Strategy derivation for small progress measures

Strategy Derivation for Small Progress Measures

M. Gazda and T.A.C. Willemse

Eindhoven University of Technology, The Netherlands

Abstract Small Progress Measures is one of the most efficient parity game solving algorithms. The original algorithm provides the full solution (winning regions and strategies) in $O(dm \cdot (n/\lceil d/2 \rceil)^{\lceil d/2 \rceil})$ time, and requires a re-run of the algorithm on one of the winning regions. We provide a novel operational interpretation of progress measures, and modify the algorithm so that it derives the winning strategies for both players in one pass. This reduces the upper bound on strategy derivation for SPM to $O(dm \cdot (n/\lfloor d/2 \rfloor)^{\lfloor d/2 \rfloor})$.

1 Introduction

A parity game [3,13,18] is an infinite duration game played on a directed graph by two players called *even* and *odd*. Each vertex in the graph is owned by one of the players, and labelled with a natural number, called a priority. The game is played by pushing a token along the edges in the graph; the choice where to move next is made by the owner of the vertex on which the token currently resides. The winner of the thus constructed play is determined by the parity of the minimal (or maximal, depending on the convention) priority that occurs infinitely often, and the winner of a vertex is the player who has a *strategy* to force every play originating from that vertex to be winning for her. Parity games are determined; that is, each vertex is won by some player [13]. *Solving* a game essentially means deciding which player wins which vertices in the game.

Parity games play an important role in several foundational results; for instance, they allow for an elegant simplification of the hard part of Rabin's proof of the decidability of a monadic second-order theory, and a number of decision problems of importance can be reduced to deciding the winner in parity games. For instance, the model checking problem for the modal μ-calculus is equivalent, via a polynomial-time reduction, to the problem of solving parity games [4,16]; this is of importance in computer-aided verification. Winning strategies for the players play a crucial role in supervisory control of discrete event systems, in which such strategies are instrumental in constructing a supervisor that controls a plant such that it reaches its control objectives and avoids bad situations; see *e.g.* [1] and the references therein. In model checking, winning strategies are essential in reporting witnesses and counterexamples, see [16].

A major impetus driving research in parity games is their computational status. Even though the solution problem belongs to both the complexity classes NP and coNP, no polynomial algorithm has been devised so far. Taking a slightly simplified view, today's deterministic algorithms for parity game solving can be

classified into three categories: a category of two early classical algorithms, *viz.* the *recursive algorithm* [18], solving games with d different priorities, n vertices and m edges in $O(m \cdot n^d)$ and the *small progress measures* (SPM) algorithm [10], solving games in $O(dm \cdot (n/\lfloor d/2 \rfloor)^{\lfloor d/2 \rfloor})$; a category of the fastest known algorithms, *viz.* the deterministic subexponential algorithm [11] and the *bigstep* algorithm [14]; and a category of strategy improvement algorithms [17,15,5].

For a considerable time, strategy improvement algorithms were perceived as likely candidates for solving parity games in polynomial time, but they were ultimately proven to be exponential in the worst-case [6]. In fact none of today's strategy improvement algorithms matches the bigstep algorithm or the deterministic subexponential algorithm. The latter is a modification of the classical recursive algorithm, running in $n^{O(\sqrt{n})}$, and the bigstep algorithm combines the recursive algorithm and the SPM algorithm, running in $O(m \cdot (\kappa n/d)^{\gamma(d)})$, where κ is a small constant and $\gamma(d) \approx d/3$.

Somewhat surprisingly, our knowledge of the classical algorithms is still far from complete. For instance, the recursive algorithm is regarded as one of the best algorithms in practice, which is corroborated by experiments [7]. However, until our recent work [8] where we showed the algorithm is well-behaved on several important classes of parity games, there was no satisfactory explanation why this would be the case. In a similar vein, in *ibid.* we provided tighter bounds on the worst-case running time, but so far, no tight bounds for this seemingly simple algorithm have been established. We expect that, if improvements on the upper bound on the parity game solving problem can be made, such improvements will come from improvements in, or through a better understanding of the classical algorithms; this expectation is fuelled by the fact that these classical algorithms are at the basis of the currently optimal algorithms.

Here, we focus on the second classical algorithm, namely Jurdziński's small progress measures algorithm. Using a fixpoint computation, it computes a *progress measure*, a labelling of vertices, that witnesses the existence of winning strategies. In general, no clear, intuitive interpretation of the information contained in the progress measures has been given, and the mechanics of the algorithm are still quite mysterious. This is in contrast to the self-explanatory recursive algorithm, and the strategy improvement algorithm, where, thanks to ordering of plays according to profiles, at every step, one has a clear picture of the currently known best strategy. Apart from Jurdziński's original article, some additional insight was offered in [9] (an intuitive progress measure in the setting of solitaire games), and also in Schewe's paper on *bigstep* [14] (restricted codomain and small dominions). Our *first* contribution is to provide a better understanding of these progress measures and the intermediate values in the fixpoint computation, see Section 3. By doing so, a better understanding of the algorithm itself is obtained.

Progress measures come in two flavours, *viz.* even-and odd-biased, and their computation time is bounded by either $O(dm \cdot (n/\lfloor d/2 \rfloor)^{\lfloor d/2 \rfloor})$ or $O(dm \cdot (n/\lceil d/2 \rceil)^{\lceil d/2 \rceil})$, depending on the parity of the extreme priorities. From an even-biased progress measure, one immediately obtains winning regions for *both* players, but only a

winning strategy for player even on its winning region and not for her opponent. Likewise for odd-biased progress measures. Obtaining the winning strategy for an opponent thus requires re-running the algorithm on the opponent's winning region. Note that the effort that needs to be taken to obtain a strategy in the same time bound as the winning region stems from a more general feature of parity games: a winning partition in itself does not allow one to efficiently compute a winning strategy (unless there is an efficient algorithm for solving parity games). This basic result, which we nevertheless were unable to find in the literature, is formalised in Section 4.

An essential consequence of this is that the algorithm solves a parity game in $O(dm \cdot (n/\lfloor d/2 \rfloor)^{\lfloor d/2 \rfloor})$, as one can always compute one of the two types of progress measures in the shorter time bound. Contrary to what is stated in [10], the same reasoning does not apply to computing the winning strategy for a fixed player; this still requires $O(dm \cdot (n/\lceil d/2 \rceil)^{\lceil d/2 \rceil})$ in the worst case, as also observed by Schewe in [14]. An intriguing open problem is whether it is at all possible to derive the winning strategies for both players while computing one type of measure only, as this would lower the exponent in the time bound to $\lfloor d/2 \rfloor$. Our *second* contribution is to give an affirmative answer to the above problem. We modify the generic SPM by picking a partial strategy when a vertex won by player $\Box$ is discovered, and subsequently adjust the lifting policy so that it prioritises the area which contains an $\Box$-dominion. Both steps are inspired by the interpretation of the progress measures that we discuss in Section 3. Like the original algorithm, our solution, which we present in Section 4, still works in polynomial space.

2 Preliminaries

We briefly introduce parity games in Section 2.1 and Jurdziński's Small Progress Measures algorithm in Section 2.2. For an in-depth treatment of both, we refer to [9] and the references therein.

2.1 Parity Games

A parity game is an infinite duration game, played by players *odd*, denoted by $\Box$ and *even*, denoted by $\Diamond$, on a directed, finite graph. The game is formally defined as follows.

Definition 1. *A parity game is a tuple* $(V, E, \mathcal{P}, (V_\Diamond, V_\Box))$*, where*

- V *is a finite set of vertices, partitioned in a set* $V_\Diamond$ *of vertices owned by player* $\Diamond$*, and a set of vertices* $V_\Box$ *owned by player* $\Box$*,*
- $E \subseteq V \times V$ *is a total edge relation,* i.e. *all vertices have at least one outgoing edge,*
- $\mathcal{P}{:}V \to \mathbb{N}$ *is a priority function that assigns priorities to vertices.*

Parity games are depicted as graphs; diamond-shaped nodes represent vertices owned by player $\diamond$, box-shaped nodes represent vertices owned by player $\Box$ and the priority assigned to a vertex is written inside the node, see the game depicted in Figure 1 Throughout this section, assume that $G = (V, E, \mathcal{P}, (V_\diamond, V_\Box))$ is an

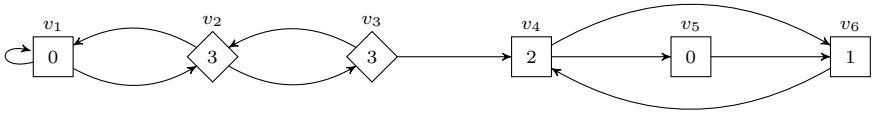

Figure 1. A simple parity game in which 4 vertices are owned by player odd, 2 vertices are owned by player even and with 4 different priorities.

arbitrary parity game. We write $v \to w$ iff $(v, w) \in E$, and $v^\bullet$ to denote the set of successors of v, *i.e.* $\{w \in V \mid v \to w\}$. For a set of vertices $W \subseteq V$, we will denote the minimal priority occurring in W with $\mathsf{min}_\mathcal{P}(W)$; by V_i we denote the set of vertices with priority i; likewise for $V_{\leq k}$. Henceforth, we assume that $\bigcirc$ denotes an arbitrary player; that is $\bigcirc \in \{\Box, \diamond\}$. We write $\bar{\bigcirc}$ for $\bigcirc$'s opponent: $\bar{\diamond} = \Box$ and $\bar{\Box} = \diamond$. For a set $A \subseteq V$, we define $G \cap A$ as the structure $(A, E \cap (A \times A), \mathcal{P}|_A, (V_\diamond \cap A, V_\Box \cap A))$; the structure $G \setminus A$ is defined as $G \cap (V \setminus A)$. The structures $G \cap A$ and $G \setminus A$ are again a parity game if their edge relations are total (in general, this is not the case).

Plays and Strategies A sequence of vertices $v_1, \ldots, v_n$ is a *path* if $v_m \to v_{m+1}$ for all $1 \leq m < n$. Infinite paths are defined in a similar manner. We write p_i to denote the i^{th} vertex in a path p.

A game starts by placing a token on some vertex $v \in V$. Players $\diamond$ and $\Box$ move the token indefinitely according to a single simple rule: if the token is on some vertex that belongs to player $\bigcirc$, that player moves the token to an adjacent vertex. An infinite sequence of vertices created this way is called a *play*. The *parity* of the *least* priority that occurs infinitely often on a play defines the *winner* of the play: player $\diamond$ wins if, and only if this priority is even. This is known as the *parity condition*.

A *strategy* for player $\bigcirc$ is a partial function $\sigma{:}V^+ \to V$ satisfying that whenever it is defined for a finite path $u_1 \cdots u_n \in V^+$, both $u_n \in V_\bigcirc$ and $\sigma(u_1 \cdots u_n) \in u_n^\bullet$. An infinite play $u_1\ u_2\ u_3 \cdots$ is *consistent* with a strategy σ if all prefixes $u_1 \cdots u_n$ of the play for which $\sigma(u_1\ \cdots u_n)$ is defined, $u_{n+1} = \sigma(u_1 \cdots u_n)$. Some strategy σ is winning for player $\bigcirc$ from vertex v iff all plays consistent with σ are won by player $\bigcirc$. Vertex v is won by player $\bigcirc$ whenever she has a winning strategy for all plays starting in vertex v. Parity games are *determined* [3], meaning that every vertex is won by one of the players; they are even *positionally determined*, meaning that if $\bigcirc$ wins a vertex then she has a winning *positional strategy*: a strategy that determines where to move the token next based solely on the vertex on which the token currently resides. Such strategies

can be represented by a function $\sigma{:}V_{\bigcirc} \to V$. *Solving* a parity game G essentially means computing the partition $(\mathsf{Win}_{\diamond}(G), \mathsf{Win}_{\Box}(G))$ of V into vertices won by player $\diamond$ and player $\Box$, respectively.

Example 1. In the parity game of Figure 1, v_1, v_2 and v_3 are won by player $\diamond$; v_4, v_5 and v_6 are won by player $\Box$. A winning positional strategy for player $\diamond$ is to play from v_2 to v_1 and from v_3 to v_2. A winning strategy for $\Box$ is to move from v_4 to v_6 and from v_5 to v_6.

Attractors and Dominions An $\bigcirc$-*attractor* into a set $U \subseteq V$ contains all those vertices from which player $\bigcirc$ can force any play to U; it is formally defined as follows.

Definition 2. *The* $\bigcirc$-attractor *into a set* $U \subseteq V$*, denoted* $\bigcirc$-$Attr(U)$*, is the least set* $A \subseteq V$ *satisfying:*

1. $U \subseteq A$
2. (a) *if* $w \in V_{\bigcirc}$ *and* $w^{\bullet} \cap A \neq \emptyset$*, then* $w \in A$
   (b) *if* $w \in V_{\overline{\bigcirc}}$ *and* $w^{\bullet} \subseteq A$*, then* $w \in A$

Observe that the complement of an attractor set of any subset of a parity game induces a parity game, *i.e.* $G \setminus \bigcirc$-$Attr(U)$ for any U and $\bigcirc$ is a well-defined parity game. Whenever we refer to an *attractor strategy* associated with $\bigcirc$-$Attr(U)$, we mean the positional strategy that player $\bigcirc$ can employ to force play to U; such a strategy can be computed in $\mathcal{O}(|V| + |E|)$ using a straightforward fixpoint iteration.

A set of vertices U is an $\bigcirc$-dominion whenever there is a strategy σ for player $\bigcirc$ such that every play starting in U and conforming to σ is winning for $\bigcirc$ and stays within U. A game is a *paradise* for player $\bigcirc$ if the entire game is an $\bigcirc$-dominion.

We shall frequently work with strategies or dominions *in the context of a certain subgame* $G' \subset G$, which do not retain their properties when moving to a larger context of G. For instance, consider a subset of vertices $W \subset V$ that induces a subgame $G \cap W$, and moreover that there is a subset $D \subseteq W$ which is a $\bigcirc$-dominion *in* $G \cap W$. Observe that, in general, D is not a $\bigcirc$-dominion within G. In such cases we always explicitly state which context is assumed.

2.2 Jurdziński's Small Progress Measures Algorithm

The SPM algorithm works by computing a *measure* associated with each vertex that characterises even (resp. odd) cycles: it is such that it decreases along the play with each "bad" odd priority encountered, and only increases upon reaching a beneficial even priority. In what follows, we recapitulate the essentials for defining and studying the SPM algorithm; our presentation is—as in the original paper by Jurdziński—from the perspective of player $\diamond$.

Let $G = (V, E, \mathcal{P}, (V_{\diamond}, V_{\Box}))$ be a parity game. Let d be a positive number and let $\alpha \in \mathbb{N}^d$ be a d-*tuple* of natural numbers. We number its components from

0 to $d-1$, *i.e.* $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_{d-1})$, and let $<$ on such d-tuples be given by the lexicographic ordering. These tuples will be used to (partially) record how often we can or must see vertices of a particular priority on all plays. We define a derived ordering $<_i$ on k-tuples and l-tuples of natural numbers as follows:

$$(\alpha_0, \alpha_1, \ldots, \alpha_k) <_i (\beta_0, \beta_1, \ldots, \beta_l) \text{ iff } (\alpha_0, \alpha_1, \ldots, \alpha_i) < (\beta_0, \beta_1, \ldots, \beta_i)$$

where, if $i > k$ or $i > l$, the tuples are suffixed with 0s. Analogously, we write $\alpha =_i \beta$ to denote that α and β are identical up-to and including position i. Intuitively, the derived ordering provides enough information to reason about the interesting bits of plays: when encountering a priority i in a play, we are only interested in how often we can or must visit vertices of a more significant (*i.e.* smaller) priority than i, whereas we no longer care about how often we have seen less significant priorities.

Now, assume from hereon that $d-1$ is the largest priority occurring in G; *i.e.*, d is one larger than the largest priority in G. For $i \in \mathbb{N}$, let $n_i = |V_i|$. Define $\mathbb{M}^{\diamond} \subseteq \mathbb{N}^d \cup \{\top\}$, containing $\top$ ($\top \notin \mathbb{N}^d$) and only those d-tuples with 0 on *even* positions and natural numbers $\leq n_i$ on *odd* positions i.

The lexicographical ordering $<$ and the family of orderings $<_i$ on d-tuples is extended to an ordering on $\mathbb{M}^{\diamond}$ by setting $\alpha < \top$ and $\top = \top$. Let $\rho{:}V \to \mathbb{M}^{\diamond}$ and suppose $w \in v^{\bullet}$. Then $\mathsf{Prog}(\rho, v, w)$ is the least $m \in \mathbb{M}^{\diamond}$, such that

- $m \geq_{\mathcal{P}(v)} \rho(w)$ if $\mathcal{P}(v)$ is even,
- $m >_{\mathcal{P}(v)} \rho(w)$, or $m = \rho(w) = \top$ if $\mathcal{P}(v)$ is odd.

Definition 3. *Function ρ is a* game parity progress measure *if for all $v \in V$:*

- *if $v \in V_{\diamond}$, then for some $w \in v^{\bullet}$, $\rho(v) \geq_{\mathcal{P}(v)} \mathsf{Prog}(\rho, v, w)$*
- *if $v \in V_{\Box}$, then for all $w \in v^{\bullet}$, $\rho(v) \geq_{\mathcal{P}(v)} \mathsf{Prog}(\rho, v, w)$*

Proposition 1 (Jurdziński [10]). *If ρ is the* least *game parity progress measure, then for all $v \in V$: $\rho(v) \neq \top$ iff $v \in W_{\diamond}$.*

The least game parity progress measure can be characterised as the least fixpoint of a monotone transformer on the complete lattice we define next. Let $\rho, \rho'{:}V \to \mathbb{M}^{\diamond}$ and define $\rho \sqsubseteq \rho'$ as $\rho(v) \leq \rho'(v)$ for all $v \in V$. We write $\rho \sqsubset \rho'$ if $\rho \sqsubseteq \rho'$ and $\rho \neq \rho'$. Then the set of all functions $V \to \mathbb{M}^{\diamond}$ with $\sqsubseteq$ is a complete lattice. The monotone transformer defined on this set is given as follows:

$$\mathsf{Lift}(\rho, v) = \begin{cases} \rho[v \mapsto \min\{\mathsf{Prog}(\rho, v, w) \mid v \to w\}] & \text{if } v \in V_{\diamond} \\ \rho[v \mapsto \max\{\mathsf{Prog}(\rho, v, w) \mid v \to w\}] & \text{if } v \in V_{\Box} \end{cases}$$

As a consequence of Tarski's fixpoint theorem, we know the least fixpoint of the above monotone transformer exists and can be computed using Knaster-Tarski's iteration scheme. This leads to the original SPM algorithm, see Algorithm 1. Upon termination of the iteration within the SPM algorithm, the computed game parity progress measure ρ is used to compute the sets $\mathsf{Win}_{\diamond}(G)$ and $\mathsf{Win}_{\Box}(G)$ of vertices won by player $\diamond$ and $\Box$, respectively.

Algorithm 1 The original Small Progress Measures Algorithm

```
1: function SPM(G)
2:     Input G = (V, E, P, (V_◇, V_□))
3:     Output Winning partition (Win_◇(G), Win_□(G))
4:     ρ ← λv ∈ V. (0, ..., 0)
5:     while ρ ⊏ Lift(ρ, v) for some v ∈ V do
6:         ρ ← Lift(ρ, v) for some v ∈ V such that ρ ⊏ Lift(ρ, v)
7:     end while
8:     return ({v ∈ V | ρ(v) ≠ ⊤}, {v ∈ V | ρ(v) = ⊤})
9: end function
```

Theorem 1 (See [10]). *Algorithm 1 solves a parity game in* $O(dm \cdot (n/\lfloor d/2 \rfloor)^{\lfloor d/2 \rfloor})$.

The above runtime complexity is obtained by considering the more optimal runtime of solving a game G, or G's 'dual', obtained by shifting all priorities by one and swapping ownership of all vertices. The runtime complexity for computing winning strategies for both players using the SPM algorithm is worse. A winning strategy $\sigma_\diamond : V_\diamond \to V$ for player $\diamond$ can be extracted from ρ by setting $\sigma_\diamond(v) = w$ for $v \in V_\diamond \cap \mathsf{Win}_\diamond(G)$ and $w \in v^\bullet$ such that $\rho(w) \leq \rho(w')$ for all $w' \in v^\bullet$. A winning strategy for player $\square$ cannot be extracted from ρ *a posteriori*, so, as also observed in [14], the runtime complexity of computing a winning strategy cannot be improved by considering the dual of a game (contrary to the claim in [10]).

Theorem 2 (See also [14]). *Algorithm 1 can compute winning strategies for both players in* $O(dm \cdot (n/\lceil d/2 \rceil)^{\lceil d/2 \rceil})$.

To facilitate the analysis of SPM, we will use the following terms and notions. The term *measure* refers to the intermediate values of ρ in the lifting process as well. Given a tuple $m \in \mathbb{M}^\diamond$, we say that the position i in m is *saturated*, if $(m)_i = |V_i|$.

A convenient abstraction of an instance of SPM being executed on a particular game is a sequence of intermediate measure values $\rho_0 \rho_1 \dots \rho_C$, where ρ_C is the current measure value (as we frequently consider partial executions of SPM, ρ_C is not necessarily the final, stable measure). Formally, we define a *lifting context* as a tuple $\langle G, ms \rangle$, where G is a parity game, and $ms = \rho_0 \rho_1 \dots \rho_C$ a sequence of all intermediate measure values.

3 An operational interpretation of progress measures

While SPM is a relatively simple algorithm in the sense that it is concise and its individual steps are elementary operations, it lacks a clear and appealing explanation of the devices used. It is therefore difficult to understand, and possibly enhance. Apart from the formal definition of progress measures, little explanation of what is hidden behind the values in tuples is offered. Notable exceptions are [12], which explains that when restricted to $\diamond$-solitaire games, one can use

the maximal degrees of 'odd stretches' (a concept we make precise below) in order to define a certain parity progress measure, and Schewe's bigstep paper [14], where it is shown that dominions of a bounded size can be detected using measures with a restricted codomain. In general, the high-level intuition is that the larger progress measure values indicate more capabilities of player $\Box$, and a value at a given position in the tuple is somehow related to the number of priorities that $\Box$ can enforce to visit.

In what follows, we present a precise and operational interpretation of measures. Our interpretation is similar in spirit to the one used in [12], but applicable to all parity games, and uses a concept known from the realm of strategy improvement algorithms – a value (or profile) of a play. Here, values are defined in terms of maximal odd-dominated stretches occurring in a prefix of a play. With this notion at hand, we can consider an optimal valuation of vertices, being the best lower bound on play values that player $\Diamond$ can enforce, or the worst upper bound that $\Box$ can achieve, *i.e.* it is an *equilibrium*. The optimal valuations range over the same codomain as progress measures, and the main result of this section states that the least game parity progress measure is equal to the optimal valuation.

Let $\mathbb{M}^{\Diamond}_{ext}$ denote all tuples in $\mathbb{N}^d \cup \{\top\}$ such that for all $m \in \mathbb{M}^{\Diamond}_{ext}$ and even positions $i \leq d$, $(m)_i = 0$; *i.e.* compared to $\mathbb{M}^{\Diamond}$, the requirement that values on odd positions i are bounded by $|V_i|$. Elements in $\mathbb{M}^{\Diamond}_{ext}$ are ordered in the same fashion as those in $\mathbb{M}^{\Diamond}$. Given a play π, a *stretch* is a subsequence $\pi_s\pi_{s+1}\ldots\pi_{s+l}$ of π. For a priority k, a *k-dominated stretch* is a stretch in which the minimal priority among all vertices in the stretch is k. The *degree* of a k-dominated stretch is the number of vertices with priority k occurring in the stretch.

Definition 4. *Let us denote all plays in the parity game by Π. An $\Diamond-$*value *(or simply value) of a play is a function $\theta_{\Diamond} : \Pi \longrightarrow \mathbb{M}^{\Diamond}_{ext}$ defined as follows:*

- *if π is winning for $\Box$, then $\theta_{\Diamond}(\pi) = \top$*
- *if π is winning for $\Diamond$, then $\theta_{\Diamond}(\pi) = m$, where $m \neq \top$, and for every odd position i, $(m)_i$ is the* degree *of the maximal i-dominated stretch that is a prefix of π*

Observe that the play value is well-defined. This is because an infinite i-dominated stretch for an odd i implies that a game is won by $\Box$, hence its value is $\top$ in such case.

Example 2. Suppose that the sequence of priorities corresponding to a certain play π is $453453213(47)^*$. Then $\theta_{\Diamond}(\pi) = (0, 1, 0, 2, 0, 0, 0, 0)$.

Successor up-to-k For $m \in \mathbb{M}^{\Diamond} \setminus \{\top\}$ and $k \in \mathbb{N}$, we will denote with $\mathsf{succ}_k(m)$ the least $m' \in \mathbb{M}^{\Diamond}$ such that $m' >_k m$.

Lemma 1. *If ρ is a game progress measure of a parity game G, then for all v there is a strategy $\sigma_{\Diamond} \in \mathbb{S}_{\Diamond}$ such that for every $\pi \in \Pi(\sigma_{\Diamond}, v)$, $\theta_{\Diamond}(\pi) \leq \rho(v)$*

Proof. We will show that player $\diamond$ has a strategy to force plays with values not exceeding $\rho(v)$. The strategy in question (denoted by σ_{min}) is the same as used by the SPM algorithm – $\diamond$ always picks the vertex minimising $\rho(v)$.

We proceed with induction on $\rho(v)$. The base case ($\rho(v) = (0, 0, \ldots, 0)$) is trivial. For the inductive step, we assume that whenever the value of $\rho(w)$ for any game progress measure of an arbitrary game G and its vertex w is lower than m, then for all plays consistent with σ_{min} and starting at w, their values do not exceed $\rho(w)$.

Take v with $\rho(v) = m$. Let $\pi = v_1 v_2 \ldots$ be a play starting at $v_1 = v$ and conforming to σ_{min}, and let $m' = \theta_\diamond(\pi)$. We will prove that $m' = m$. Let k be the smallest (most significant) position such that $(m')_k > 0$. Since $m' = \theta_\diamond(\pi)$, there exists a non-trivial k-dominated stretch in the prefix of π. Let v_n be the first vertex with priority k occurring in π. From game progress measure property, the way σ_{min} is defined, and the fact that k is the least priority seen until v_n, we know that $\rho(v_i) \geq_k \rho(v_{i+1})$ for $v_1 \leq i \leq n$, and moreover for all v_i with $\mathcal{P}(v_i) = k$ the inequality is strict. Hence we have $m = \rho(v) \geq \rho(v_n) >_k \rho(v_{n+1})$, and by applying the inductive hypothesis to v_{n+1}, we know that for all plays $\pi' \in \Pi(\sigma_{min}, v_{n+1})$, we have $\theta_\diamond(\pi') \leq \rho(v_{n+1}) < m$. Let π_{post} be the postfix of π starting with v_{n+1}. Since v_n was the first occurrence of priority k in π, and dominating the prefix, we have $(\theta_\diamond(\pi))_k = (\theta_\diamond(\pi_{post}))_k + 1$, and $(\theta_\diamond(\pi))_i = (\theta_\diamond(\pi_{post}))_i$ for $i < k$. In short, we have thus $m' = \theta_\diamond(\pi) = \mathsf{succ}_k(\theta_\diamond(\pi_{post}))$. Since $m > \theta_\diamond(\pi_{post})$, we obtain $m \geq \mathsf{succ}_k(\theta_\diamond(\pi_{post})) = m'$.

Lemma 2. *If $\overline{\rho}$ is the least game progress measure of a parity game G, then there is a strategy $\sigma_\square \in \mathbb{S}^*_\square$ such that for every $\pi \in \Pi(\sigma_\square, v)$, $\theta_\diamond(\pi) \geq \overline{\rho}(v)$*

Proof. We will prove the above statement using the definition of the least progress measure as the least fixed point of the lifting operator. Fix a game G. We proceed by induction on the number of liftings that were performed until the partial measure ρ has been reached.

Base case (no liftings performed, $\rho(w) = (0, 0, \ldots, 0)$ for all w) is trivial. For the induction step we fix a vertex v which was the last vertex to be lifted before the measure ρ was obtained, and as the inductive hypothesis, we assume that for all partial measures ρ' that were obtained in all intermediate stages of lifting before v was lifted to ρ, and all vertices w, it holds that $\varphi^*_\diamond(w) \geq \rho(w)$.

We need to show that $\square$ can force a strategy on v whose value is at least $\rho(v)$ (we only need to show it for v, as the ρ-values of all other nodes are as in the previous measure approximation ρ_{prev}, and for them IH applies). Let σ_v be defined as: if $v \in V_\square$, a one-point strategy defined only on v as the successor maximising ρ, or if $v \in V_\diamond$, $\sigma_v = \emptyset$. We consider two cases, depending on the priority of v. If $\mathcal{P}(v)$ is even, then from the definition of the lifting operator, for any vertex w in a play consistent with σ_v that appears right after v, we have $\rho(v) \leq_{\mathcal{P}(v)} \rho_{prev}(w)$. From the inductive hypothesis we know that $\square$ has a strategy σ_w that forces any play starting at w to have a value at least $\rho_{prev}(w) \geq_{\mathcal{P}(v)} \rho(v)$. If we define σ as σ_v for an empty history starting at v, and then proceeding as σ_w, any play π consistent with σ has some value

$m \geq_{\mathcal{P}(v)} \rho_{prev}(w)$ (as $\mathcal{P}(v)$ is even, $\mathcal{P}(v)$ does not influence any odd i-dominated stretch for $i \leq \mathcal{P}(v)$); therefore $m \geq_{\mathcal{P}(v)} \rho(v)$, and since $(\rho(v))_i = 0$ for $i \geq 0$, we have $m \geq \rho(v)$.

From lemmata 1 and 2 we obtain the following theorem.

Theorem 3. *If $\overline{\rho}$ is the least progress measure of a parity game G, then, for all v:*

1. *there is a strategy $\sigma_\Box \in \mathbb{S}^*_\Box$ such that for every $\pi \in \Pi(\sigma_\Box, v)$, $\theta_\diamond(\pi) \geq \overline{\rho}(v)$*
2. *there is a strategy $\sigma_\diamond \in \mathbb{S}_\diamond$ such that for every $\pi \in \Pi(\sigma_\diamond, v)$, $\theta_\diamond(\pi) \leq \overline{\rho}(v)$*

The above theorem links the progress measure values to players' capabilities to enforce beneficial plays or avoid harmful ones, where the benefit from a play is measured by a specially devised play value, as it is done in strategy improvement algorithms. This offers a more operational view on progress measure values, which, combined with a more fine-grained analysis of the mechanics of SPM allows us to extract winning strategies for both players in the next section.

A notable difference between strategy improvement algorithms and SPM is that SPM does not work with explicit strategies, and the intermediate measure values do not represent any proper valuation induced by strategies – only the final least progress measure does. Still, these intermediate values give an underapproximation of the capabilities of player $\Box$ in terms of odd-dominated stretches that she can enforce.

Note that a consequence of Theorem 3 is that the least (resp. greatest) play values that player $\Box$ (resp. $\diamond$) can enforce are equal, and coincide with $\overline{\rho}$.

4 Strategy construction for player $\Box$

Computing winning strategies is typically part of a practical solution to a complex verification or a controller synthesis problem. In such use cases, these strategies are employed to construct witnesses and counterexamples for the verification problems, or for constructing control strategies for the controller [1]. As we explained in Section 2.2, the SPM algorithm can easily be extended to construct a winning strategy for player $\diamond$. The problem of deriving a winning strategy for player $\Box$ in SPM (other than by running the algorithm on the 'dual' game, or by using a 'dual' domain $\mathbb{M}^\Box$) has, however, never been addressed. Note that the problem of computing strategies is at least as hard as solving a game. Indeed, even if we are equipped with a valid winning partition $(\mathsf{Win}_\diamond(G), \mathsf{Win}_\Box(G))$ for a game G, then deriving the strategies witnessing these partitions involves the same computational overhead as the one required to compute $(\mathsf{Win}_\diamond(G), \mathsf{Win}_\Box(G))$ in the first place.

Proposition 2. *The problem of finding the winning partition $(\mathsf{Win}_\diamond(G), \mathsf{Win}_\Box(G))$ of a given game G can be reduced in polynomial time to the problem of computing a winning strategy for player $\bigcirc$ in a given $\bigcirc$-dominion.*

Proof. We will reduce the problem of recognising whether a given set D is a dominion of a given player to the strategy derivation problem. The former problem is known to be polynomially equivalent to the winning partition problem [2].

Suppose there is an algorithm $\mathcal{A}$ that, given a dominion $D \subseteq V(G)$ of player $\bigcirc$, computes a winning strategy σ of player $\bigcirc$, closed on D. Moreover, we assume that the worst-case running time of $\mathcal{A}$ has an upper bound $T(|V|, |E|, d)$. We can construct an algorithm $\mathcal{A}'$ that decides whether D is a $\bigcirc$-dominion in $O(T(|V|, |E|, d) + (|V| + |E|) \cdot \log d)$ by simply running $\mathcal{A}$ on D and analysing the outcome.

- $\mathcal{A}$ has not returned a well-defined strategy σ within $T(|V|, |E|, d)$ steps. In this case the answer is **no**
- $\mathcal{A}$ has returned some answer σ within $T(|V|, |E|, d)$ steps. By solving the induced solitaire game in $(|V| + |E|) \cdot \log d$ time, we verify whether σ is indeed a winning strategy for $\bigcirc$ on D. Is so, return **yes**, otherwise return **no**.

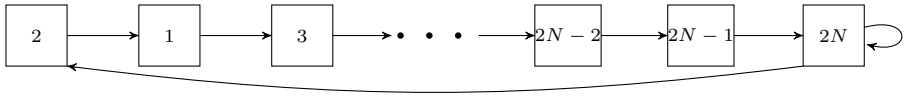

Figure 2. A parity game won by player $\square$ for which the SPM using $\mathbb{M}^{\diamond}$ is significantly faster than using $\mathbb{M}^{\square}$ to compute the non-top stable measure. For $\mathbb{M}^{\diamond}$, the first $\top$ value will be reached after the first full pass of the cycle containing priority 1 ($O(N^2)$ using a fair lifting strategy), and it will be propagated immediately to the rest of the nodes. Computation using $\mathbb{M}^{\square}$ will take an exponential time to lift the node with priority $2N$.

Deriving a strategy for both players by using the SPM to compute $\mathbb{M}^{\diamond}$ measures and $\mathbb{M}^{\square}$ measures consecutively, or even simultaneously, affects, as we already discussed in Section 2.2, SPM's runtime. This is nicely illustrated by the family of games depicted in Figure 2, for which lifting to top using an even-biased measure is exponentially faster than arriving at a stable "non-top" odd-biased measure. Being able to compute $\square$ strategies without resorting to the aforementioned methods would allow us to potentially significantly improve efficiency on such instances. It may be clear, though, that extracting a winning strategy from the small progress measures algorithm for vertices with measure $\top$ will require modifying the algorithm (storing additional data, augmenting the lifting process). For instance, simply following the vertex that caused the update to top, fails, as the example below shows.

Example 3. Reconsider the game depicted in Figure 1. Recall that in this game, vertices v_4, v_5 and v_6 are won by player $\square$, and in all possible lifting schemes, the first vertex whose measure becomes $\top$ is v_6. After that, a possible sequence

of liftings is that first $\rho(v_5)$ is set to $\top$ (due to v_6), followed by $\rho(v_4) = \top$ (due to v_5). In case we set the strategy based on the vertex that caused the given vertex to be lifted to top, we obtain $\sigma(v_4) = v_5$, which is clearly not winning for player $\Box$.

The key problem is that for vertices won by player $\Box$, from some point onwards, the lifting process discards significant information. This is best seen in case of lifting to $\top$ – a partial characterisation of reachable odd priorities contained in a tuple (see also our previous section) is ultimately replaced with a mere indication that player $\Box$ can win.

4.1 Key observation

At this point we shall give an intuitive explanation of the main insight that enables us to define part of player $\Box$ strategy in the course of lifting, once a top value is reached. In section 5 the observations made here will be formalised and proved, leading to Theorem 4, which forms the basis of our algorithm.

Consider a game G on which a standard SPM algorithm with an arbitrary lifting policy has been applied. Suppose that at some point a vertex v is the first one to be lifted to $\top$, and after lifting of v the algorithm is suspended, resulting is some temporary measure ρ. Let k be the priority of v.

We will start with two straightforward observations. The first one is that k must be an odd number; this is because a vertex with an even priority obtains, after lifting, a ρ-value equal to the ρ-value of one of its successors, and therefore it cannot be the first vertex lifted to $\top$. Another immediate conclusion is that at least one (or all, if $v \in V_\Diamond$) successor(s) of v has (have) a ρ-value saturated up to the k-th position, *i.e.* it is of the form $m = (0, |V_1|, 0, |V_3|, \ldots, 0, |V_k|, * * *)$; were it not the case, then a non-top value m' such that $m' >_k m$ would exist, which would be inconsistent with the definition of Prog.

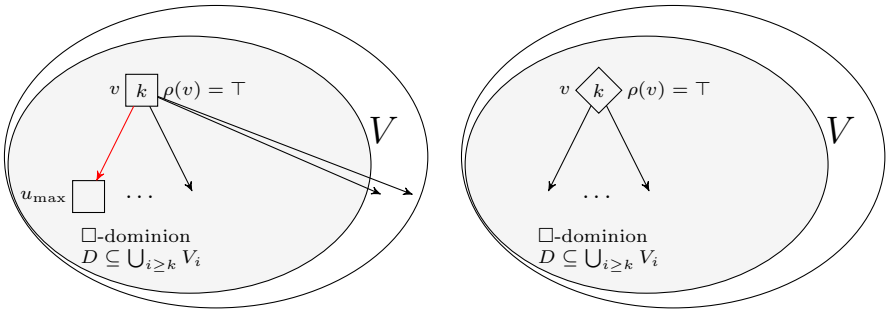

Figure 3. Snapshot of the SPM algorithm after the first vertex v is lifted to top.

There are two more complex properties, which we can utilise to modify the SPM algorithm and compute the winning strategy for player $\Box$ (see Figure 3).

1. Vertex v belongs to an $\Box$-dominion D within G such that the minimal priority in D is k.
2. If $v \in V_\Box$, then picking the successor u_{max} of v with the maximal current ρ-value among $v^\bullet$ is a part of a (memoryless) winning strategy for $\Box$ that stays within such a dominion D as described above.

The intuition concerning the above facts is as follows. Vertices with a measure value m saturated up to but possibly excluding a certain position i have a neat interpretation of the measure value at position i:

Player $\Box$ can force the following outcome of a play:

1. *priority i appears m_i times without any lower priority in between*
2. *it will reach a $\top$-labelled vertex via priorities not more significant than i*
3. *it enters a cycle with an odd dominating priority larger (less significant) than i.*

Therefore, in the situation as described above, $\Box$ can force a play starting at v to first go in one step to the successor u_{max} of v with a measure of the form $(0, |V_1|, 0, |V_3|, \ldots, 0, |V_k|, * * *)$, and then to play further and either force a less significant odd-dominated cycle (cases 2 and 3, since v is the only $\top$-labelled vertex), or to visit vertices with priority k $|V_k|$ times without any lower priority in between. But in the latter case, since v has priority k, we have in fact $|V_k|+1$ vertices with priority k not "cancelled" by a lower priority. Hence player $\Box$ has forced an odd-dominated cycle with the lowest (most significant) priority k. Note that this does not imply we can simply construct a winning strategy for $\Box$ by always picking a successor with the maximal measure to further vertices that can be visited along the play; such a method may lead to an erroneous strategy, as illustrated by Figure 4.

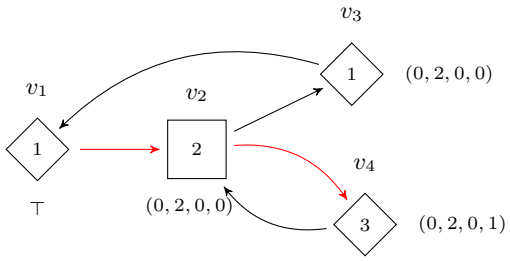

Figure 4. A simple parity game, won entirely by player $\Box$, and demonstrating that a strategy defined by a greedy choice of vertex with the maximal tuple does not work. After lifting the vertices in order v_1, v_3, v_2, v_4, v_1, we obtain the measure values as above. Player $\Box$ would then choose v_2, which leads to a losing play, whereas the choice of the other successor (v_1) yields a winning play for $\Box$.

5 A bounded $\Box$ dominion

5.1 Lifting History Graph

We first introduce an auxiliary notion of a Lifting History Graph. Its nodes (states) contain all snapshots of ρ-values that appeared at every parity game vertex in the course of the lifting, whereas the edges explain the causal dependency between ρ-values at a given vertex and its successors. In other words, the graph contains the entire history of lifting up to a certain point, and along its edges we can "go back" in the history of updates.

Definition 5. *(Lifting History Graph) Suppose we are in the context of some partial execution of the SPM algorithm, in which t liftings have been performed on a certain parity game $G = (V, E, \mathcal{P}, (V_\Diamond, V_\Box))$, starting with $\rho_0 = \lambda w \in V.\ (0, \ldots, 0)$ and yielding after each i-th lifting a temporary measure ρ_i. We define the corresponding* Lifting History Graph *$LH = (V_{LH}, E_{LH})$. The set of nodes $V_{LH} \subseteq V \times \mathbb{M}^\Diamond$ contains all pairs (v, m) such that at some stage of lifting v had a value m (i.e. there is i such that $\rho_i(v) = m$), and we define the edge relation as:*

$$\begin{aligned}(v,m)^\bullet \triangleq \{(w,m') \mid\ & w \in v^\bullet \text{ (in } G) \wedge \exists i \leq t : m = \rho_i(v) > \rho_{i-1}(v) \\ & \text{and either } v \neq w \wedge \rho_{i-1}(w) = \rho_i(w) = m' \\ & \text{or } v = w \wedge \rho_{i-1}(v) = m'\}\end{aligned}$$

that is, the successors of (v, m) in LH are those pairs (w, m') such that $w \in v^\bullet$ and when v was lifted to m, $\rho(w)$ had value m'. In other words, $(v, m)^\bullet$ constitutes a "snapshot" of ρ-values of v's successors just before v was lifted to m.

The following technical proposition summarises how the $\rho-$values in the Lifting History Graph change as we move one step back in the history of depenendencies.

Proposition 3. *Let $LH = (V_{LH}, E_{LH})$ be a lifting history graph, and $(w, m) \in V_{LH}$ a position in LH such that $(w, m)^\bullet \neq \emptyset$. Let us denote $m^{\min}$ and $m^{\max}$ respectively the minimal and maximal value of the set $\{m' \mid (u, m') \in (w, m)^\bullet\}$.*

1. *if $m \neq \top$, then*

$$\text{for all } i > \mathcal{P}(w),\ (m)_i = 0 \quad \textit{(PLH.0)}$$

and one of the following holds:

$w \in V_\Diamond$	$\mathcal{P}(w)$ *even*	$m^{\min} =_{\mathcal{P}(w)} m$	*(PLH.11)*
$w \in V_\Diamond$	$\mathcal{P}(w)$ *odd*	$m^{\min} =_{i-1} m \ \wedge\ (m^{\min})_i = (m)_i - 1$ $\wedge$ *for all* $j \in \{i+1, \ldots, \mathcal{P}(w)\}$ $(m^{\min})_j = \lvert V_j \rvert$ *where* $i = \max\{l \mid (m)_l > 0\}$	*(PLH.12)*
$w \in V_\Box$	$\mathcal{P}(w)$ *even*	$m^{\max} =_{\mathcal{P}(w)} m$	*(PLH.13)*
$w \in V_\Diamond$	$\mathcal{P}(w)$ *odd*	$m^{\max} =_{i-1} m \ \wedge\ (m^{\max})_i = (m)_i - 1$ $\wedge$ *for all* $j \in \{i+1, \ldots, \mathcal{P}(w)\}$ $(m^{\max})_j = \lvert V_j \rvert$ *where* $i = \max\{l \mid (m)_l > 0\}$	*(PLH.14)*

2. *if* $m = \top$*, then one of the following holds:*

$w \in V_\Diamond$	$\mathcal{P}(w)$ *even*	$m^{\min} = \top$	*(PLH.21)*
$w \in V_\Diamond$	$\mathcal{P}(w)$ *odd*	$m^{\min} = \top$ *or for all* $i \leq \mathcal{P}(w)$ $(m^{\min})_i = \lvert V_i \rvert$	*(PLH.22)*
$w \in V_\Box$	$\mathcal{P}(w)$ *even*	$m^{\max} = \top$	*(PLH.23)*
$w \in V_\Box$	$\mathcal{P}(w)$ *odd*	$m^{\max} = \top$ *or for all* $i \leq \mathcal{P}(w)$ $(m^{\max})_i = \lvert V_i \rvert$	*(PLH.24)*

Proof. Directly from the definitions of Prog and Lift. Observe that the value i in two subcases of the first part is well-defined, since $(w, m)^\bullet \neq \emptyset$.

Another important property of the Lifting History Graph is that the measure strictly decreases when a LH state with the same vertex is re-encountered along the path in LH.

Proposition 4. *If there is a non-trivial path in LH (i.e. containing at least one edge) from* (w, m) *to* (w, m')*, then* $m > m'$*.*

Proof. The property is easy to observe, since, intuitively, moving to a successor of (w, m) entails moving to a time spot just before w was lifted to m; when w is encountered again, the corresponding snapshot (w, m') comes from some earlier moment, and from the monotonicity of lifting we have $m > m'$ (for a more formal proof, see appendix).

We can use the lifting history graph to define a strategy of player $\Box$ that witnesses some useful capabilities of player $\Box$: being able to force a certain number of vertices with priority k to appear during the play with no lower (more significant) priority in between, or to force a winning play within a set of priorities bounded by k. We call this strategy a lifting history-based (LH-based) strategy σ^v_{LH}. Note that the strategy is not memoryless, and it is of theoretical importance only: its existence serves as a proof of certain properties from which we can in turn derive correctness of our algorithm.

LH-based strategy

Convention Thoughout the entire section about LH-based strategy, we assume a parity game $G = (V, E, \mathcal{P}, (V_\Diamond, V_\Box))$ on which some sequence of liftings has been applied, yielding a temporary, not necessarily stable, measure ρ. Let k be an *odd* number and let v_0 be a vertex such that the ρ-value of v_0 is saturated on all positions smaller than k, and at position k equal to $kval$. That is, more formally, for all odd $i < k$, $(\rho(v_0))_i = |V_i|$, and $(\rho(v_0))_k = kval$. We also assume a lifting history graph $LH = (V_{LH}, E_{LH})$ associated with the aforementioned sequence of liftings performed on G.

In such a context, we define the *lifting history-based strategy* $\sigma^{v_0}_{LH}$: a memory-wise partial strategy of player $\Box$ that guarantees the following objective: for all plays π starting at v_0 and conforming to $\sigma^{v_0}_{LH}$, either of the three holds:

1. π is finite and contains $kval$ occurrences of vertices with priority k, or

2. π is finite and it terminates in a vertex v such that $\rho(v) = \top$
3. π is infinite, winning for $\Box$, and visits only vertices with priorities larger or equal to k

For ease of presentation, we will present the definition of $\sigma_{LH}^{v_0}$ using an on-line construction procedure (i.e. an algorithm selecting the desired strategy on-the-fly as the play progresses). The procedure utilises a lifting history graph, on which it performs moves in parallel to those in the play. Intuitively we move backwards along the history of updates (liftings) of the corresponding nodes. The measures are thus successively decreased[1], until a useful (odd) cycle is encountered. We also keep track of the sequence of states in LH visited so far. If the current node in the game is $w \in V_\Box$ and the corresponding current state in the lifting history graph is (w, m), the strategy always picks the successor that had the maximal measure value when the current node w was lifted to m. Moreover, whenever an odd-dominated cycle is encountered, we remove the entire corresponding suffix from the history and revert to the last state in the lifting history graph that contained w.

We proceed with a more formal description of the on-line strategy construction procedure OddResponse, which starts a play at the inital vertex v_0 and, depending on the ownership of the current node, either receives a choice of successor of player $\Diamond$, or generates such a choice for player $\Box$. The procedure maintains the following current state information:

- (u, m): current state in LH, u is the current vertex and m one of its measure values from the lifting history. Initially $(u, m) = (v_0, \rho(u))$, where $\rho(u)$ is of the form $(0, |V_1|, 0, , \ldots, |V_{k-2}|, 0, kval, * * *)$, $* * *$ denoting some arbitrary values.
- $\lambda = \lambda_1 \ldots \lambda_n \in V^*$: history of the play (in the parity game G) so far, excluding the current vertex, initialised to an empty sequence ϵ
- $vis = vis_1 \ldots vis_{vislen} \in V_{LH}^*$: a sequence of states in LH already visited, initialised to an empty sequence ϵ

OddResponse proceeds as follows:

1. If $m = \top$, or there are $kval$ different nodes with priority k in vis, terminate.
2. If an odd-dominated cycle has been encountered, we prune vis accordingly. That is, if $(u, m') = vis_j$ for some $j < vislen$ (we have already visited u and at that point it had a measure m'), and moreover the corresponding induced cycle $u = \lambda_i \ldots \lambda_n.u$ in G is odd-dominated, then we remove the suffix containing the cycle from vis, i.e. $vis := vis_1 \ldots vis_{j-1}$. Moreover, we replace the current measure value with the previously encountered one, i.e. $(u, m) := (u, m')$.
3. we update the history: $\lambda := \lambda.u$ and $vis := vis.(u, m)$

[1] Strictly speaking, the measure values do not necessarily decrease with every single *step* in the LH graph, but re-visiting a vertex in LH graph entails a decrease in measure - see Proposition 4.

4. If $u \in V_\Diamond$, then we receive an input from player $\Diamond$ who picks the next state x from the successors of u. We set the current LH state $(u, m) := (x, m_x)$ such that $(x, m_x) \in (u, m)^\bullet$ in LH.
5. Otherwise, if $u \in V_\Box$, then we define a choice for player $\Box$: $\sigma^v_{LH}(\lambda) := x : (x, m_x) \in (u, m)^\bullet$ and m_x is maximal within $(u, m)^\bullet$ in LH. We set $(u, m) := (x, m_x)$.

The following technical lemma states an invariant that holds before every new state is processed by OddResponse. The intuition is as follows. Decrease in the tuple at position k can happen either due to an occurrence of priority k, which is good for the objective of the procedure, or due to the phenomenon of "carrying" – successor w with a lower priority than k had saturated (maximal) values on all positions from $\mathcal{P}(w)$ till k. Decrease at a given position due to carrying comes at the expense of saturated positions of less significant priorities, hence a consecutive decrease in measure must finally be compensated by visiting stretches of odd priorities greater or equal k that are not cancelled out by more important even priorities. This will finally lead to an odd cycle, or visiting *kval* nodes with priority k.

Let us denote with $stretch(i)$ the number of distinct vertices of priority i occurring in the suffix of the sequence *vis* without a lower priority in between (counting also u if $\mathcal{P}(u) = i$). The intuition behind the following lemma is that during the execution of OddResponse, the current tuple m can be split into three parts:

1. positions $i < k$ (more significant than k): saturated, $(m)_i = |V_i|$
2. positions $k \leq i \leq L$: the sum $stretch(i) + (m)_i$ has a fixed lower bound (depending on *kval* or $|V_i|$)
3. positions $i > L$: irrelevant

In addition, the lower bound for $stretch(L) + (m)_L$ is greater by 1 than on the other positions, which guarantees that always $(m)_L > 0$.

Lemma 3. *If $m \neq \top$ and $stretch(k) < kval$ (i.e. the termination condition is not satisfied), then there is a position $L \geq k$ such that for all odd $i \leq L$:*

position in m *(i ranges over odd numbers)*	***invariant***	
$i < k$	$(m)_i = \|V_i\|$ *and* $stretch(i) = 0$	*(C1)*
$i = k$	*if* $L = k$, *then* $stretch(k) + (m)_k \geq kval$	*(C21)*
	if $L > k$, *then* $stretch(k) + (m)_k \geq kval - 1$	*(C22)*
$k < i < L$	$stretch(i) + (m)_i \geq \|V_i\|$	*(C3)*
$i = L$ *and* $L > k$	$stretch(L) + (m)_L \geq \|V_L\| + 1$	*(C4)*

Moreover, the inequalities at position L can be further strengthened if L coincides with the priority of the current state:

– *if* $\mathcal{P}(u) = L = k$, *then* $stretch(k) + (m)_k \geq kval + 1$ *(C5)*

– *if* $\mathcal{P}(u) = L > k$, *then* $stretch(L) + (m)_L \geq |V_L| + 2$ *(C6)*

The above invariant implies in particular that $(m)_L > 0$, *and* $\mathcal{P}(u) \geq k$.

Proof. See the appendix.

The first corollary of the above lemma, already explained in the proof, is that during the execution of OddResponse only vertices with priorities at least equal to k are visited. The other solves one potential problem with OddResponse: if it happened that at some point $m = (0, \ldots, 0)$, we would have $(u, m)^\bullet = \emptyset$ and the choice of succesor in lines 4 and 5 would not be well-defined. But if the termination condition is not satisfied, then $(m)_L > 0$, hence $(u, m)^\bullet \neq \emptyset$.

Corollary 1. *If during the execution of* OddResponse *the current state is* (u, m) *and the termination condition is not satisfied, then* $\mathcal{P}(u) \geq k$ *and moreover the choice of successors in* OddResponse *is well-defined.*

Lemma 4. *Provided that the procedure* OddResponse *leads to an infinite play* π, *the minimal priority occurring infinitely often in* π *is odd.*

Proof. Assume towards contradiction that there is an execution of OddResponse that leads to an infinite play π such that the minimal priority occurring infinitely often on π is an even number e. Take the suffix π' of π such that e is also the lowest value occurring in π' (so we exclude transient more important priorities). Consider the path in the lifting history graph induced by π, in particular its part once π' has been entered. Take a node u_e with priority e that has been encountered infinitely many times (there must be one, since there are finitely many nodes). As u_e was not part of any odd cycle in the suffix π', the correponding occurrences of u_e in *vis* were not removed in the pruning step and hence the measures m' with which u_e occurs in *vis* form a strictly decreasing sequence (by Proposition 4). But since a sequence of decreasing measures must be finite, it means that u_e occurred only finitely many times – a contradiction.

Corollary 1 and Lemma 4 together yield correctness of the OddResponse procedure.

Proposition 5. *Assume a parity game* $G = (V, E, \mathcal{P}, (V_\Diamond, V_\Box))$ *on which a sequence of liftings has been applied, resulting in a temporary measure* ρ. *Let* k *be an* odd *number and let* v_0 *be a vertex such that for all odd* $i < k$, $(\rho(v_0))_i = |V_i|$, *and* $(\rho(v_0))_k = kval$.

There exists a strategy $\sigma^{v_0}_{LH}$ *of player* $\Box$ *that guarantees the following objective: for all plays* π *starting at* v_0 *and conforming to* $\sigma^{v_0}_{LH}$, *either of the three holds:*

1. π *has a finite prefix that is a* k*-dominated stretch of degree* $kval$ *(*$kval$ *occurrences of vertices with priority* k*)*
2. π *has a finite prefix that contains only vertices in* $V_{\geq k}$, *and terminates in a vertex* v *such that* $\rho(v) = \top$

3. *π is infinite, winning for $\Box$, and visits only vertices with priorities larger or equal to k*

It is not difficult to observe that if there is a strategy which forces an objective consisting of a disjunction of a winning condition for one of the players, and a reachability objective, then there is a memoryless strategy that guarantees the same objective (one can formally prove this, for instance, using a straightforward conversion to a winning condition in a parity game).

The second observation is that while executing OddResponse, whenever the play returns to v_0, a k-dominated cycle is formed, and the history is reset to the initial one. OddResponse then picks the same successor as in the beginning (it can be any of the neighbours maximising ρ). Because of this, we can be more precise as to the choice made by the abovementioned memoryless strategy on v_0 – any choice of a maximal neighbour works.

We summarise our considerations in the following lemma.

Lemma 5. *Assume a parity game $G = (V, E, \mathcal{P}, (V_\Diamond, V_\Box))$ on which a sequence of liftings has been applied, resulting in a temporary measure ρ such that $\rho(v) = \top$ for some vertex $v \in V$.*

There exists a memoryless strategy σ of player $\Box$ that guarantees the following objective: for all plays π starting at v and conforming to σ, either of the two holds:

1. *π visits a vertex $v_\top$ such that $\rho(v_\top) = \top$; moreover, before visiting $v_\top$, only priorities larger or equal to k are encountered*
2. *π is winning for $\Box$, and visits only vertices with priorities larger or equal to k*

In addition, if $v \in V_\Box$, then for every successor u of v with a maximal measure among $v^\bullet$ the local strategy $\sigma_u(v) = u$ can be extended to strategy σ_u with all the above properties.

5.2 Existence of the bounded $\Box$ dominion and a partial strategy assignment

We are now in a position

Corollary 2. *Assume a parity game $G = (V, E, \mathcal{P}, (V_\Diamond, V_\Box))$ on which a sequence of liftings has been applied, resulting in a temporary measure ρ such that there is exactly one vertex v with $\rho(v) = \top$. Let $k = \mathcal{P}(v)$.*

- *if $v \in V_\Box$, then for every successor u of v with a maximal measure among $v^\bullet$ there is an $\Box$-dominion D_u such that for all $w \in D_u$, $\mathcal{P}(w) \geq k$. Moreover, there is an $\Box$ strategy σ winning for $\Box$, closed on D_u, and defined on v as $\sigma(v) = u$*
- *if $v \in V_\Diamond$, then there is an $\Box$-dominion D such that $v \in D$ and for all $w \in D$, $\mathcal{P}(w) \geq k$. Note that in this case it must hold that $v^\bullet \subseteq D$.*

Proof. Observe that the abovementioned maximal successor u (or all successors, in case of $v \in V_\Diamond$), has a measure saturated on all positions up to *and including* k, and hence meets the condition of Proposition 5. We are thus able to construct a lifting history-based strategy σ^u_{LH} that either yields an infinite play within priorities larger or equal k, or will visit exactly $|V_k|$ nodes with priority k, or a top-labelled vertex. In the last two cases, v must be visited as well. Hence the desired strategy for player $\Box$ picks the successor u maximising the measure on $v^\bullet$, and then follows σ^u_{LH}; if v is re-visited, the same choices are repeatedly made.

Let D be the set of all vertices that can occur in plays conforming to σ^u_{LH} (or equivalently vertices that can be visited while executing OddResponse). They constitute a dominion of $\Box$ on which no vertex has a priority exceeding k. From memoryless determinacy of parity games, we know that there is a memoryless strategy on D winning for $\Box$, and hence a memoryless winning strategy visiting vertices that have priority values of at least k. Moreover, we know precisely the kind of choice made by this memoryless strategy on v – picking the successor with maximal measure.

The above observations are important from an algorithmic perspective, because they allow us to set the strategy of player $\Box$ on the first node v lifted to top while executing the SPM. In fact at this stage we may be able to set an $\Box$ strategy for even more nodes, following a reasoning similar to that in Zielonka's recursive algorithm – by using a strategy with which $\Box$ can "attract" the play from other nodes to v. However, to retain soundness, we use a special guarded attractor $\Box\text{-}Attr^{\geq k}(\{v\})$, which can pass only through nodes of priority not more significant than k.

The definition of the guarded attractor given below may be parameterised with a subset of vertices $W \subseteq V$, if we wish to consider only part of the game (in the remainder of the paper, we always use as W a set of vertices inducing a well-defined subgame $G \cap W$).

If we assume $U \subseteq W \cap V_{\geq k}$, then $\Box\text{-}Attr^{\geq k}_W(U)$ is the least set A satisfying:

1. $U \subseteq A \subseteq W \cap V_{\geq k}$
2. (a) if $u \in V_\Box$ and $u^\bullet \cap A \neq \emptyset$, then $u \in A$
   (b) if $u \in V_\Diamond$ and $u^\bullet \cap W \subseteq A$, then $u \in A$

Let σ_1 and σ_2 be two strategies of the same player $\bigcirc$. By $\sigma_1 \triangleright \sigma_2$ we will denote a strategy of player $\bigcirc$ defined on $\mathbf{dom}(\sigma_1) \cup \mathbf{dom}(\sigma_2)$ as $\sigma_1(w)$ for all $w \in \mathbf{dom}(\sigma_1)$, and as $\sigma_2(w)$ for all $w \in \mathbf{dom}(\sigma_2) \setminus \mathbf{dom}(\sigma_1)$.

Lemma 6. *Let $D \subseteq G$ be any dominion of $\Box$ in G and k an odd number such that all vertices in D have priority at least k, $v \in D$ such that $\mathcal{P}(v) = k$ and σ_D be a winning strategy for $\Box$ on D and closed on D. Let $\sigma^{\geq k}_{Attr}$ be a strategy defined on all vertices in the attractor $\Box\text{-}Attr^{\geq k}(\{v\}) \setminus \{v\}$ as the strategy attracting towards v. Then $\sigma^{\geq k}_{Attr} \triangleright \sigma_D$ defined on $D \cup \Box\text{-}Attr^{\geq k}(\{v\})$ is winning for $\Box$ and only visits priorities greater than or equal to k.*

Proof. Consider an arbitrary infinite play π, conforming to $\sigma^{\geq k}_{Attr} \triangleright \sigma_D$. If π visits $\Box\text{-}Attr^{\geq k}(\{v\})$ infinitely often, then from the construction of $\sigma^{\geq k}_{Attr} \triangleright \sigma_D$, it will visit v infinitely often, and from the assumption about D the lowest priority in π is k, hence π is winning for $\Box$. Otherwise, π has a suffix that stays within D, on which it conforms to σ_D, and therefore is winning for $\Box$ as well.

Finally, as an immediate consequence of Corollary 2 and Lemma 6, we obtain the main result of this section. The theorem below forms the basis of our algorithm; it describes the relevant information about an $\Box$-dominion that can be extracted once the first vertex in the game is lifted to top.

Theorem 4. *Let G be a parity game on which an arbitrary lifting sequence is applied, such that at some point a vertex v with $\mathcal{P}(v) = k$ is the first vertex whose measure value becomes top. Let ρ be the temporary measure at that point. The following holds:*

- *if $v \in V_\Box$, then for every successor u of v with a maximal measure among $v^\bullet$ there is an $\Box$-dominion D_u containing $\Box\text{-}Attr^{\geq k}(\{v\})$ such that for all $w \in D_u$, $\mathcal{P}(w) \geq k$. Moreover, there is an $\Box$ strategy σ winning for $\Box$, closed on D_u, defined on v as $\sigma(v) = u$, and on $\Box\text{-}Attr^{\geq k}(\{v\}) \setminus \{v\}$ as the strategy attracting towards v*
- *if $v \in V_\Diamond$, then there is an $\Box$-dominion D containing $\Box\text{-}Attr^{\geq k}(\{v\})$ such that for all $w \in D$, $\mathcal{P}(w) \geq k$. Moreover, there is an $\Box$ strategy σ winning for $\Box$, closed on D, and defined on $\Box\text{-}Attr^{\geq k}(\{v\}) \setminus \{v\}$ as the strategy attracting towards v. Note that in this case it must hold that $v^\bullet \subseteq D$.*

6 The extended SPM algorithm

Theorem 4 captures the core idea of our algorithm. It provides us with the means to locally resolve (*i.e.* define a local strategy for) at least one vertex in $\mathsf{Win}_\Box(G)$, once a top value is found while lifting. Moreover, it indicates in which part of the game the remainder of the $\Box$-dominion resides, implying that one can temporarily restrict the lifting to that area until the dominion is fully resolved. There is still a non-trivial task ahead: how to proceed such that the composition of all local strategy assignments will be globally valid. We will give a description of our solution (Algorithm 2), and informally argue the correctness of our approach. The formal correctness proof can be found in section 6.1.

The algorithm proceeds as follows. First, a standard SPM is run until the first vertex reaches top [l. 12–14 in Alg. 2]. Whenever v is the first vertex lifted to top, then the issue of the winning strategy for v can be resolved immediately [l. 18], as well as for vertices in the 'at-least-k' attractor of v (if there are any). We will denote this set of 'resolved' vertices with RES. Moreover, we can restrict our search for the remainder of the $\Box$-dominion D only to vertices with priorities less significant than k, in fact only those from which player $\Diamond$ cannot attract a play to visit a priority more significant than k. Hence we can remove from

Algorithm 2 Modified SPM with strategy derivation for player Odd

```
1: function SOLVE(G)
2:     Input G = (V, E, P, (V_◇, V_□))
3:     Output Winning partition and strategies ((Win_◇(G), σ'), (Win_□(G), σ))
4:     initialise σ to an empty assignment
5:     ρ ← λw ∈ V. (0, ..., 0)
6:     SPM-WITHIN(V)
7:     compute strategy σ' for player Even by picking min. successor w.r.t. ρ
8:     return (({v ∈ V | ρ(v) ≠ ⊤}, σ'), ({v ∈ V | ρ(v) = ⊤, σ))
9:
10:    procedure SPM-WITHIN(W)
11:        while (W ≠ ∅) do
12:            while ρ ⊏ Lift(ρ, w)for some w ∈ W and for all w ∈ W:ρ(w) ≠ ⊤ do
13:                ρ ← Lift(ρ, w) for w ∈ W such that ρ ⊏ Lift(ρ, w)
14:            end while
15:            if for all w ∈ W: ρ(w) ≠ ⊤ return ρ end if
16:            v ← the only vertex in W such that ρ(v) = ⊤
17:            k ← P(v)
18:            σ(v) ← u where u ∈ v• ∩ W for which ρ(u') ≤_k ρ(u) for all u' ∈ v• ∩ W
19:            RES ← □-Attr_W^{≥k}({v})
20:            for all w ∈ RES \ {v} do
21:                ρ(w) ← ⊤
22:                if w ∈ V_□ then assign σ(w) the strategy attracting to v end if
23:            end for
24:            DOM ← RES
25:            IRR ← ◇-Attr_W({w ∈ W | P(w) < k})
26:            REM ← W \ (RES ∪ IRR)
27:            SPM-WITHIN(REM)
28:            DOM ← DOM ∪ {w ∈ REM | ρ(w) = ⊤}
29:            A ← □-Attr_W(DOM)
30:            for all w ∈ A \ DOM do
31:                ρ(w) ← ⊤
32:                if w ∈ V_□ then assign σ(w) to be the strategy attracting to DOM
33:                end if
34:            end for
35:            W ← W \ A
36:        end while
37:    end procedure
38: end function
```

the current computation context the set $\mathsf{IRR} = \Diamond\text{-}Attr(\{w \in W \mid \mathcal{P}(w) < k\})$, vertices that may be considered at the moment irrelevant.

After discarding the resolved and currently irrelevant vertices, the algorithm proceeds in the remaining set of vertices that constitutes a proper subgame (i.e. without dead ends) induced by the set REM. After the subroutine returns, all vertices labelled with top are won by player $\Box$ in the subgame $G \cap \mathsf{REM}$. In other words, those vertices are won by $\Box$ provided that the play does not leave

REM. Since the only way for player $\Diamond$ to escape from REM is to visit RES, where every vertex is won by player $\Box$, the top-labelled vertices from REM are in fact won by $\Box$ in the context of the larger game $G \cap W$. Therefore the set DOM computed in line 28 constitutes an $\Box$-dominion within the game $G \cap W$, in which we have moreover fully defined a winning strategy σ for player $\Box$. Finally, every vertex from $V \setminus \mathsf{DOM}$ that can be attracted by player $\Box$ to the dominion DOM is certainly won by $\Box$, and for those vertices we assign the standard strategy attracting to DOM. The $\Box$-dominion A is then removed, and the computation continues in the remaining subgame.

The algorithm may at first sight appear to deviate much from the standard SPM algorithm. However, the additional overlay, apart from defining the strategy, can be seen as a special lifting policy that temporarily restricts the lifting to parts where an $\Box$ dominion is known to reside.

6.1 Correctness of the modified algorithm

The core of the correctness proof consists of showing that upon return procedure SPM-Within computes the winning strategy for player $\Box$ for all vertices in $\mathsf{Win}_\Box(G \cap W)$, provided that the input set W meets certain guards, which we call *suitability* conditions.

We need to define a few notions first. Let $A \subseteq V$ in the lifting context $\langle G, \rho \rangle$. We will say that A *has only nonprofitable* $\Box$*-escapes with respect to* ρ if for every $w \to u$ such that $w \in A \cap V_\Box$ and $u \in V \setminus A$, it holds that $u \in \Diamond\text{-}Attr_{\textsc{NonTop}(\rho)}(\{w' \in V \mid \mathcal{P}(w') < \mathsf{min}_{\mathcal{P}}(A)\})$, where $\textsc{NonTop}(\rho) = \{w' \in V \mid \rho(w') \neq \top\}$.

Moreover, we will say that A *has only top* $\Diamond$*-escapes with respect to* ρ, if for every $w \to u$ such that $w \in A \cap V_\Diamond$ and $u \in V \setminus A$, it holds that $\rho(u) = \top$.

We will call a subset $W \subseteq V$ *suitable* w.r.t ρ if:

S1 for all $w \in W$, $\rho(w) \neq \top$
S2 $G \cap W$ is a proper sugbame
S3 W has only nonprofitable $\Box$-escapes w.r.t. ρ
S4 W has only top $\Diamond$-escapes w.r.t. ρ

Theorem 4 captures the main idea behind our algorithm, and for the sake of understanding and readability it was singled out in a simplified form, as compared to the version that is formally required to prove correctness of the algorithm. The latter version is the following generalisation of Theorem 4, that allows us to reason about a context in which possibly more than one top value has occurred in the course of the lifting and certain parts of the game have already been resolved.

Theorem 5. *Suppose that $W \subseteq V$ induces a proper subgame and W has only nonprofitable $\Box$-escapes. Let ρ be a measure corresponding to a lifting sequence in which $v \in W$ is the only vertex in W such that $\rho(v) = \top$, and v was the last lifted vertex. Let $k = \mathcal{P}(v)$. Then there is a (memoryless) strategy σ, winning*

for $\Box$ in the context of the subgame $G \cap W$, such that all plays conforming to σ visit only vertices with priorities not smaller than k. Moreover, if $v \in V_\Box$, then $\sigma(v)$ is (one of the) maximal successor(s) of v w.r.t. ρ.

The key property concerning correctness of Algorithm 2 is proved in Proposition 6. The proposition utilises several lemmata, which we state below. Their proofs can be found in the appendix.

Lemma 7. *If W is a subgame, then the set $\mathsf{REM} = W \setminus (\mathsf{RES} \cup \mathsf{IRR})$ that is computed in line 26 induces a subgame (i.e. it does not have "dead ends").*

Lemma 8. *Suppose some arbitrary lifting procedure has been applied on the entire G, yielding a temporary measure $\bar{\rho}$ Assume that W is a set of vertices that induces a well-defined subgame of G, $G \cap W$, and moreover the only edges leading from the even-owned vertices in W to $G \setminus W$, have top-labelled vertices as endpoints. Furthermore, suppose that $D \subseteq W$ induces an $\Box$-dominion on the subgame $G \cap W$. Then lifting of ρ restricted to W will finally yield a top value.*

Lemma 9. *Let D be a dominion within a game G. Let $D' \subseteq D$ be a nonempty subset of D such that D' has only $\Diamond$-escapes to $D \setminus D'$. That is, for all $u \to w$ such that $u \in D'$ and $w \in D \setminus D'$, it holds that $u \in V_\Diamond$. Then D' is a dominion within any subgame G' containing the entire D', but not containing any vertices from $D \setminus D'$.*

We are now in the position to prove the key result of this section (here, we provide a high-level description of the main steps of the proof, and for the details we refer to the appendix).

Proposition 6. *Assume a lifting context $\langle G, ms \rangle$. Suppose that during the execution of the procedure SPM-Within, before some while-loop iteration (line 11), ρ has a certain value ρ_I, and it holds that W is suitable w.r.t. ρ_I. Let ρ_F be the final value of ρ when SPM-Within returns. Then after executing SPM-Within, the following hold:*

- *for all $w \in W$, $w \in \mathsf{Win}_\Box(G \cap W) \iff \rho_F(w) = \top$*
- *$\sigma|_{\mathsf{Win}_\Box(W)}$ is winning for $\Box$ in the context of a subgame $G \cap W$*

Proof. We proceed with structural induction on W; assume that the statement holds for all suitable subsets of W. We will prove that it holds for W.

I If $\mathsf{Win}_\Box(G \cap W) \neq \emptyset$, then the iteration of liftings in lines 12–14 will eventually lead to a top-value in some vertex v.
II Vertex v defined in line 16 belongs to $\mathsf{Win}_\Box(G \cap W)$
III In line 26, REM is suitable w.r.t. ρ
IV For any $D \subseteq (\mathsf{RES} \cup \mathsf{REM})$ which is an $\Box$ dominion in the context of $G \cap W$, all vertices in $D \setminus \mathsf{RES}$ are also won by $\Box$ in $G \cap \mathsf{REM}$, i.e. $D \setminus \mathsf{RES} \subseteq \mathsf{Win}_\Box(G \cap \mathsf{REM})$

V $\sigma|_{\mathsf{REM}}$ is winning for $\Box$ on $\mathsf{Win}_\Box(G \cap \mathsf{REM})$
VI $\sigma|_{\mathsf{RES} \cup \mathsf{Win}_\Box(G \cap \mathsf{REM})}$ is a winning strategy for $\Box$ in $\mathsf{Win}_\Box(G \cap \mathsf{REM}) \cup \mathsf{RES}$ in the context of the subgame $G \cap W$
VII $\sigma|_A$ is a winning strategy for $\Box$ in A in the context of the subgame $G \cap W$
VIII if the new $W^{new} := W \setminus A$, computed in line 35, is not empty, then it is suitable w.r.t. ρ.

Theorem 6. *SPM-Within returns the least game progress measure of G, and the strategy σ, fully defined on $\mathsf{Win}_\Box(G) \cap V_\Box$, is a winning strategy of player $\Box$ on $\mathsf{Win}_\Box(G)$.*

Proof. As V is obviously suitable w.r.t. the initial $\rho = \lambda v.(0, \ldots, 0)$, from Proposition 6 we immediately obtain correct resolution of the $\mathsf{Win}_\Box(G)$ part.

Running time Every attractor computation takes $O(n+m)$ time, and whenever it occurs, at least one new vertex is 'resolved'. Hence the total extra time introduced by the attractor computations is bounded by $O(n(n+m))$. As with the standard SPM, the lifting operations dominate the running time, and their total number for every vertex is bounded by the size of $\mathbb{M}^\Diamond$.

Theorem 7. *For a game G consisting of n vertices, m edges, and d priorities, the extended algorithm solves G and computes winning strategies for player $\Diamond$ and $\Box$ in worst-case $O(dm \cdot (n/\lfloor d/2 \rfloor)^{\lfloor d/2 \rfloor})$.*

Proof.

$$O\left(n\,(n+m) + dm\left(\frac{n}{\lfloor \frac{d}{2} \rfloor}\right)^{\lfloor \frac{d}{2} \rfloor}\right) = O\left(dm\left(\frac{n}{\lfloor \frac{d}{2} \rfloor}\right)^{\lfloor \frac{d}{2} \rfloor}\right)$$

Example 4. We illustrate the various aspects of Algorithm 2 on the game G depicted in Figure 5, with two (overlapping) subgames G_1 and G_2.

Note that the entire game is an $\Box$-paradise: every vertex eventually is assigned measure $\top$ by Algorithm 2 (and Algorithm 1, for that matter). Suppose we use a lifting strategy prioritising v_2, v_3, v_7 and v_8; then vertex v_3's measure is the first to reach $\top$, and the successor with maximal measure is v_7. Therefore, $\Box$'s strategy is to move from v_3 to v_7. The set RES, computed next consists of vertices v_3 and v_2; the strategy for v_2 is set to v_3 and its measure is set to $\top$. The $\Diamond$-attractor into those vertices with priorities ≥ 3, *i.e.*, vertices v_1 and v_4, is exactly those vertices, so, next, the algorithm zooms in on solving the subgame G_1.

Suppose that within the latter subgame, we prioritise the lifting of vertex v_7 and v_8; then vertex v_7's measure is set to $\top$ first, and v_7's successor with the largest measure is v_8; therefore $\Box$'s strategy is to move from v_7 to v_8. At this point in the algorithm, RES is assigned the set of vertices v_7 and v_8, and the measure of v_8 is set to $\top$. Note that in this case, in this subgame, the winning strategy for $\Box$ on v_7 is to remain within the set RES. Since all remaining vertices have more signficant priorities than v_7, or are forced by $\Diamond$ to move there, the next

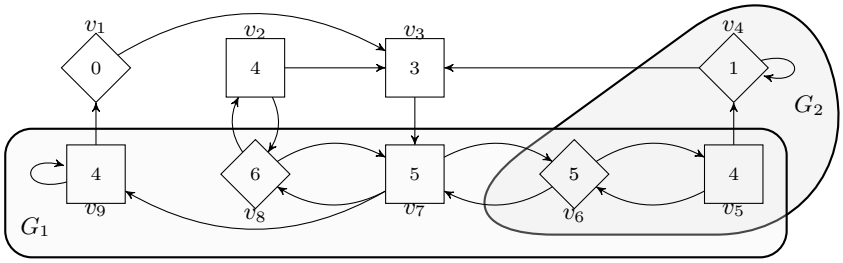

Figure 5. An example game G with two (overlapping) subgames G_1 and G_2.

recursion is run on an empty subgame and immediately returns without changing the measures. Upon return, the $\Box$-attractor to all $\Box$-won vertices (within the subgame G_1, so these are only the vertices v_7 and v_8) is computed, and the algorithm continues solving the remaining subgame (*i.e.* the game restricted to vertices v_5, v_6 and v_9), concluding that no vertex in this entire game will be assigned measure $\top$.

At this point, the algorithm returns to the global game again and computes the $\Box$-attractor to the vertices won by player $\Box$ at that stage (*i.e.* vertices v_2, v_3, v_7 and v_8), adding vertices v_1 and v_9, setting their measure to $\top$ and setting $\Box$'s strategy for v_9 to move to v_1.

As a final step, the algorithm next homes in on the subgame G_2, again within the larger game. The only vertex assigned measure $\top$ in the above subgame is vertex v_4; at this point RES is assigned all vertices in G_2, the measure of v_5 and v_6 is set to $\top$ and the $\Box$ strategy for vertex v_5 is set to v_4. This effectively solves the entire game.

7 Conclusions and Future Work

The two key contributions of our work are:

1. We have proposed a more operational interpretation of progress measures by characterising the types of plays that players can enforce.
2. We have provided a modification of the SPM algorithm that allows to compute the winning strategies for both players in one pass, thus improving the worst-case running time of strategy derivation.

The second enhancement has been made possible due to a thorough study of the contents of progress measures, and their underapproximations in the intermediate stages of the algorithm (building on the proposed operational interpretation).

As for the future work, we would like to peform an analysis of SPM behaviour on special classes of games, along the same lines as we have done in case of the

recursive algorithm [8], specifically, identifying the games for which SPM runs in polynomial time, and studying enhancements that allow to solve more types of games efficiently.

References

1. A. Arnold, A. Vincent, and I. Walukiewicz. Games for synthesis of controllers with partial observation. *TCS*, 303(1):7–34, 2003.
2. C. Dittmann, S. Kreutzer, and A. I. Tomescu. Graph operations on parity games and polynomial-time algorithms. *CoRR*, abs/1208.1640, 2012.
3. E.A. Emerson and C.S. Jutla. Tree automata, mu-calculus and determinacy. In *FOCS'91*, pages 368–377, Washington, DC, USA, 1991. IEEE Computer Society.
4. E.A. Emerson, C.S. Jutla, and A.P. Sistla. On model-checking for fragments of μ-calculus. In *CAV*, volume 697 of *Lecture Notes in Computer Science*, pages 385–396. Springer, 1993.
5. J. Fearnley. Non-oblivious strategy improvement. In *LPAR-16*, volume 6355 of *Lecture Notes in Computer Science*, pages 212–230. Springer, 2010.
6. O. Friedmann. Recursive algorithm for parity games requires exponential time. *RAIRO – Theor. Inf. and Applic.*, 45(4):449–457, 2011.
7. O. Friedmann and M. Lange. Solving parity games in practice. In *ATVA*, volume 5799 of *Lecture Notes in Computer Science*, pages 182–196. Springer, 2009.
8. M. Gazda and T.A.C. Willemse. Zielonka's recursive algorithm: dull, weak and solitaire games and tighter bounds. In *GandALF*, volume 119 of *EPTCS*, pages 7–20, 2013.
9. E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *LNCS*. Springer, 2002.
10. M. Jurdziński. Small progress measures for solving parity games. In *STACS'00*, volume 1770 of *LNCS*, pages 290–301. Springer, 2000.
11. M. Jurdziński, M. Paterson, and U. Zwick. A Deterministic Subexponential Algorithm for Solving Parity Games. In *SODA'06*, pages 117–123. ACM/SIAM, 2006.
12. H. Klauck. Algorithms for parity games. In *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*, chapter 7, pages 107–129. Springer, 2001.
13. R. McNaughton. Infinite games played on finite graphs. *APAL*, 65(2):149–184, 1993.
14. S. Schewe. Solving parity games in big steps. In *FSTTCS'07*, volume 4855 of *LNCS*, pages 449–460. Springer, 2007.
15. S. Schewe. An optimal strategy improvement algorithm for solving parity and payoff games. In *CSL*, volume 5213 of *Lecture Notes in Computer Science*, pages 369–384. Springer, 2008.
16. P. Stevens and C. Stirling. Practical model checking using games. In *TACAS'98*, volume 1384 of *LNCS*, pages 85–101. Springer, 1998.
17. J. Vöge and M. Jurdziński. A discrete strategy improvement algorithm for solving parity games. In *CAV*, volume 1855 of *Lecture Notes in Computer Science*, pages 202–215. Springer, 2000.
18. W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *TCS*, 200(1-2):135 – 183, 1998.

Appendix

Proofs of Section 5.1

Proposition 4. *If there is a non-trivial path in LH (i.e. containing at least one edge) from (w, m) to (w, m'), then $m > m'$.*

Proof. Suppose there is a path in LH $(w, m) = (w_0, m_0) \to (w_1, m_1) \to \cdots \to (w_n, m_n) = (w, m')$. Let ρ_t denote the value of ρ after the t-th lifting; for $j \in \{0, \ldots, n\}$ we will define t_j as the step in which w_j was lifted to m_j. From monotonicity of lifting and the fact that two vertices cannot be lifted at the same moment, it is not difficult to observe that $t_j > t_{j+1}$, and hence from transitivity we have $t_0 > t_n$ We consider two cases:

- if $n > 1$, then

$$m = \rho_{t_0}(w_0) \overset{\text{def. of } t_0}{>} \rho_{t_0-1}(w_0) \overset{t_0-1 \geq t_n}{\geq} \rho_{t_n}(w_0) = \rho_{t_n}(w_n) = m'$$

- if $n = 1$, then

$$m = \rho_{t_0}(w_0) \overset{\text{def. of } t_0}{>} \rho_{t_0-1}(w_0) = \rho_{t_n}(w_n) = m'$$

Lemma 3. *If $m \neq \top$ and $stretch(k) < kval$ (i.e. the termination condition is not satisfied), then there is a position $L \geq k$ such that for all odd $i \leq L$:*

position in m *(i ranges over odd numbers)*	***invariant***	
$i < k$	$(m)_i = \lvert V_i \rvert$ *and* $stretch(i) = 0$	*(C1)*
$i = k$	*if* $L = k$, *then* $stretch(k) + (m)_k \geq kval$	*(C21)*
	if $L > k$, *then* $stretch(k) + (m)_k \geq kval - 1$	*(C22)*
$k < i < L$	$stretch(i) + (m)_i \geq \lvert V_i \rvert$	*(C3)*
$i = L$ *and* $L > k$	$stretch(L) + (m)_L \geq \lvert V_L \rvert + 1$	*(C4)*

Moreover, the inequalities at position L can be further strengthened if L coincides with the priority of the current state:

- *if* $\mathcal{P}(u) = L = k$, *then* $stretch(k) + (m)_k \geq kval + 1$ *(C5)*
- *if* $\mathcal{P}(u) = L > k$, *then* $stretch(L) + (m)_L \geq \lvert V_L \rvert + 2$ *(C6)*

The above invariant implies in particular that $(m)_L > 0$, and $\mathcal{P}(u) \geq k$.

$i < k$	$i = k$	$k < i < L$	$i = L$
saturated	lower bound on $stretch(i) + (m)_i$		
$(m)_i = \lvert V_i \rvert$	$kval$	$\lvert V_i \rvert$	$\lvert V_L \rvert + 1$ ($kval + 1$ if $L = k$)

Proof. Let us start with proving the last observation: if the invariant holds and the termination condition is not satisfied, then $(m)_L > 0$, and $k \leq L \leq \mathcal{P}(u)$.

Suppose that the invariant holds and $m \neq \top$ and $stretch(k) < kval$. Then there must exist $L \geq k$ with the above properties (C1-C6). If $L = k$, then from C21 we have $(m)_L \geq kval - stretch(L)$. Since $stretch(L) = stretch(k) < kval$, we obtain $(m)_L > 0$. If $L > k$, then we have $(m)_L \geq |V_L| - stretch(L) + 1$ (C4). Observe that $stretch(L) \leq |V_L|$, because a longer stretch than V_L means that a cycle has been encountered in which L was the dominating priority, an in this case OddResponse prunes the list *vis* and reverts to the previously encountered LH state (step 2). Hence in this case we have $(m)_L > 0$ as well. Since for all states in the lifting history graph (w, m') it holds that $(m')_i = 0$ for i larger than $\mathcal{P}(w)$ (Proposition 3), $(m)_L > 0$ implies that $k \leq L \leq \mathcal{P}(u)$.

Initialisation We proceed to prove that the invariant holds at the start of the procedure OddResponse, when $u = v_0$. We simply take $L = k$. Since at this point m is of the form

$$(0, |V_1|, 0, , \ldots, |V_{k-2}|, 0, kval, * * *)$$

the condition that positions smaller than k are saturated (C1) is satisfied. Since $stretch(k) \geq 0$ and $(m)_k = kval$, C21 holds as well. In case when $P(u) = L = k$, we have $stretch(k) \geq 1$ (at least u belongs to the k-dominated stretch), hence $stretch(k) + (m)_k \geq kval + 1$ and thus C5 holds.

Maintenance We will now show that the invariant is always maintained. Suppose the invariant held on each step before the current one; we show that it must hold after the entire current step (processing the current state in lines 1–5) is finished.

In case the termination condition in line 1 is not satisfied, we proceed and check for the odd cycle (step 2). If an odd cycle has been encountered, a certain suffix is removed from *vis* and we revert to the previous state containing u, say (u, m') with a pruned list $vis = vis_1 \ldots vis_{j-1}$. This state has already been encountered, and from the assumption we know that the invariant must have held there. Hence the invariant holds before we enter step 3 of OddResponse.

In steps 3-5 the list *vis* is extended with the current state and the new state (u, m) is picked by one of the players. Let us denote the old and new values of (u, m) and other artefacts with superscripts "O" and "N" respectively, e.g. (u^O, m^O) and (u^N, m^N).

Suppose that the termination condition is not satisfied in the new state, in particular $m^N \neq \top$. We need to show that (u^N, m^N) together with the extended list $vis.(u^O, m^O)$ meet the requirements C1-C6.

From Proposition 3 we can derive certain relationships between m^O and m^N:

(*) Let $gnz = \max\{l \mid (m^O)_l > 0\}$. Then either:
1. $m^N \geq_{gnz} m^O$, or
2. $m^N =_{gnz-1} m^O$ and $(m^N)_{gnz} = (m^O)_{gnz} - 1$
   and for all $j \in \{gnz + 1, \ldots, \mathcal{P}(u^O)\}(m^N)_j = |V_j|$

Moreover, the second case is possible only if $\mathcal{P}(u^O)$ is odd.

Note that $L^O \leq gnz$.

Let us first explain that all odd positions smaller than k must remain saturated (C1). Since $(m^O)_{L^O} > 0$, we have $k \leq L^O \leq gnz$, and hence from (*) $m^N \geq_{k-1} m^O$. Therefore if $m^N \neq \top$, then for all odd $i < k$, $(m^N)_i = |V_i|$.

We proceed to prove that the lower bounds for $stretch(j) + (m)_j$ hold on positions $k \leq j \leq L^N$ (C2-C6).

Let us first make a straightforward remark that for all odd $j < \mathcal{P}(u^N)$, $stretch^N(j) = stretch^O(j)$, and if moreover $\mathcal{P}(u^N)$ is odd, then $stretch^N(\mathcal{P}(u^N)) = stretch^O(\mathcal{P}(u^N)) + 1$.

We consider the following cases:

- if $\mathcal{P}(u^N) < L^O$: in this case $\mathcal{P}(u^N) < L^O \leq gnz$; we first prove that $m^N >_{\mathcal{P}(u^N)} m^O$.
  From $m^N \geq_{gnz-1} m^O$ and $\mathcal{P}(u^N) < gnz$, we obtain $m^N \geq_{\mathcal{P}(u^N)} m^O$. But then it must be the case that $m^N >_{\mathcal{P}(u^N)} m^O$, because the suffix of m^N consisting of positions larger than $\mathcal{P}(u^N)$ contains only zeros, and the corresponding suffix in m^O has at least one position with a non-zero value, namely L^O. Hence the smaller suffix in m^N must be compensated by a strictly larger prefix up to $\mathcal{P}(u^N)$.
  We have thus established that $m^N >_{\mathcal{P}(u^N)} m^O$. We can now take as the new L^N the smallest j such that $(m^N)_j > (m^O)_j$ ($k \leq j$, because m^O was saturated up to k). The invariant held at (u^O, m^O), so the inequality C22 or C3 held at position L^N; by increasing $(m^N)_{L^N}$ at least by 1, and with the same (at least) length of L^N-dominated stretch, we obtain inequality with a strengthened right-hand side – C21, or C4 respectively. If it is also the case that $L^N = \mathcal{P}(u^N)$, then $stretch^N(L^N) = stretch^O(L^N) + 1$, therefore the lower bound can be further strengthened by 1, yielding C5 or C6, respectively.
- if $\mathcal{P}(u^N) \geq L^O$:
  - if $m^N >_{L^O} m^O$, then L^N can be defined in exactly the same manner as above (the smallest $j \geq k$ such that $(m^N)_j > (m^O)_j$), and the same argument applies
  - if $m^N =_{L^O} m^O$, then we can take $L^N := L^O$; since lengths of stretches remain the same – apart from possibly $stretch(L^O)$, which increases by 1 if $\mathcal{P}(u^N) = L^O$, the required inequalities are easily derived from those that held in the previous state (u^O, m^O)
  - if $m^N <_{L^O} m^O$: in this case we must be in the second case of (*) and $\mathcal{P}(u^O)$ is odd. We also have $L^O = gnz$, because on one hand $L^O \leq gnz$, and on the other gnz has the property that it is the smallest position j on which $m^N <_j m^O$.
    - if $gnz = L^O = \mathcal{P}(u^O)$, then we take $L^N := L^O$. We have $stretch^N(L^O) = stretch^O(L^O) + 1$, hence the fact that value at position L^O in the tuple has been decremented is compensated by a larger value of the stretch, and the invariant is preserved.
    - if $gnz = L^O < \mathcal{P}(u^O)$, then we take $L^N := \mathcal{P}(u^O)$. The value at position L^O in m^N has been decremented as compared to m^O, but it

was the position on which an inequality with a strengthened right-hand side (C21 or C4) held, therefore a weaker condition (C22 or C3) will be satisfied in the new state. Furthermore, from Proposition 3 we have $(m^N)_j = |V_j|$ for all odd $j \leq L^N$, and since $stretch^N(j) \geq 0$, the condition C3 is readily satisfied. Finally, for position L^N we have $stretch^N(L^N) \geq 1$, and $(m^N)_{L^N} = |V_{L^N}|$, hence C4 holds, and if it also happens that $\mathcal{P}(u^N) = L^N = \mathcal{P}(u^O)$, then $stretch^N(L^N) \geq 2$, and C6 holds as well.

Theorem 5. *Suppose that $W \subseteq V$ induces a proper subgame and W has only nonprofitable $\Box$-escapes. Let ρ be a measure corresponding to a lifting sequence in which $v \in W$ is the only vertex in W such that $\rho(v) = \top$, and v was the last lifted vertex. Let $k = \mathcal{P}(v)$. Then there is a (memoryless) strategy σ, winning for $\Box$ in the context of the subgame $G \cap W$, such that all plays conforming to σ visit only vertices with priorities not smaller than k. Moreover, if $v \in V_\Box$, then $\sigma(v)$ is (one of the) maximal successor(s) of v w.r.t. ρ.*

Proof. Consider the strategy σ from Lemma 5. No play conforming to σ can visit a vertex with priority smaller than k, unless it is a top-labelled node occurring after a prefix of non-top nodes having priorities at least k. Therefore no such play can enter $\Diamond\text{-}Attr_{\textsc{NonTop}(\rho)}(\{w' \in V \mid \mathcal{P}(w') < k\}) \subseteq \Diamond\text{-}Attr_{\textsc{NonTop}(\rho)}(\{w' \in V \mid \mathsf{min}_{\mathcal{P}}(W)\})$. From this and the fact that W has only non-profitable $\Box$-escapes, we know that $\sigma(w) \in W$ for all $w \in \mathbf{dom}(\sigma) \cap W$. Hence σ is a well-defined strategy in the context of the subgame $G \cap W$.

Consider any play π conforming to σ within $G \cap W$. In case when π is finite and visits the first top-labelled vertex, it can only be v, on which σ is defined. From this cyclic property we know that for all vertices $w \in V_\Box$ encountered in π, σ is defined. If π visits v inifinitely often, then π is winning for $\Box$, because π does not visit any more significant priority in between. If v is visited only finitely many times, from Lemma 5 we know that π is winning for $\Box$. In both cases no priority more significant than k is encountered.

Proofs of Section 6.1

Lemma 7. *If W is a subgame, then the set $\mathsf{REM} = W \setminus (\mathsf{RES} \cup \mathsf{IRR})$ that is computed in line 26 induces a subgame (i.e. it does not have "dead ends").*

Proof. Since G is a well-defined game, we know that $u^\bullet \neq \emptyset$. Suppose, towards contradiction, that there is a node $u \in V \setminus (\mathsf{RES} \cup \mathsf{IRR})$ such that $u^\bullet \subseteq \mathsf{RES} \cup \mathsf{IRR}$. We distinguish two cases:

- if $u \in V_\Diamond$, then, since $V \setminus \mathsf{IRR}$ is an $\Box$-closed set, we have $u^\bullet \cap \mathsf{IRR} = \emptyset$, so the only possibility is that $u^\bullet \subseteq \mathsf{RES}$. However, since $u \in V \setminus (\mathsf{RES} \cup \mathsf{IRR})$, we have $\mathcal{P}(u) \geq k$ and hence it must be the case that $u \in \Box\text{-}Attr^{\geq k}(\mathsf{RES}) = \mathsf{RES}$, a contradiction.
- if $u \in V_\Box$, then $u^\bullet \cap \mathsf{RES} = \emptyset$ (because $\mathcal{P}(u) \geq k$ and in that case we would have $u \in \Box\text{-}Attr^{\geq k}(\mathsf{RES}) = \mathsf{RES}$). Therefore $u^\bullet \subseteq \mathsf{IRR}$ - but it means that $u \in \Diamond\text{-}Attr(\mathsf{IRR}) = \mathsf{IRR}$, a contradiction.

Lemma 8. *Suppose some arbitrary lifting procedure has been applied on the entire G, yielding a temporary measure $\bar{\rho}$ Assume that W is a set of vertices that induces a well-defined subgame of G, $G \cap W$, and moreover the only edges leading from the even-owned vertices in W to $G \setminus W$, have top-labelled vertices as endpoints. Furthermore, suppose that $D \subseteq W$ induces an $\Box$-dominion on the subgame $G \cap W$. Then lifting of ρ restricted to W will finally yield a top value.*

Proof. We can transform G to a game G', in which we remove all edges from W, to $V \setminus W$. The codomain of SPM for G' is the same as for G. Lifiting in G' must reach a top because of D, and it will yield smaller values that lifting in G. Indeed, the removed edges leading from $W \cap V_\Diamond$ (vertices on which min is taken) outside lead only to top-labelled vertices, so the effect is the same as removing these edges. Edges originating in $W \cap V_\Box$ may only increase the measure.

The purpose of the following lemma is to establish that the remainder of the dominion containing RES, and contained in REM, is a dominion within REM. Note that the subgame G' mentioned in the lemma may actually not exist (however, in our case it always exists (REM)– see Lemma 7).

Lemma 9. *Let D be a dominion within a game G. Let $D' \subseteq D$ be a nonempty subset of D such that D' has only $\Diamond$-escapes to $D \setminus D'$. That is, for all $u \to w$ such that $u \in D'$ and $w \in D \setminus D'$, it holds that $u \in V_\Diamond$. Then D' is a dominion within any subgame G' containing the entire D', but not containing any vertices from $D \setminus D'$.*

Proof. Consider the $\Box$ strategy σ, winning for $\Box$, and closed on D. Since there are no $\Box$-escapes from D' to $D \setminus D'$, for any $w \in \mathbf{dom}(\sigma) \cap D'$ we have $\sigma(w) \in D'$. Hence for any subgame G' containing D', but not any vertex from $D \setminus D'$, the strategy σ restricted to D' is well-defined, and the two desired properties are carried over from the original strategy.

Proposition 6. *Assume a lifting context $\langle G, ms \rangle$. Suppose that during the execution of the procedure SPM-Within, before some while-loop iteration (line 11), ρ has a certain value ρ_I, and it holds that W is suitable w.r.t. ρ_I. Let ρ_F be the final value of ρ when SPM-Within returns. Then after executing SPM-Within, the following hold:*

- *for all $w \in W$, $w \in \mathsf{Win}_\Box(G \cap W) \iff \rho_F(w) = \top$*
- *$\sigma|_{\mathsf{Win}_\Box(W)}$ is winning for $\Box$ in the context of a subgame $G \cap W$*

Proof. We proceed with structural induction on W; assume that the statement holds for all suitable subsets of W. We will prove that it holds for W.

I If $\mathsf{Win}_\Box(G \cap W) \neq 0$, then the iteration of liftings in lines 12–14 will eventually lead to a top-value in some vertex v.
Proof Follows from Lemma 8 and the assumption of W having only top $\Diamond$-escapes.

II Vertex v defined in line 16 belongs to $\mathsf{Win}_\Box(G \cap W)$
Proof Immediately from Theorem 5.

III In line 26, REM is suitable w.r.t. ρ

Proof

S1 obvious, since v is the only vertex in W such that $\rho(v) = \top$, and $v \notin \mathsf{REM}$

S2 proved in Lemma 7

S3 from the assumption about W, all $\Box$ escapes from REM outside W are non-profitable. Consider any $\Box$-escape to $W \setminus \mathsf{REM}$, that is, $u \to w$ such that $u \in \mathsf{REM} \cap V_\Box$, and $u \in W \setminus \mathsf{REM} = \mathsf{IRR} \cup \mathsf{RES}$. Suppose towards contradiction that $w \notin \mathsf{IRR}$, hence $w \in \mathsf{RES}$. But since $\mathcal{P}(u) \geq k$ and $u \in W \cap V_\Box$, we have $u \in \mathsf{RES}$, and therefore $u \notin \mathsf{REM}$, a contradiction.

S4 from the assumption about W, all $\Diamond$-escapes from REM outside W are to top-labelled nodes. Observe that no even-owned vertex $w \in \mathsf{REM}$ can have an edge to $\mathsf{IRR} = \Diamond\text{-}Attr_W(\{w \in W \mid \mathcal{P}(w) < k\})$, because w would then belong to IRR. Hence the only $\Diamond$-escapes outside REM lead to RES, and every vertex therein has measure top.

IV For any $D \subseteq (\mathsf{RES} \cup \mathsf{REM})$ which is an $\Box$ dominion in the context of $G \cap W$, all vertices in $D \setminus \mathsf{RES}$ are also won by $\Box$ in $G \cap \mathsf{REM}$, i.e. $D \setminus \mathsf{RES} \subseteq \mathsf{Win}_\Box(G \cap \mathsf{REM})$

Proof Follows from Lemma 9 and the fact that there are no $\Box$-escapes from REM to RES.

V $\sigma|_{\mathsf{REM}}$ is winning for $\Box$ on $\mathsf{Win}_\Box(G \cap \mathsf{REM})$

Proof Follows from the inductive hypothesis.

VI $\sigma|_{\mathsf{RES} \cup \mathsf{Win}_\Box(G \cap \mathsf{REM})}$ is a winning strategy for $\Box$ in $\mathsf{Win}_\Box(\mathsf{REM}) \cup \mathsf{RES}$ in the context of the subgame $G \cap W$

Proof Follows from Lemma 6 and the previous point.

VII $\sigma|_A$ is a winning strategy for $\Box$ in A in the context of the subgame $G \cap W$

Proof Follows from the previous point and the obvious fact that extending the dominion with its attractor, and assigning the attracting strategy for the extended part yields a winning strategy.

VIII if the new $W := W \setminus A$, computed in line 35, is not empty, then it is suitable w.r.t. ρ

S1 by removing A, all top-labelled vertices have been removed from W

S2 W, as a complement of an $\Box$-attractor, is $\Diamond$-closed, and constitutes a proper subgame

S3 the new W doesn't have any additional $\Box$-escapes as compared to the old one

S4 the only possible new $\Diamond$-escapes lead to A, in which all vertices are top-labelled