $\sigma'\}$
  for each $v \in V_0$ $\{$Optimize σ locally according to $\varphi\}$
    if $\varphi(\sigma(v)) \prec \max_\prec\{d \in \mathcal{D} | \exists v' \in V : v\, E\, v' \wedge d = \varphi(v')\}$ **then**
      select $\sigma(v) \in V_1$
      with $\varphi(\sigma(v)) = \max_\prec\{d \in \mathcal{D} | \exists v' \in V : v\, E\, v' \wedge d = \varphi(v')\}$
until $\sigma = \sigma'$
for each $v \in V_1$
  select $\tau(v) \in V_0$
    with $\varphi(\tau(v)) = \min_\prec\{d \in \mathcal{D} | \exists v' \in V : v\, E\, v' \wedge d = \varphi(v')\}$
$W_0 = \{v \in V \mid \varphi_0(v) \in V_+\}$
$W_1 = \{v \in V \mid \varphi_0(v) \in V_-\}$
return W_0, W_1, σ, τ

Figure 1: This function computes the winning sets and a winning strategy for each player from a given game graph.

- The function *maximal_distances*(G, u) yields a vector $\delta : V_G \to \{0, \dots, |V_G| - 1\}$ where $\delta(v)$ is the length of the longest path from v to u that does not contain u as an intermediate vertex. This is done by a backward search starting from u where a new vertex is visited if all its successors are visited before. This algorithm only works if every cycle in the graph contains u.

The function *valuation(H)* produces a locally progressive valuation for the graph H that is optimal for player 1. It does this by splitting the graph into a set of subgraphs for which *subvaluation()* can compute a locally progressive valuation of this kind.

The function searches a loop and then the set of vertices R from which this loop can be reached. Computing the locally progressive valuation for the subgraph induced by R is done by *subvaluation()*. The rest of the graph (i.e., the subgraph induced by $V_H \setminus R$) is treated in the same way.

The algorithm *valuation(H)* is given in Figure 2. It works as follows:

1. φ is set 'undefined' for all v.

2. In ascending reward order those vertices w are found which belong to a loop L consisting solely of $\leq$-smaller vertices. Then, for a fixed w, the set R_w of all vertices is determined from which this loop (and hence w) is reachable (excluding vertices which have been used in sets $R_{w'}$ for previously considered w'). The valuation is updated on the set R_w.

valuation(H):

for each $v \in V_H$ **do**
    $\varphi(v) = \bot$.
for each $w \in V_H$ (ascending order with respect to $\prec$) **do**
    if $\varphi(w) = \bot$ **then**
        $L =$ *reach*$(H|_{\{v \in V_H | v \leq w\}}, w)$
        if $E_H \cap \{w\} \times L \neq \emptyset$ **then**
            $R =$ *reach*(H, w)
            $\varphi|_R =$ *subvaluation*$(H|_R, w)$
            $E_H = E_H \setminus (R \times (V_H \setminus R))$
return φ

Figure 2: This function computes for graph H a locally progressive valuation that is optimal for player 1.

The new valuation is determined as follows. By backward depth-first search from w the vertices v in R_w are scanned, and edges leading outside R_w deleted, in order to prohibit entrance to R_w by a later search (from a different w').

In *subvaluation(K, w)* the role of vertices v in R_w w.r.t. $\prec_w$ is analyzed. We shall have $\varphi_0(v) = w$ for them. One proceeds in decreasing relevance order:

- One distinguishes whether u is positive or negative. If u is positive, one computes the set $\overline{U}$ of vertices from where w can be reached while avoiding u, and for those v from which a visit to u is unavoidable, we add u to $\varphi_1(v)$, and we remove edges leading from $\overline{U} \cup \{u\}$ to $V \setminus \overline{U}$.
- If u is negative then let U be the set of vertices v, such that u can be visited on a path from v to w. We add u to $\varphi_1(v)$ for all $v \in U$, and we remove edges leading from $U \setminus \{u\}$ to $V \setminus U$.

Finally, if $w \in V+$ ($w \in V_-$), for each $v \in R_w$ the maximal (resp. minimal) distance from v to w is assigned to $\varphi_2(v)$.

The function *subvaluation(K, w)* is given in Figure 3. It computes the paths to w that have minimal reward with respect to $\prec_w$ and stores for each vertex the resulting path in φ-value. Edges that belong to paths with costs that are not minimal are removed successively.

5 Correctness

We show that the algorithm terminates for a given game graph and returns winning sets and winning strategies for both players.

Lemma 6 *Let $G = (V_0, V_1, E, c)$ be a game graph and $w \in V$ with the following properties:*

subvaluation(K, w)*:*

1. **for each** $v \in V_K$ **do**
2. $\quad\varphi_0(v) = w$
3. $\quad\varphi_1(v) = \emptyset$
4. **for each** $u \in \{v \in V_K \mid v > w\}$ (descending order with respect to $<$) **do**
5. $\quad$**if** $u \in V_+$ **then**
6. $\quad\quad\overline{U} = \mathit{reach}(K|_{V_K \setminus \{u\}}, w)$
7. $\quad\quad$**for each** $v \in V_K \setminus \overline{U}$ **do**
8. $\quad\quad\quad\varphi_1(v) = \varphi_1(v) \cup \{u\}$
9. $\quad\quad E_K = E_K \setminus ((\overline{U} \cup \{u\}) \times (V_K \setminus \overline{U}))$
10. $\quad$**else**
11. $\quad\quad U = \mathit{reach}(K|_{V_K \setminus \{w\}}, u)$
12. $\quad\quad$**for each** $v \in U$ **do**
13. $\quad\quad\quad\varphi_1(v) = \varphi_1(v) \cup \{u\}$
14. $\quad\quad E_K = E_K \setminus ((U \setminus \{u\}) \times (V_K \setminus U))$
15. **if** $w \in V_+$ **then**
16. $\quad\varphi_2 = \mathit{maximal\_distances}(K, w)$
17. **else**
18. $\quad\varphi_2 = \mathit{minimal\_distances}(K, w)$
19. **return** φ

Figure 3: This function computes a locally progressive valuation for a subgraph K with most relevant vertex w.

1. *From all vertices the vertex* w *is reachable.*
2. *There exists a loop* $L \subset V$ *with* $w = \max_< L$.
3. *There exists no loop* $L' \subset V$ *with* $\max_< L' \prec w$.

Then the function call subvaluation(G, w) *(see Figure 3) terminates and returns a locally progressive valuation* φ *that is optimal for player* 1.

Proof: For the proof of termination observe that in the function all for–loops have a fixed finite number of iterations. So it suffices to show that all function calls used in *subvaluation()* terminate. For *reach()* and *minimal_distances()* termination is obvious. For the function call *maximal_distances(K, w)* we have to show that in K all loops contain w, if $w \in V_+$. The graph K is a subgraph of G. If $w \in V_+$ then from assumption 3 of this lemma we know that for any loop L' one has $w \leq \max_< L' \in V_+$. But by line 9 in *subvaluation()* it is guaranteed that there is no loop through $\max_< L'$ in subgraph K. So for every loop L' in K we have $w \leq \max_< L'$, which means that w occurs in L'. So the function call *subvaluation(G, w)* terminates.

Now we show that the returned valuation φ is locally progressive. So we have to prove:

$$\forall x \in V_G \; \exists y \in V_G : xEy \;\wedge\; x \lhd_\varphi y.$$

In executing the function, some edges of G are removed, yielding a subgraph K. We show that φ is a locally progressive valuation for this subgraph, which means that it is also a locally progressive valuation for G.

Since $\varphi_0(v) = w$ for all $v \in V_G$, the φ_0–conditions in the definition of $\lhd_\varphi$ are satisfied. The loop in line 4 starts with $\forall v \in V_G : \varphi_1(v) = \emptyset$. For this loop the following invariant clearly holds after each iteration, for the respective u:

$$\forall x, y \in V : xEy \Longrightarrow (x < u \land \varphi_1(x) = \varphi_1(y)) \lor (x \geq u \land \varphi_1(x) = \varphi_1(y) \,\dot\cup \{x\}).$$

When the loop terminates, we have:

$$\forall x, y \in V : xEy \Longrightarrow (x < w \land \varphi_1(x) = \varphi_1(y)) \lor (x \geq w \land \varphi_1(x) = \varphi_1(y) \,\dot\cup \{x\}).$$

Because the value of $\varphi_1(w)$ is not modified, we have $\varphi_1(w) = \emptyset$. So the φ_1–conditions for $\lhd_\varphi$ are satisfied.

Finally for every $v \in V$ the value of $\varphi_2(v)$ is set to the minimal, respectively maximal distance to w. Hence we have:

$$\forall x \in V : (x = w \land \varphi_2(w) = 0) \;\lor\; \exists y \in V : xEy \land \varphi_2(x) = \varphi_2(y) + 1,$$

which implies the φ_2–conditions for $\lhd_\varphi$. So φ is a locally progressive valuation for K.

It remains to show that φ is an optimal valuation for player 1 on G. First we show that for each edge which the function removes, there is an edge with the same source vertex which has a target vertex that is assigned a $\prec$-smaller play profile by φ. Given this, it suffices to prove that φ is an optimal valuation for player 1 on the final subgraph K.

In line 6 those vertices are collected in $\bar{U}$ for which a path to w exists that avoids u. That means each vertex in $\bar{U} \setminus \{w\}$ has an E-successor in $\bar{U}$. In line 9 all edges from $\bar{U} \cup \{u\}$ to $V \setminus \bar{U}$ are removed. For such an edge with source vertex x and target vertex y let vertex z be an E-successor of x with $z \in \bar{U}$. From the invariant we know

$$\varphi_1(y) \setminus \{u\} = \varphi_1(x) \setminus \{u, x\} = \varphi_1(z) \setminus \{u\}.$$

Because $u \in \varphi_1(y)$, $u \notin \varphi_1(z)$ and $u \in V_+$ we have $\varphi_1(z) \prec_u \varphi_1(y)$. Later in this function only $<$-smaller vertices are added to φ_1-values. So finally we have $\varphi_1(z) \prec_u \varphi_1(y)$ and so $\varphi(z) \prec \varphi(y)$. Hence these removed edges do not effect the optimality of the valuation for player 1.

In line 11 those vertices are collected in U for which a path to u exists. That means each vertex in $U \setminus \{u\}$ has an E-successor in U. In line 14 all edges from $U \setminus \{u\}$ to $V_K \setminus U$ are removed. For such an edge with source vertex x and target vertex y let vertex z be an E-successor of x with $z \in U$. From the invariant we know

$$\varphi_1(y) \setminus \{u\} = \varphi_1(x) \setminus \{u, x\} = \varphi_1(z) \setminus \{u\}.$$

Because $u \notin \varphi_1(y)$, $u \in \varphi_1(z)$ and $u \in V_-$ it follows $\varphi_1(z) \prec_u \varphi_1(y)$. Again later in this function only $<$-smaller vertices are added to φ_1-values. So finally we have $\varphi_1(z) \prec_u \varphi_1(y)$ and so $\varphi(z) \prec \varphi(y)$. So these removed edges do also not effect the optimality of the valuation for player 1.

In line 19 we know from the invariant that for each vertex all E-successors have the same φ_1-value. That means the φ-values of E-successors only differ in φ_2. So the optimality property for player 1 is already satisfied for vertex w. If $w \in V_+$ the φ_2-value is the maximal distance to w. Let $x \in V \setminus \{w\}$. Then we have for every E-successor $y \in V$ of x with maximal φ_2-value: $\varphi_2(x) = \varphi_2(y)+1$, which means $x \lhd_\varphi y$, and thus the optimality property for player 1 is satisfied. The case $w \in V_-$ is analogous. □

Lemma 7 *Let $G = (V_0, V_1, E, c)$ be a game graph. Let $\sigma : V_0 \to V_1$ be a strategy for player 0. Let G_σ be the game graph G restricted to strategy σ, i.e. $G_\sigma = (V_0, V_1, E', c)$, where for all $u, v \in V$:*

$$uE'v \iff (\sigma(u) = v \land v \in V_0) \ \lor \ (uEv \land v \in V_1).$$

Then the function call valuation(G_σ) (see Figure 2) terminates and returns a locally progressive valuation. The strategy σ is compatible with this valuation and this valuation is optimal for player 1.

Proof: To prove that the function call *valuation(G_σ)* returns a valuation such that σ is compatible with it, it suffices to prove that the call returns a locally progressive valuation, because for every locally progressive valuation there is at least one strategy for each player compatible with this valuation, and in G_σ the only such strategy for player 0 is σ.

First the play profiles of all vertices are set undefined. In the main loop (line 3) a smallest vertex w with respect to $\prec$ is found among those vertices that are undefined for which the following property is satisfied: It exists a loop (line 5) containing this vertex w as its most relevant vertex (line 6). For such a vertex w the set of all vertices R from which it is reachable is computed (line 7). By calling *subvaluation()* the locally progressive valuation of the subgraph induced by the vertex set R is computed, and it is assigned to φ. Finally all edges leaving R are removed. These edges cannot violate the optimality for player 1, because for all $v \in R$: $\varphi_0(v) = w$, and any of these edges leads to a vertex v' with $\varphi(v') = \perp$, which means that the 0-component of any later assigned play profile becomes $\prec$-greater than w. So all edges between the identified subgraphs do not violate the optimality condition for player 1, and by Lemma 7 the valuation for the subgraphs is optimal for player 1, hence the valuation of the whole graph is optimal for player 1. □

Lemma 8 *Let $G = (V_0, V_1, E, c)$ be a game graph. Then the function call main(G) (see Figure 1) terminates.*

Proof: The only nontrivial function call in *main()* is *valuation(G_σ)*. By Lemma 7 we know that it terminates. So it remains to check the repeat–loop, which is the only loop in *main()* with a possible unbounded number of iterations.

From Lemma 7 we know that the valuations produced in each iteration are optimal for player 1 and that they are induced by the strategy in σ before the call *valuation(G_σ)*. Let φ be a valuation computed in an iteration and φ' be the valuation computed in the next iteration. With Lemma 5 we get

$$\forall v \in V : \varphi(v) \preceq \varphi'(v). \tag{11}$$

There are only finitely many strategies for player 0, so after a finite number of iterations the same strategy σ is chosen for player 0 again. For equal strategies the same valuations are computed. From (11) we get immediately that all valuations from the first iteration for strategy σ onwards are the same. For equal valuations, the same strategies are chosen for player 0. So after the first occurrence of σ this strategy is chosen in the next iteration again, which makes the loop terminate. □

Theorem 9 *Let $G = (V_0, V_1, E, c)$ be a game graph. Let the sets W_0, W_1 and strategies σ for player 0 and τ for player 1 be the results of the function call main(G) (see Figure 1).*

Then W_0 and W_1 form a partition of V, and σ is a winning strategy on W_0 for player 0 and τ is a winning strategy on W_1 for player 1.

Proof: By definition of W_0 and W_1 given in lines 13, 14 of Figure 1 the sets W_0, W_1 form a partition of V.

From Lemma 7 we know that the finally computed valuation is locally progressive and optimal for player 1. Because player 0 cannot improve his strategy with respect to the previous valuation (see lines 8–10 in Figure 1), it follows that the valuation is also optimal for player 0. So the finally computed valuation is an optimal valuation. From Lemma 7 it follows that σ is compatible with this valuation and in line 12 in Figure 1 the strategy τ is selected such that it is also compatible with this valuation. By Theorem 4 the strategy σ is a winning strategy for player 0 (on W_0) and τ a winning strategy for player 1 (on W_1). □

6 Time complexity

In this section we give the time complexity of the algorithm of section 4, by providing a bound for one improvement step (Lemma 12). The overall complexity is given in Proposition 13. We leave open the question whether the (exponential) bound appearing in Proposition 13 in terms of the number of strategies can be substantially improved.

Lemma 10 *Let $G = (V_0, V_1, E, c)$ be a game graph and $w \in V$. Let $m = |E|$ and $n = |V|$. Then the function call subvaluation(G, w) is executed in time $O(mn)$.*

Proof: The loop in lines 1–3 takes time $O(n)$. Lines 8 and 13 are constant time operations. So the loops in lines 7–8 and 12–13 can be done in time $O(n)$. The two depth first searches in lines 6 and 11 can be done in time $O(m)$, as well as removing some edges in lines 9 and 14. This means the body of the loop in lines 5 – 14 has time complexity $O(m)$. The loop is executed $O(n)$ times. Hence, lines 4 – 14 are done in time $O(mn)$. The backward searches in lines 16 and 18 are can be done in time $O(m)$, which means lines 15–18 take time $O(m)$. Line 19 takes time $O(n^2)$, because for each vertex φ_1 contains a set of states. Because for each vertex there is an out–edge, we have $n \leq m$. This yields the claim. □

Lemma 11 *Let $G = (V_0, V_1, E, c)$ be a game graph. Let $m = |E|$ and $n = |V|$. Then the function call valuation(G) is executed in time $O(mn)$.*

Proof: The loop in lines 1–2 takes time $O(n)$. The depth first searches in lines 5 and 7 take time $O(m)$. Removing edges in line 9 also takes time $O(m)$. The loop in lines 3–9 is executed $O(n)$ times and its body can be done in time $O(m)$, except line 8. The call of *subvaluation()* in line 8 is done with disjoint subgraphs of G. So it takes time $O(mn)$ for all these subgraphs. So the loop in lines 3–9 takes time $O(mn)$. Line 10 takes time $O(n^2)$. Because $n \leq m$, the claim follows. □

Lemma 12 *Let $G = (V_0, V_1, E, c)$ be a game graph and $w \in V$. Let $m = |E|$ and $n = |V|$. Then each iteration of the main loop in main(G) is executed in time $O(mn)$.*

Proof: In line 4, building a subgraph, takes time $O(m)$. By Lemma 11, computing *valuation()* in line 5 takes time $O(mn)$. The assignment of line 6 can be done in time $O(n)$. The loop in lines 7–9 checks each edge at most once and does at most one profile comparison for each check. A profile comparison can be done in time $O(n)$, and so lines 7–9 take time $O(mn)$. The test in line 10 takes time $O(n)$. So a whole iteration step takes time $O(mn)$. □

Proposition 13 *Let $G = (V_0, V_1, E, c)$ be a game graph and $w \in V$. Let $m = |E|$, $n = |V|$ and d_v the out-degree for vertex $v \in V_0$. Then the function call main(G) is executed in time $O(mn \prod_{v \in V_0} d_v)$.*

Proof: The loop in lines 1–2 can be done in time $O(m)$. By Lemma 12 the body of the loop in lines 3–10 takes time $O(mn)$. This loop is executed at most for once each possible strategy of player 0, which is $\prod_{v \in V_0} d_v$ times. So lines 3–10 take time $O(mn \prod_{v \in V_0} d_v)$. The loop in lines 11–12 checks each edge at most once and does at most one profile comparison for each check. A profile comparison can be done in time $O(n)$, and so lines 11–12 take time $O(mn)$. Lines 13, 14 and 15 take time $O(n)$. So the function call *main(G)* can be done in time $O(mn \prod_{v \in V_0} d_v)$. $\square$

7 Conclusion

We have presented a discrete strategy improvement algorithm for solving parity games and hence also for the modal μ-calculus model checking. The use of a strategy improvement algorithm for solving parity games has been already advocated by Puri in his PhD thesis [12]. Our contribution is a discrete version of the algorithm, whose advantage over its predecessor in the continuous framework is twofold: it allows an efficient implementation and it provides a better grasp of the combinatorial structure underlying strategy improvement in the special case of parity games.

Our work leaves open the question whether parity games are solvable in polynomial time, but we hope that an analysis of our discrete algorithm could be of help.

We have also found some evidence to confirm Puri's claims [12] that a strategy improvement algorithm is superior to other known algorithms for solving parity games. In particular, we have verified with an implementation of our discrete strategy improvement algorithm that the algorithm needs only a linear number of strategy improvement steps on families of hard examples on which algorithms with best provable performance bounds (see [9]) have exponential time running time. In fact, we are not aware of any families of examples for which more than a linear number of strategy improvement steps are needed.

References

[1] A. Browne, E. M. Clarke, S. Jha, D. E. Long, and W. Marrero. An improved algorithm for the evaluation of fixpoint expressions. *Theoretical Computer Science*, 178(1–2):237–255, May 1997.

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

[3] Anne Condon. On algorithms for simple stochastic games. In Jin-Yi Cai, editor, *Advances in Computational Complexity Theory*, volume 13 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 51–73. American Mathematical Society, 1993.

[4] A. Ehrenfeucht and J. Mycielski. Positional strategies for mean payoff games. *Int. Journal of Game Theory*, 8(2):109–113, 1979.

[5] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy (Extended abstract). In *Proceedings of 32nd Annual Symposium on Foundations of Computer Science*, pages 368–377. IEEE Computer Society Press, 1991.

[6] E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model-checking for fragments of μ-calculus. In Costas Courcoubetis, editor, *Computer Aided Verification, 5th International Conference, CAV'93*, volume 697 of *LNCS*, pages 385–396, Elounda, Greece, June/July 1993. Springer-Verlag.

[7] A. Hoffman and R. Karp. On nonterminating stochastic games. *Management Science*, 12:359–370, 1966.

[8] Marcin Jurdziński. Deciding the winner in parity games is in UP $\cap$ co-UP. *Information Processing Letters*, 68(3):119–124, November 1998.

[9] Marcin Jurdziński. Small progress measures for solving parity games. To appear in *Proceedings of 17th International Symposium on Theoretical Aspects of Computer Science, STACS 2000*, Lille, France, February 2000.

[10] Nimrod Megiddo. Towards a genuinely polynomial algorithm for linear programming. *SIAM Journal on Computing*, 12:347–353, 1983.

[11] A. W. Mostowski. Games with forbidden positions. Technical Report 78, University of Gdańsk, 1991.

[12] Anuj Puri. *Theory of Hybrid Systems and Discrete Event Systems*. PhD thesis, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, December 1995. Memorandum No. UCB/ERL M95/113.

[13] Helmut Seidl. Fast and simple nested fixpoints. *Information Processing Letters*, 59(6):303–308, September 1996.

[14] Colin Stirling. Local model checking games (extended abstract). In Insup Lee and Scott A. Smolka, editors, *CONCUR'95: Concurrency Theory, 6th International Conference*, volume 962 of *LNCS*, pages 1–11. Springer-Verlag, 1995.

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