A Delayed Promotion Policy for Parity Games[☆]

Massimo Benerecetti[a], Daniele Dell'Erba[a], Fabio Mogavero[b]

[a] *Università degli Studi di Napoli Federico II, Napoli, Italy*
[b] *University of Oxford, Oxford, UK*

Abstract

Parity games are two-player infinite-duration games on graphs that play a crucial role in various fields of theoretical computer science. Finding efficient algorithms to solve these games in practice is widely acknowledged as a core problem in formal verification, as it leads to efficient solutions of the model-checking and satisfiability problems of expressive temporal logics, *e.g.*, the modal μCALCULUS. Their solution can be reduced to the problem of identifying sets of positions of the game, called dominions, in each of which a player can force a win by remaining in the set forever. Recently, a novel technique to compute dominions, called *priority promotion*, has been proposed, which is based on the notions of quasi dominion, a relaxed form of dominion, and dominion space. The underlying framework is general enough to accommodate different instantiations of the solution procedure, whose correctness is ensured by the nature of the space itself. In this paper we propose a new such instantiation, called *delayed promotion*, that tries to reduce the possible exponential behaviors exhibited by the original method in the worst case. The resulting procedure often outperforms both the state-of-the-art solvers and the original priority promotion approach.

Keywords: Parity games; Infinite-duration games on graphs; Formal methods.

1. Introduction

The abstract concept of *game* has proved to be a fruitful metaphor in theoretical computer science [2]. Several *decision problems* can, indeed, be encoded as *path-forming games on graphs*, where a player willing to achieve a certain goal, usually the verification of some property on the plays derived from the original problem, has to face an opponent whose aim is to pursue the exact opposite task. One of the most prominent instances of this connection is represented by the notion of *parity game* [3], a simple two-player turn-based perfect-information game played on directed graphs, whose nodes are labeled with natural numbers called *priorities*. The goal of the first (*resp.*, second) player, *a.k.a.*, *even* (*resp.*, *odd*) player, is to force a play π, whose maximal priority occurring infinitely often along π is of even (*resp.*, odd) parity. The importance of these games is due to the numerous applications in the area of system specification, verification, and

[☆] This work is based on [1], which appeared in GandALF'16.

synthesis, where it is used as algorithmic back-end of satisfiability and model-checking procedures for temporal logics [4, 5, 6], logics for games [7, 8, 9, 10, 11, 12, 13, 14], and as a core for several techniques employed in automata theory [15, 16, 17, 18]. In particular, it has been proved to be linear-time interreducible with the model-checking problem for the *modal* μCALCULUS [5] and it is closely related to other games of infinite duration, such as *mean payoff* [19, 20], *discounted payoff* [21], *simple stochastic* [22], *energy* [23] games, and *prompt* games [24, 25, 26, 27]. Besides the practical importance, parity games are also interesting from a computational complexity point of view, since their solution problem is one of the few inhabitants of the UPTIME $\cap$ COUPTIME class [28]. That result improves the NPTIME $\cap$ CONPTIME membership [5], which easily follows from the property of *memoryless determinacy* [16, 3]. Still open is the question about the membership in PTIME.

The literature on the topic is reach of algorithms for solving parity games, which can be mainly classified into two families. The first one contains the algorithms that, by employing a *divide et impera* approach, recursively decompose the problem into subproblems, whose solutions are then suitably assembled to obtain the desired result. In this category fall, for example, *Zielonka's recursive algorithm* [29] and its *dominion decomposition* [30] and *big step* [31] improvements. The second family, instead, groups together those algorithms that try to compute a winning strategy for the two players on the entire game. The principal members of this category are represented by *Jurdziński's progress measure* algorithm [32] and the *strategy improvement* approaches [33, 8, 34].

A recent breakthrough [35] by Calude *et al.* proposes a succinct reduction from parity to reachability games based on a clever encoding of the sequences of priorities that a player finds along a play. This allows for a mere quasi-polynomial blow up in the size of the underlying graph and sets the basis of the fixed-parameter tractability *w.r.t.* the number of priorities. The approach has been then considerably refined in [36], where these encodings are modeled as progress measures. A similar technique is also used in [37]. Despite the theoretical relevance of this new idea, preliminary experiments seem to suggest that the practical impact of the result does not match the theoretical one, as all exponential algorithms outperform, often by orders of magnitude, the current implementations of the quasi-polynomial ones, which do not scale beyond few hundred vertexes. This evaluation is consistent with the fact that the new techniques essentially amount to clever and succinct encodings embedded within a brute force search, which makes matching quasi-polynomial worst cases quite easy to find.

Recently, a new *divide et impera* solution algorithm, called *priority promotion* (PP, for short), has been proposed in [38] and further developed in [39], which is fully based on the decomposition of the winning regions into *dominions*. The idea is to find a dominion for some of the two players and then remove it from the game, thereby allowing for a recursive solution. The important difference *w.r.t.* the other two approaches [30, 31] based on the same notion is that these procedures only look for dominions of a certain size in order to speed up classic Zielonka's algorithm in the worst case. Consequently, they strongly rely on this algorithm for their completeness. On the contrary, the PP procedure autonomously computes dominions of any size, by suitably composing quasi dominions, a weaker notion of dominion. Intuitively, a *quasi dominion* Q for player $\alpha \in \{0, 1\}$ is a set of vertices from each of which player α can enforce a winning play that never leaves the region, unless one of the following two conditions holds: *(i)* the

opponent $\overline{\alpha}$ can escape from Q (*i.e.*, there is an edge from a vertex of $\overline{\alpha}$ exiting from Q) or *(ii)* the only choice for player α itself is to exit from Q (*i.e.*, no edge from a vertex of α remains in Q). A crucial feature of quasi dominion is that they can be ordered by assigning to each of them a measure corresponding to an under-approximation of the best priority for α the opponent $\overline{\alpha}$ can be forced to visit along any play exiting from it. Indeed, under suitable and easy to check assumptions, a higher priority quasi α-dominion Q_1 and a lower priority one Q_2, can be merged into a single quasi α-dominion of the higher priority, thus improving the approximation for Q_2. This merging operation is called a priority promotion of Q_2 to Q_1. A refinement of this approach, where the attention is restricted to strongly connected quasi-dominions called tangles, has also been recently proposed in [40].

The PP solution procedure has been shown to be very effective in practice and to often significantly outperform all other solvers. Moreover, it also improves on the space complexity of the best known algorithm with an exponential gain *w.r.t.* the number of priorities and by a logarithmic factor *w.r.t.* the number of vertexes. Indeed, it only needs $\mathrm{O}(n \cdot \log k)$ space against the $\mathrm{O}(k \cdot n \cdot \log n)$ required by Jurdziński's approach [32], where n and k are, respectively, the numbers of vertexes and priorities of the game. It also improves *w.r.t.* the recently introduced quasi-linear space algorithms [36, 37]. Unfortunately, the PP algorithm also exhibits exponential behaviors on a simple family of games. This is due to the fact that, in general, promotions to higher priorities requires resetting promotions previously performed at lower ones.

In this article, we continue the study of the priority promotion approaches trying to find a remedy to this problem. We propose a new algorithm, called DP, built on top of a slight variation of PP, called PP+. The PP+ algorithm simply avoids resetting previous promotions to quasi dominions of the same parity. In this case, indeed, the relevant properties of those quasi dominions are still preserved. This variation enables the new DP promotion policy, that delays promotions that require a reset and only performs those leading to the highest quasi dominions among the available ones. Experiments on randomly generated games show that the new approach performs much better than PP in practice, while still preserving the same space complexity.

2. Preliminaries

Let us briefly recall the notation and basic definitions concerning parity games. We refer to [2, 29] for a comprehensive presentation of the subject.

Given a partial function $\mathsf{f} : \mathrm{A} \rightharpoonup \mathrm{B}$, by $\mathsf{dom}(\mathsf{f}) \subseteq \mathrm{A}$ and $\mathsf{rng}(\mathsf{f}) \subseteq \mathrm{B}$ we indicate the domain and range of f, respectively. In addition, $\uplus$ denotes the *completion operator* that, taken f and another partial function $\mathsf{g} : \mathrm{A} \rightharpoonup \mathrm{B}$, returns the partial function $\mathsf{f} \uplus \mathsf{g} \triangleq (\mathsf{f} \setminus \mathsf{dom}(\mathsf{g})) \cup \mathsf{g} : \mathrm{A} \rightharpoonup \mathrm{B}$, which is equal to g on its domain and assumes the same values of f on the remaining part of A.

A two-player turn-based *arena* is a tuple $\mathcal{A} = \langle \mathrm{Ps}^{0}, \mathrm{Ps}^{1}, Mv \rangle$, with $\mathrm{Ps}^{0} \cap \mathrm{Ps}^{1} = \emptyset$ and $\mathrm{Ps} \triangleq \mathrm{Ps}^{0} \cup \mathrm{Ps}^{1}$, such that $\langle \mathrm{Ps}, Mv \rangle$ is a finite directed graph without sinks. Ps^{0} (*resp.*, Ps^{1}) is the set of positions of player 0 (*resp.*, 1) and $Mv \subseteq \mathrm{Ps} \times \mathrm{Ps}$ is a left-total relation describing all possible moves. A *path* in $\mathrm{V} \subseteq \mathrm{Ps}$ is a finite or infinite sequence $\pi \in \mathrm{Pth}(\mathrm{V})$ of positions in V compatible with the move relation, *i.e.*, $(\pi_i, \pi_{i+1}) \in Mv$, for all $i \in [0, |\pi| - 1[$. For a finite path π, with $\mathsf{lst}(\pi)$ we denote the last position

of π. A positional *strategy* for player $\alpha \in \{0,1\}$ on $\mathrm{V} \subseteq \mathrm{Ps}$ is a partial function $\sigma_\alpha \in \mathrm{Str}^\alpha(\mathrm{V}) \subseteq (\mathrm{V} \cap \mathrm{Ps}^\alpha) \rightharpoonup \mathrm{V}$, mapping each α-position $v \in \mathsf{dom}(\sigma_\alpha)$ to position $\sigma_\alpha(v)$ compatible with the move relation, *i.e.*, $(v, \sigma_\alpha(v)) \in Mv$. With $\mathrm{Str}^\alpha(\mathrm{V})$ we denote the set of all α-strategies on V. A *play* in $\mathrm{V} \subseteq \mathrm{Ps}$ from a position $v \in \mathrm{V}$ *w.r.t.* a pair of strategies $(\sigma_0, \sigma_1) \in \mathrm{Str}^0(\mathrm{V}) \times \mathrm{Str}^1(\mathrm{V})$, called $((\sigma_0, \sigma_1), v)$-*play*, is a path $\pi \in \mathrm{Pth}(\mathrm{V})$ such that $\pi_0 = v$ and, for all $i \in [0, |\pi| - 1[$, if $\pi_i \in \mathrm{Ps}^0$ then $\pi_{i+1} = \sigma^0(\pi_i)$ else $\pi_{i+1} = \sigma^1(\pi_i)$. The *play function* $\mathsf{play} : (\mathrm{Str}^0(\mathrm{V}) \times \mathrm{Str}^1(\mathrm{V})) \times \mathrm{V} \to \mathrm{Pth}(\mathrm{V})$ returns, for each position $v \in \mathrm{V}$ and pair of strategies $(\sigma_0, \sigma_1) \in \mathrm{Str}^0(\mathrm{V}) \times \mathrm{Str}^1(\mathrm{V})$, the maximal $((\sigma_0, \sigma_1), v)$-play $\mathsf{play}((\sigma^0, \sigma^1), v)$.

A *parity game* is a tuple $\mathcal{D} = \langle \mathcal{A}, \mathrm{Pr}, \mathsf{pr} \rangle \in \mathcal{P}$, where $\mathcal{A}$ is an arena, $\mathrm{Pr} \subset \mathbb{N}$ is a finite set of priorities, and $\mathsf{pr} : \mathrm{Ps} \to \mathrm{Pr}$ is a *priority function* assigning a priority to each position. The priority function can be naturally extended to games and paths as follows: $\mathsf{pr}(\mathcal{D}) \triangleq \mathsf{max}_{v \in \mathrm{Ps}}\, \mathsf{pr}(v)$; for a path $\pi \in \mathrm{Pth}$, we set $\mathsf{pr}(\pi) \triangleq \mathsf{max}_{i \in [0,|\pi|[}\, \mathsf{pr}(\pi_i)$, if π is finite, and $\mathsf{pr}(\pi) \triangleq \limsup_{i \in \mathbb{N}} \mathsf{pr}(\pi_i)$, otherwise. A set of positions $\mathrm{V} \subseteq \mathrm{Ps}$ is an α-*dominion*, with $\alpha \in \{0,1\}$, if there exists an α-strategy $\sigma_\alpha \in \mathrm{Str}^\alpha(\mathrm{V})$ such that, for all $\overline{\alpha}$-strategies $\sigma_{\overline{\alpha}} \in \mathrm{Str}^{\overline{\alpha}}(\mathrm{V})$ and positions $v \in \mathrm{V}$, the induced play $\pi = \mathsf{play}((\sigma_0, \sigma_1), v)$ is infinite and $\mathsf{pr}(\pi) \equiv_2 \alpha$. In other words, σ_α only induces on V infinite plays whose maximal priority visited infinitely often has parity α. By $\mathcal{D} \setminus \mathrm{V}$ we denote the maximal subgame of $\mathcal{D}$ with set of positions Ps' contained in $\mathrm{Ps} \setminus \mathrm{V}$ and move relation Mv' equal to the restriction of Mv to Ps'.

The α-predecessor of V, in symbols $\mathsf{pre}^\alpha(\mathrm{V}) \triangleq \{v \in \mathrm{Ps}^\alpha : Mv(v) \cap \mathrm{V} \neq \emptyset\} \cup \{v \in \mathrm{Ps}^{\overline{\alpha}} : Mv(v) \subseteq \mathrm{V}\}$, collects the positions from which player α can force the game to reach some position in V with a single move. The α-attractor $\mathsf{atr}^\alpha(\mathrm{V})$ generalities the notion of α-predecessor $\mathsf{pre}^\alpha(\mathrm{V})$ to an arbitrary number of moves, and corresponds to the least fix-point of that operator. When $\mathrm{V} = \mathsf{atr}^\alpha(\mathrm{V})$, we say that V is α-maximal. Intuitively, V is α-maximal if player α cannot force any position outside V to enter the set. For such a V, the set of positions of the subgame $\mathcal{D} \setminus \mathrm{V}$ is precisely $\mathrm{Ps} \setminus \mathrm{V}$. Finally, the set $\mathsf{esc}^\alpha(\mathrm{V}) \triangleq \mathsf{pre}^\alpha(\mathrm{Ps} \setminus \mathrm{V}) \cap \mathrm{V}$, called the α-*escape* of V, contains the positions in V from which α can leave V in one move. The dual notion of α-*interior*, defined as $\mathsf{int}^\alpha(\mathrm{V}) \triangleq (\mathrm{V} \cap \mathrm{Ps}^\alpha) \setminus \mathsf{esc}^\alpha(\mathrm{V})$, contains, instead, the α-positions from which α cannot escape with a single move. Observe that all the operators and sets described above actually depend on the specific game $\mathcal{D}$ they are applied to. In the rest of the paper, we shall only add $\mathcal{D}$ as subscript of an operator, *e.g.*, $\mathsf{esc}^\alpha_{\mathcal{D}}(\mathrm{V})$, when the game is not clear from the context.

3. The Priority Promotion Approach

The priority promotion approach proposed in [39] attacks the problem of solving a parity game $\mathcal{D}$ by computing one of its dominions D, for some player $\alpha \in \{0,1\}$, at a time. Indeed, once the α-attractor $\mathrm{D}^\star$ of D is removed from $\mathcal{D}$, the smaller game $\mathcal{D} \setminus \mathrm{D}^\star$ is obtained, whose positions are winning for one player iff they are winning for the same player in the original game. This allows for decomposing the problem of solving a parity game into iteratively finding its dominions [30].

In order to solve the dominion problem, the idea is to start from a much weaker notion than that of dominion, called *quasi dominion*. Intuitively, a quasi α-dominion is

a set of positions on which player α has a strategy whose induced plays either remain inside the set forever and are winning for α or can exit from it passing through a specific set of escape positions.

Definition 3.1 (Quasi Dominion)**.** *Let $\partial \in \mathcal{P}$ be a game and $\alpha \in \{0,1\}$ a player. A non-empty set of positions $\mathrm{Q} \subseteq \mathrm{Ps}$ is a* quasi α-dominion *in ∂ if there exists an α-strategy $\sigma_\alpha \in \mathrm{Str}^\alpha(\mathrm{Q})$ such that, for all $\overline{\alpha}$-strategies $\sigma_{\overline{\alpha}} \in \mathrm{Str}^{\overline{\alpha}}(\mathrm{Q})$, with $\mathsf{int}^{\overline{\alpha}}(\mathrm{Q}) \subseteq \mathsf{dom}(\sigma_{\overline{\alpha}})$, and positions $v \in \mathrm{Q}$, the induced play $\pi = \mathsf{play}((\sigma_0, \sigma_1), v)$ satisfies $\mathsf{pr}(\pi) \equiv_2 \alpha$, if π is infinite, and $\mathsf{lst}(\pi) \in \mathsf{esc}^{\overline{\alpha}}(\mathrm{Q})$, otherwise.*

Observe that, if all the induced plays remain in the set Q forever, this is actually an α-dominion and, therefore, a subset of the winning region Wn_α of α. In this case, the escape set of Q is empty, *i.e.*, $\mathsf{esc}^{\overline{\alpha}}(\mathrm{Q}) = \emptyset$, and Q is said to be α-*closed*. In general, however, a quasi α-dominion Q that is not an α-dominion, *i.e.*, such that $\mathsf{esc}^{\overline{\alpha}}(\mathrm{Q}) \neq \emptyset$, need not to be a subset of Wn_α and it is called α-*open*. Indeed, in this case, some induced play may not satisfy the winning condition for that player once exited from Q, by visiting a cycle containing a position with maximal priority of parity $\overline{\alpha}$. The set of pairs $(\mathrm{Q}, \alpha) \in 2^{\mathrm{Ps}} \times \{0,1\}$, where Q is a quasi α-dominion, is denoted by QD, and is partitioned into the sets QD^- and QD^+ of open and closed quasi α-dominion pairs, respectively.

Algorithm 1: Parity Game Solver.

signature $\mathsf{sol}_\Gamma : \mathcal{P} \to_\partial (2^{\mathrm{Ps}_\partial} \times 2^{\mathrm{Ps}_\partial})$
function $\mathsf{sol}_\Gamma(\partial)$
1 **if** $\mathrm{Ps}_\partial = \emptyset$ **then**
2 **return** $(\emptyset, \emptyset)$
else
3 $(\mathrm{R}, \alpha) \leftarrow \mathsf{search}_\Gamma(\partial)$
4 $\mathrm{R}^\star \leftarrow \mathsf{atr}^\alpha_\partial(\mathrm{R})$
5 $(\mathrm{W}_0', \mathrm{W}_1') \leftarrow \mathsf{sol}_\Gamma(\partial \setminus \mathrm{R}^\star)$
6 $(\mathrm{W}_\alpha, \mathrm{W}_{\overline{\alpha}}) \leftarrow (\mathrm{W}_\alpha' \cup \mathrm{R}^\star, \mathrm{W}_{\overline{\alpha}}')$
7 **return** $(\mathrm{W}_0, \mathrm{W}_1)$

Algorithm 2: The Searcher.

signature $\mathsf{search}_\Gamma : \mathcal{P} \to_\partial \mathrm{Rg}^+_\partial$
function $\mathsf{search}_\Gamma(\partial)$
1 **return** $\mathsf{search}_{\Gamma(\partial)}(\top_{\Gamma(\partial)})$

signature $\mathsf{search}_\mathcal{D} : \mathrm{S}_\mathcal{D} \to \mathrm{QD}^+_{\partial_\mathcal{D}}$
function $\mathsf{search}_\mathcal{D}(s)$
1 $(\mathrm{Q}, \alpha) \leftarrow \Re_\mathcal{D}(s)$
2 **if** $(\mathrm{Q}, \alpha) \in \mathrm{QD}^+_{\partial_\mathcal{D}}$ **then**
3 **return** (Q, α)
else
4 **return** $\mathsf{search}_\mathcal{D}(s \downarrow_\mathcal{D} (\mathrm{Q}, \alpha))$

The priority promotion algorithm explores a partial order $\mathcal{D}$, whose elements, called *states*, record information about the open quasi dominions computed along the way. Different partial orders can be associated to the same game. Therefore, we shall implicitly assume a function Γ mapping every game ∂ to a partial order $\mathcal{D} = \Gamma(\partial)$, where $\partial_\mathcal{D} \triangleq \partial$. The initial state of the search is the top element of the order, where the quasi dominions are initialized to the sets of positions with the same priority. At each step, a new quasi dominion is extracted from the current state, by means of a *query* operator $\Re$, and used to compute a successor state, by means of a *successor* operator $\downarrow$, if the quasi dominion is open. If, on the other hand, it is closed, the search is over. Algorithm 2 implements the dominion search procedure $\mathsf{search}_\mathcal{D}$. A *compatibility relation* $\succ$ connects the query and the successor operators. The relation holds between

states of the partial order and the quasi dominions that can be extracted by the query operator. Such a relation defines the domain of the successor operator. The partial order together with the query and successor operators and the compatibility relation form what is called a *dominion space*.

Definition 3.2 (Dominion Space)**.** *A* dominion space *for a game* $\partial \in \mathcal{P}$ *is a tuple* $\mathcal{D} \triangleq \langle \partial, \mathcal{S}, \succ, \Re, \downarrow \rangle$*, where* (1) $\mathcal{S} \triangleq \langle \mathrm{S}, \top, \prec \rangle$ *is a well-founded partial order* w.r.t. $\prec \subset \mathrm{S} \times \mathrm{S}$ *with distinguished element* $\top \in \mathrm{S}$*,* (2) $\succ \subseteq \mathrm{S} \times \mathrm{QD}^{-}$ *is the* compatibility relation*,* (3) $\Re : \mathrm{S} \to \mathrm{QD}$ *is the* query operator *mapping each element* $s \in \mathrm{S}$ *to a quasi dominion pair* $(\mathrm{Q}, \alpha) \triangleq \Re(s) \in \mathrm{QD}$ *such that, if* $(\mathrm{Q}, \alpha) \in \mathrm{QD}^{-}$ *then* $s \succ (\mathrm{Q}, \alpha)$*, and* (4) $\downarrow : \succ \to \mathrm{S}$ *is the* successor operator *mapping each compatible pair* $(s, (\mathrm{Q}, \alpha)) \in \succ$ *to the element* $s^{\star} \triangleq s\downarrow(\mathrm{Q}, \alpha) \in \mathrm{S}$ *with* $s^{\star} \prec s$*.*

The notion of dominion space is quite general and can be instantiated in different ways, by providing specific query and successor operators. In [39], indeed, it is shown that the search procedure $\mathsf{search}_{\mathcal{D}}$ is sound and complete on any dominion space $\mathcal{D}$. In addition, its time complexity is linear in the *execution depth* of the dominion space, namely the length of the longest chain in the underlying partial order compatible with the successor operator, while its space complexity is only logarithmic in the space *size*, since only one state at the time needs to be maintained. A specific instantiation of dominion space, called PP *dominion space*, is the one proposed and studied in [39]. Here we propose a different one, starting from a slight optimization, called PP+, of that original version.

PP+ *Dominion Space.* In order to instantiate a dominion space, we need to define a suitable query function to compute quasi dominions and a successor operator to ensure progress in the search for a closed dominion. The priority promotion algorithm proceeds as follows. The input game is processed in descending order of priority. At each step, a subgame of the entire game, obtained by removing the quasi dominions previously computed at higher priorities, is considered. At each priority of parity α, a quasi α-dominion Q is extracted by the query operator from the current subgame. If Q is closed in the entire game, the search stops and returns Q as result. Otherwise, a successor state in the underlying partial order is computed by the successor operator, depending on whether Q is open in the current subgame or not. In the first case, the quasi α-dominion is removed from the current subgame and the search restarts on the new subgame that can only contain positions with lower priorities. In the second case, Q is merged together with some previously computed quasi α-dominion with higher priority. Being a dominion space well-ordered, the search is guaranteed to eventually terminate and return a closed quasi dominion. The procedure requires the solution of two crucial problems: *(a) extracting a quasi dominion* from a subgame and *(b) merging together two quasi* α*-dominions* to obtain a bigger, possibly closed, quasi α-dominion.

The solution of the first problem relies on the definition of a specific class of quasi dominions, called *regions*. An α-region R of a game ∂ is a special form of a quasi α-dominion of ∂ with the additional requirement that all the escape positions in $\mathsf{esc}^{\overline{\alpha}}(\mathrm{R})$ have the maximal priority $p \triangleq \mathsf{pr}(\partial) \equiv_2 \alpha$ in ∂. In this case, we say that the α-region R has priority p. As a consequence, if the opponent $\overline{\alpha}$ can escape from the α-region R, he must visit a position with the highest priority in it, which is of parity α.

Definition 3.3 (Region)**.** *A quasi α-dominion* R *is an α-region in* $\mathcal{D}$ *if* $\mathsf{pr}(\mathcal{D}) \equiv_2 \alpha$ *and all the positions in* $\mathsf{esc}^{\overline{\alpha}}(\mathrm{R})$ *have priority* $\mathsf{pr}(\mathcal{D})$*,* i.e. $\mathsf{esc}^{\overline{\alpha}}(\mathrm{R}) \subseteq \mathsf{pr}^{-1}(\mathsf{pr}(\mathcal{D}))$.

It is important to observe that, in any parity game, an α-region always exists, for some $\alpha \in \{0, 1\}$. In particular, the set of positions of maximal priority in the game always forms an α-region, with α equal to the parity of that maximal priority. In addition, the α-attractor of an α-region is always an α-maximal α-region. A closed α-region in a game is clearly an α-dominion in that game. These observations give us an easy and efficient way to extract a quasi dominion from every subgame: collect the α-attractor of the positions with maximal priority p in the subgame, where $p \equiv_2 \alpha$, and assign p as priority of the resulting region R. This priority, called *measure* of R, intuitively corresponds to an under-approximation of the best priority player α can force the opponent $\overline{\alpha}$ to visit along any play exiting from R.

Proposition 3.1 (Region Extension [39])**.** *Let $\mathcal{D} \in \mathcal{P}$ be a game and $\mathrm{R} \subseteq \mathrm{Ps}$ an α-region in $\mathcal{D}$. Then, $\mathrm{R}^\star \triangleq \mathsf{atr}^\alpha(\mathrm{R})$ is an α-maximal α-region in $\mathcal{D}$.*

A solution to the second problem, the merging operation, is obtained as follows. Given an α-region R in some game $\mathcal{D}$ and an α-dominion D in a subgame of $\mathcal{D}$ that does not contain R itself, the two sets are merged together, if the only moves exiting from $\overline{\alpha}$-positions of D in the entire game lead to higher priority α-regions and R has the lowest priority among them. The priority of R is called the *best escape priority* of D for $\overline{\alpha}$. The correctness of this merging operation is established by the following proposition.

Proposition 3.2 (Region Merging [39])**.** *Let $\mathcal{D} \in \mathcal{P}$ be a game, $\mathrm{R} \subseteq \mathrm{Ps}$ an α-region, and $\mathrm{D} \subseteq \mathrm{Ps}_{\mathcal{D} \setminus \mathrm{R}}$ an α-dominion in the subgame $\mathcal{D} \setminus \mathrm{R}$. Then, $\mathrm{R}^\star \triangleq \mathrm{R} \cup \mathrm{D}$ is an α-region in $\mathcal{D}$. Moreover, if both R and D are α-maximal in $\mathcal{D}$ and $\mathcal{D} \setminus \mathrm{R}$, respectively, then $\mathrm{R}^\star$ is α-maximal in $\mathcal{D}$ as well.*

The merging operation is implemented by promoting all the positions of α-dominion D to the measure of R, thus improving the measure of D. For this reason, it is called a *priority promotion*. In [39] it is shown that, after a promotion to some measure p, the regions with measure lower than p might need to be destroyed, by resetting all the contained positions to their original priority. This necessity derives from the fact that the new promoted region may attract positions from lower ones, thereby potentially invalidating their status as regions. Indeed, in some cases, the player that wins by remaining in the region may even change from α to $\overline{\alpha}$. As a consequence, the reset operation is, in general, unavoidable. The original priority promotion algorithm applies the reset operation to all the lower priority regions. However, the following property ensures that this can be limited to the regions belonging to the opponent player only.

Proposition 3.3 (Region Splitting)**.** *Let $\mathcal{D}^\star \in \mathcal{P}$ be a game and $\mathrm{R}^\star \subseteq \mathrm{Ps}_{\mathcal{D}^\star}$ an α-maximal α-region in $\mathcal{D}^\star$. For any subgame $\mathcal{D}$ of $\mathcal{D}^\star$ and α-region $\mathrm{R} \subseteq \mathrm{Ps}$ in $\mathcal{D}$, if $\mathrm{R}^\sharp \triangleq \mathrm{R} \setminus \mathrm{R}^\star \neq \emptyset$, then $\mathrm{R}^\sharp$ is an α-region in $\mathcal{D} \setminus \mathrm{R}^\star$.*

Proof. Let us first prove that $\mathsf{esc}^{\overline{\alpha}}_{\mathcal{D} \setminus \mathrm{R}^\star}(\mathrm{R}^\sharp) \subseteq \mathsf{esc}^{\overline{\alpha}}(\mathrm{R})$. Assume, by contradiction, that there exists a position $v \in \mathsf{esc}^{\overline{\alpha}}_{\mathcal{D} \setminus \mathrm{R}^\star}(\mathrm{R}^\sharp)$ which does not belong to $\mathsf{esc}^{\overline{\alpha}}(\mathrm{R})$. Let us consider the case where v is an α-position of $\mathrm{R}^\sharp$ first. Then, one of its moves leads to

R in ∂. If such move leads to $\mathrm{R}^\sharp$, then it does so also in the subgame $\partial \setminus \mathrm{R}^\star$, and v cannot belong to the escape set of $\mathrm{R}^\sharp$. If, on the other hand, it leads to $\mathrm{R}^\star$, then, due to the maximality of $\mathrm{R}^\star$, v must belong to $\mathrm{R}^\star$ as well, contradicting the assumption that $v \in \mathsf{esc}^{\overline{\alpha}}_{\partial\setminus\mathrm{R}^\star}(\mathrm{R}^\sharp)$. Assume now that v is a $\overline{\alpha}$-position. Then, all of its moves lead to R in ∂. Clearly, all the moves from v that remain in the subgame $\partial \setminus \mathrm{R}^\star$, if any, must lead to $\mathrm{R}^\sharp$. Hence, v cannot be an escaping position of $\mathrm{R}^\sharp$ in $\partial \setminus \mathrm{R}^\star$ in this case either. As a consequence, all the positions in $\mathsf{esc}^{\overline{\alpha}}_{\partial\setminus\mathrm{R}^\star}(\mathrm{R}^\sharp)$ have maximal priority in ∂ and, *a fortiori*, in $\partial \setminus \mathrm{R}^\star$ as well.

Let us now prove that $\mathrm{R}^\sharp$ is a quasi α-dominion. Let $\sigma \in \mathrm{Str}^\alpha(\mathrm{R})$ be the witness α-strategy for R as in Definition 3.1. Hence, regardless of the $\overline{\alpha}$-strategy used by the opponent, each play induced by σ, either is infinite, hence remains in R forever, and is winning for α, or is finite and ends in some position of the escape set $\mathsf{esc}^{\overline{\alpha}}(\mathrm{R})$. Consider now the restriction $\sigma' \triangleq \sigma{\restriction}\mathrm{R}^\sharp$ of σ to the α-positions of $\mathrm{R}^\sharp$. Let π be a play induced by σ'. If π remains in $\mathrm{R}^\sharp \subseteq \mathrm{R}$ forever, then it is clearly winning for α, as that play is also compatible with σ. If, on the other hand, it is finite, then it must end in some position $v \in \mathrm{R}^\sharp$. If v is a $\overline{\alpha}$-position, then it has a move leaving $\mathrm{R}^\sharp$ and, therefore, must belong to the escape $\mathsf{esc}^{\overline{\alpha}}_{\partial\setminus\mathrm{R}^\star}(\mathrm{R}^\sharp)$. Assume v is an α-position. If v is not in the domain of σ', then it is not in the domain of σ either. Hence, it must belong to $\mathsf{esc}^{\overline{\alpha}}_{\partial}(\mathrm{R})$. Being $\mathsf{esc}^{\overline{\alpha}}_{\partial\setminus\mathrm{R}^\star}(\mathrm{R}^\sharp) \subseteq \mathsf{esc}^{\overline{\alpha}}_{\partial}(\mathrm{R})$ and $v \in \mathrm{R}^\sharp$, we can conclude that $v \in \mathsf{esc}^{\overline{\alpha}}_{\partial\setminus\mathrm{R}^\star}(\mathrm{R}^\sharp)$. Finally, assume, by contradiction, that v is in the domain of σ'. Then, $\sigma'(v) = \sigma(v) \in \mathrm{R}$. Since π ends in v, however, $\sigma'(v) \notin \mathrm{R}^\sharp$. But then, v would be an α-position in R with a move leading to $\mathrm{R}^\star$, contradicting the α-maximality of $\mathrm{R}^\star$. We can, thus, conclude that v must belong to $\mathsf{esc}^{\overline{\alpha}}_{\partial\setminus\mathrm{R}^\star}(\mathrm{R}^\sharp)$. □

This proposition, together with the observation that $\overline{\alpha}$-regions that can be extracted from the corresponding subgames cannot attract positions contained in any retained α-region, allows for preserving all the lower α-regions computed so far.

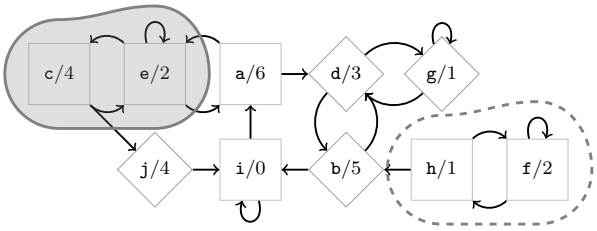

Figure 1: Running example.

	1	2	3	4	5	6
6	$\mathtt{a}\downarrow$	···	···	···	···	$\mathtt{a},\mathtt{b},\mathtt{d},\mathtt{g},\mathtt{i},\mathtt{j}\downarrow$
5	$\mathtt{b},\mathtt{f},\mathtt{h}\downarrow$	···	···	$\mathtt{b},\mathtt{d},\mathtt{f},\mathtt{g},\mathtt{h}\downarrow$	···	
4	$\mathtt{c},\mathtt{j}\downarrow$	$\mathtt{c},\mathtt{e},\mathtt{j}\downarrow$	···	$\mathtt{c},\mathtt{j}\downarrow$	$\mathtt{c},\mathtt{e},\mathtt{j}\downarrow$	$\mathtt{c},\mathtt{e}\uparrow_6$
3	$\mathtt{d}\downarrow$	$\mathtt{d}\downarrow$	$\mathtt{d},\mathtt{g}\uparrow_5$			
2	$\mathtt{e}\uparrow_4$			$\mathtt{e}\uparrow_4$		
1		$\mathtt{g}\uparrow_3$				
0					$\mathtt{i}\uparrow_6$	

Table 1: PP+ simulation.

Example 3.1. *To exemplify the idea, Table 1 shows a simulation of the resulting procedure on the parity game of Figure 1, where diamond shaped positions belong to player* 0 *and square shaped ones to its opponent* 1*. Player* 0 *wins the entire game, hence the* 0*-region containing all the positions is a* 0*-dominion in this case. Each cell of the table contains a computed region. A downward arrow denotes a region that is open in the subgame where it is computed, while an upward arrow means that the region gets to be promoted to the priority in the subscript. The index of each row corresponds to the measure of the region. Following the idea sketched above, the first region obtained is the single-position* 0*-region* $\{\mathtt{a}\}$ *of measure 6, which is open because of the two moves leading to* `d` *and* `e`*. The open* 1*-region* $\{\mathtt{b},\mathtt{f},\mathtt{h}\}$ *of measure 5 is, then, formed by attracting both* `f` *and* `h` *to* `b`*, which is open in the subgame where* $\{\mathtt{a}\}$ *is removed. Similarly, the* 0*-region* $\{\mathtt{c},\mathtt{j}\}$ *of measure* 4 *and the* 1*-region* $\{\mathtt{d}\}$ *of measure* 3 *are open, once removed* $\{\mathtt{a},\mathtt{b},\mathtt{f},\mathtt{h}\}$ *and* $\{\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{f},\mathtt{h}\}$*, respectively, from the game. At priority* 2*, the* 0*-region* $\{\mathtt{e}\}$ *is closed in the corresponding subgame. However, it is not closed in the whole game, because of the move leading to* `c`*,* i.e.*, to the region of measure* 4*. Proposition 3.2 can now be applied and a promotion of* $\{\mathtt{e}\}$ *to* 4 *is performed, resulting in the new* 0*-region* $\{\mathtt{c},\mathtt{e},\mathtt{j}\}$ *that resets* 1*-region* $\{\mathtt{d}\}$*. The search resumes at the corresponding priority and, after computing the extension of such a region via the attractor, we obtain that it is still open in the corresponding subgame. Consequently, the* 1*-region* $\{\mathtt{d}\}$ *of measure* 3 *is recomputed and, then, priority* 1 *is processed to build the* 1*-region* $\{\mathtt{g}\}$*. The latter is closed in the associated subgame, but not in the original game, because of a move leading to position* `d`*. Hence, another promotion is performed, leading to the closed region of measure 3 at Column 3, which in turn triggers a promotion to* 5*. When the promotion of* 0*-region* $\{\mathtt{i}\}$ *to priority* 6 *is performed, however,* 0*-region* $\{\mathtt{c},\mathtt{e},\mathtt{j}\}$ *of measure* 4 *is not reset. This leads directly to the configuration in Column 6, after the maximization of* 0*-region* 6*, which attracts* `b`*,* `d`*,* `g`*, and* `j`*. Notice that, as prescribed by Proposition 3.3, the set* $\{\mathtt{c},\mathtt{e}\}$*, highlighted by the gray area, is still a* 0*-region. On the other hand, the set* $\{\mathtt{f},\mathtt{h}\}$*, highlighted by the dashed line and originally included in* 1*-region* $\{\mathtt{b},\mathtt{d},\mathtt{f},\mathtt{g},\mathtt{h}\}$ *of priority* 5*, needs to be reset, since it is not a* 1*-region any more. It is, actually, an open* 0*-region instead. Now,* 0*-region* 4 *is closed in its subgame and it is promoted to* 6*. As result of this promotion, we obtain the closed* 0*-region* $\{\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{g},\mathtt{i},\mathtt{j}\}$*, which is a dominion for player* 0*.*

We can now provide the formal account of the PP+ dominion space. We shall denote with Rg the set of region pairs in $\mathcal{D}$ and with Rg^- and Rg^+ the sets of open and closed region pairs, respectively.

Similarly to the priority promotion algorithm, during the search for a dominion, the computed regions, together with their current measure, are kept track of by means of an auxiliary priority function $\mathsf{r} \in \Delta \triangleq \mathrm{Ps} \to \mathrm{Pr}$, called *region function*. Given a priority $p \in \mathrm{Pr}$, we denote by $\mathsf{r}^{(\geq p)}$ (*resp.*, $\mathsf{r}^{(>p)}$, $\mathsf{r}^{(<p)}$, and $\mathsf{r}^{(\equiv_2 p)}$) the function obtained by restricting the domain of r to the positions with measure greater than or equal to p (*resp.*, greater than, lower than, and congruent modulo 2 to p). Formally, $\mathsf{r}^{(\sim p)} \triangleq \mathsf{r}{\restriction}\{v \in \mathrm{Ps} : \mathsf{r}(v) \sim p\}$, for $\sim \in \{\geq, >, <, \equiv_2\}$. By $\mathcal{D}_{\mathsf{r}}^{\leq p} \triangleq \mathcal{D} \setminus \mathsf{dom}\big(\mathsf{r}^{(>p)}\big)$, we denote the largest subgame obtained by removing from $\mathcal{D}$ all the positions in the domain

of $\mathsf{r}^{(>p)}$. The *maximization* of a priority function $\mathsf{r} \in \Delta$ is the unique priority function $\mathsf{m} \in \Delta$ such that $\mathsf{m}^{-1}(q) = \mathsf{atr}^{\alpha}_{\partial^{\leq q}_{\mathsf{m}}}\left(\mathsf{r}^{-1}(q) \cap \mathrm{Ps}_{\partial^{\leq q}_{\mathsf{m}}}\right)$, for all priorities $q \in \mathsf{rng}(\mathsf{r})$ with $\alpha \triangleq q \bmod 2$. In addition, we say that r is *maximal* above $p \in \mathrm{Pr}$ iff $\mathsf{r}^{(>p)} = \mathsf{m}^{(>p)}$.

As opposed to the PP approach, where a promotion to $p \equiv_2 \alpha$ resets all the regions lower than p, here we need to take into account the fact that the regions of the opponent $\overline{\alpha}$ are reset, while the ones of player α are retained. In particular, we need to ensure that, as the search proceeds from p downward to any priority $q < p$, the maximization of the regions contained at priorities higher than q can never make the region recorded in r at q invalid. To this end, we consider only priority functions r that satisfy the requirement that, at all priorities, they contain regions *w.r.t.* the subgames induced by their maximizations m. Formally, $\mathsf{r} \in \mathbb{R} \subseteq \Delta$ is a *region function* iff, for all priorities $q \in \mathsf{rng}(\mathsf{m})$ with $\alpha \triangleq q \bmod 2$, it holds that $\mathsf{r}^{-1}(q) \cap \mathrm{Ps}_{\partial^{\leq q}_{\mathsf{m}}}$ is an α-region in the subgame $\partial^{\leq q}_{\mathsf{m}}$, where m is the maximization of r.

The status of the search of a dominion is encoded by the notion of *state* s of the dominion space, which contains the current region function r and the current priority p reached by the search in ∂. Initially, r coincides with the priority function pr of the entire game ∂, while p is set to the maximal priority $\mathsf{pr}(\partial)$ available in the game. To each of such states $s \triangleq (\mathsf{r}, p)$, we then associate the *subgame at* s defined as $\partial_s \triangleq \partial^{\leq p}_{\mathsf{r}}$, representing the portion of the original game that still has to be processed.

The following state space specifies the configurations in which the PP+ procedure can reside and the relative order that the successor function must satisfy.

Definition 3.4 (State Space for PP+)**.** *A* PP+ state space *is a tuple* $\mathcal{S} \triangleq \langle \mathrm{S}, \top, \prec \rangle$*, where:*

1. $\mathrm{S} \subseteq \mathbb{R} \times \mathrm{Pr}$ *is the set of all pairs* $s \triangleq (\mathsf{r}, p)$*, called* states*, composed of a region function* $\mathsf{r} \in \mathbb{R}$ *and a priority* $p \in \mathrm{Pr}$ *such that* (a) r *is maximal above* p *and* (b) $p \in \mathsf{rng}(\mathsf{r})$*;*
2. $\top \triangleq (\mathsf{pr}, \mathsf{pr}(\partial))$*;*
3. *two states* $s_1 \triangleq (\mathsf{r}_1, p_1), s_2 \triangleq (\mathsf{r}_2, p_2) \in \mathrm{S}$ *satisfy* $s_1 \prec s_2$ *iff either* (a) ${\mathsf{r}_1}^{(>q)} = {\mathsf{r}_2}^{(>q)}$ *and* $\mathsf{r}_2^{-1}(q) \subset \mathsf{r}_1^{-1}(q)$*, for some priority* $q \in \mathsf{rng}(\mathsf{r}_1)$ *with* $q \geq p_1$*, or* (b) *both* $\mathsf{r}_1 = \mathsf{r}_2$ *and* $p_1 < p_2$ *hold.*

Condition 1 requires that every region $\mathsf{r}^{-1}(q)$ with measure $q > p$ be α-maximal, where $\alpha = q \bmod 2$. This implies that $\mathsf{r}^{-1}(q) \subseteq \mathrm{Ps}_{\partial^{\leq q}_{\mathsf{r}}}$. Moreover, the current priority p of the state must be one of the measures recorded in r. In addition, Condition 2 specifies the initial state, while Condition 3 defines the ordering relation among states, which the successor operation has to comply with. It asserts that a state s_1 is strictly smaller than another state s_2 if either there is a region recorded in s_1 with some higher measure q that strictly contains the corresponding one in s_2 and all regions with measure grater than q are equal in the two states, or state s_1 is currently processing a lower priority than the one of s_2.

A region pair (R, α) is compatible with a state $s \triangleq (\mathsf{r}, p)$ if it is an α-region in the current subgame ∂_s. Moreover, if such a region is α-open in that game, it has to be α-maximal and needs to necessarily contain the current region $\mathsf{r}^{-1}(p)$ of priority p in r.

Definition 3.5 (Compatibility Relation)**.** *An open quasi dominion pair* $(\mathrm{R},\alpha) \in \mathrm{QD}^{-}$ *is* compatible *with a state* $s \triangleq (\mathsf{r},p) \in \mathrm{S}$*, in symbols* $s \succ (\mathrm{R},\alpha)$*, iff* (1) $(\mathrm{R},\alpha) \in \mathrm{Rg}^{-}_{\mathcal{D}_s}$ *and* (2) *if* R *is* α*-open in* $\mathcal{D}_s$ *then* $\mathrm{R} = \mathsf{atr}^{\alpha}_{\mathcal{D}_s}(\mathsf{r}^{-1}(p))$.

Algorithm 3 provides the implementation of the query function compatible with the priority-promotion mechanism. Line 1 simply computes the parity α of the priority to process in the state $s \triangleq (\mathsf{r},p)$. Line 2, instead, computes the attractor *w.r.t.* player α in subgame $\mathcal{D}_s$ of the region contained in r at the current priority p. The resulting set R is, according to Proposition 3.1, an α-maximal α-region of $\mathcal{D}_s$ containing $\mathsf{r}^{-1}(p)$.

Algorithm 3: Query Function.

```
   signature ℜ : S → 2^Ps × {0,1}
   function ℜ(s)
       let (r, p) = s in
1          α ← p mod 2
2          R ← atr^α_{D_s}(r^{-1}(p))
3      return (R, α)
```

The promotion operation is based on the notion of best escape priority mentioned above, namely the priority of the lowest α-region in r that has an incoming move coming from the α-region, closed in the current subgame, that needs to be promoted. This concept is formally defined as follows. Let $I \triangleq Mv \cap ((\mathrm{R} \cap \mathrm{Ps}^{\overline{\alpha}}) \times (\mathsf{dom}(\mathsf{r}) \backslash \mathrm{R}))$ be the *interface relation* between R and r, *i.e.*, the set of $\overline{\alpha}$-moves exiting from R and reaching some position within a region recorded in r. Then, $\mathsf{bep}^{\overline{\alpha}}(\mathrm{R},\mathsf{r})$ is set to the minimal measure of those regions that contain positions reachable by a move in I. Formally, $\mathsf{bep}^{\overline{\alpha}}(\mathrm{R},\mathsf{r}) \triangleq \mathsf{min}(\mathsf{rng}(\mathsf{r}{\restriction}\mathsf{rng}(I)))$. Such a value represents the best priority associated with an α-region contained in r and reachable by $\overline{\alpha}$ when escaping from R. Note that, if R is a closed α-region in $\mathcal{D}_s$, then $\mathsf{bep}^{\overline{\alpha}}(\mathrm{R},\mathsf{r})$ is necessarily of parity α and greater than the measure p of R. This property immediately follows from the maximality of r above p. Indeed, no move of an $\overline{\alpha}$-position can lead to a $\overline{\alpha}$-maximal $\overline{\alpha}$-region. For instance, for 0-region $\mathrm{R} = \{\mathsf{e}\}$ with measure 2 in Column 1 of Figure 1, we have that $I = \{(\mathsf{e},\mathsf{a}),(\mathsf{e},\mathsf{c})\}$ and $\mathsf{r}{\restriction}\mathsf{rng}(I) = \{(\mathsf{a},6),(\mathsf{c},4)\}$. Hence, $\mathsf{bep}^{1}(\mathrm{R},\mathsf{r}) = 4$.

Algorithm 4 reports the pseudo-code of the successor function, which differs from the one proposed in [39] only in Line 5, where Proposition 3.3 is applied. Given the current state s and a compatible region pair (R,α) open in the whole game as inputs, it produces a successor state $s^{\star} \triangleq (\mathsf{r}^{\star},p^{\star})$ in the dominion space. It first checks whether R is open also in the subgame $\mathcal{D}_s$ (Line 1). If this is the case, it assigns the measure p to region R and stores it in the new region function $\mathsf{r}^{\star}$ (Line 2). The new current priority $p^{\star}$ is, then, computed as the highest priority lower than p in $\mathsf{r}^{\star}$ (Line 3). If, on the other hand, R is closed in $\mathcal{D}_s$, a promotion, merging R with some other α-region contained in r, is required. The next priority $p^{\star}$ is set to the bep of R for player $\overline{\alpha}$ in the entire game $\mathcal{D}$ *w.r.t.* r (Line 4). Region R is, then, promoted to priority $p^{\star}$ and all and only the regions of the opponent $\overline{\alpha}$ with lower measure than $p^{\star}$ in the region function r

Algorithm 4: Successor Function.

```
   signature ↓ : ≻ → Δ × Pr
   function s↓(R, α)
       let (r, p) = s in
1          if (R, α) ∈ Rg^-_{D_s} then
2              r* ← r[R ↦ p]
3              p* ← max(rng(r*^(<p)))
           else
4              p* ← bep^ᾱ(R, r)
5              r* ← pr ⊎ r^((≥p*)∨(≡2p*))[R ↦ p*]
6      return (r*, p*)
```

are reset by means of the completion operator defined in Section 2 (Line 5).

The following theorem asserts that the PP+ state space, together with the same query function of PP and the successor function of Algorithm 4 is a dominion space.

Theorem 3.1 (PP+ Dominion Space)**.** *For a game ∂, the* PP+ structure $\mathcal{D} \triangleq \langle \partial, \mathcal{S}, \succ, \Re, \downarrow \rangle$*, where $\mathcal{S}$ is given in Definition 3.4, $\succ$ is the relation of Definition 3.5, and $\Re$ and $\downarrow$ are the functions computed by Algorithms 3 and 4 is a dominion space.*

The PP+ procedure does reduce, *w.r.t.* PP, the number of reset needed to solve a game and the exponential worst-case game presented in [39] does not work any more. However, a worst-case, which is a slight modification of the one for PP, does exist for this procedure as well.

Consider the game $\partial_h^{\text{PP+}}$ containing h chains of length 2 that converge into a single position of priority 0 with a self loop. The i-th chain has a head of priority $2(h+1)-i$ and a body composed of a single position with priority i and a self loop. An instance of this game with $h = 4$ is depicted in Figure 2. The labels of the positions correspond to the associated priorities. Intuitively, the execution depth of the PP+ dominion space for this game is exponential, since the consecutive promotion operations performed on each chain can simulate the increments of a partial form of binary counter, some of whose configurations are missing. As a result, the number of configurations of the counter follows a Fibonacci-like sequence of the form $F(h) = F(h-1)+F(h-2)+1$, with $F(0) = 1$ and $F(1) = 2$. It is interesting to note that this is the same recurrence equation of the number of minimal AVL trees of hight h. The search procedure on $\partial_4^{\text{PP+}}$ starts by building the following four open regions: the 1-region $\{9\}$, the 0-region $\{8\}$, the 1-region $\{7\}$, and 0-region $\{6\}$. This state represents the configuration of the counter, where all four digits are set to 0. The closed 0-region $\{4\}$ is then found and promoted to 6. Now, the counter is set to 0001. After that, the closed 1-region $\{3\}$ is computed that is promoted to 7. Due to the promotion to 7, the positions in the 0-region with priority 6 are reset to their original priority, as they belong to the opponent player. This releases the chain with head 6, which corresponds to the reset of the least significant digit of the counter caused by the increment of the second one, *i.e.*, the counter displays 0010. The search resumes at priority 6 and the 0-regions $\{6\}$ and $\{4\}$ are computed once again. A second promotion of $\{4\}$ to 6 is performed, resulting in the counter assuming value 0011. When the closed 0-region $\{2\}$ is promoted to 8, however, only the 1-region $\{7, 3\}$ is reset, leading to configuration 0101 of the counter. Hence, configuration 0100 is skipped. Similarly, when, the counter reaches configuration 0111 and 1-region $\{1\}$ is promoted to 9, the 0-regions $\{8, 2\}$ and $\{6, 4\}$ are reset, leaving 1-region $\{7, 3\}$ intact. This leads directly to configuration 1010 of the counter, skipping configurations 1000 and 1001.

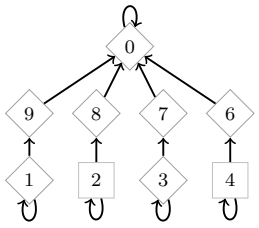

Figure 2: The $\partial_4^{\text{PP+}}$ game.

An estimate of the depth of the PP+ dominion space on the game $\partial_h^{\text{PP+}}$ is given by the following theorem.

Theorem 3.2 (Execution-Depth Lower Bound)**.** *For all $h \in \mathbb{N}$, there exists a* PP+ *dominion space $\mathcal{D}_h^{\text{PP+}}$ with $k = 2h+1$ positions and priorities, whose execution depth is* $\mathsf{Fib}(2(h+4))/\mathsf{Fib}(h+4) - (h+6) = \Theta\big(((1+\sqrt{5})/2)^{k/2}\big)$.

Proof. The game $\partial_h^{\text{PP+}}$ encodes a partial binary counter, where each one of the h chains, composed of two positions connected to position 0, represents a digit. The promotion operation of region i to region $2(h+1)-i$ corresponds to the increment of the i-th digit. As a consequence, the number of promotions for the PP+ algorithm is equal to the number of configurations of the counter, except for the initial one. To count them all on game $\partial_h^{\text{PP+}}$, consider the recursive function $\mathsf{P}(h)$ having base case $\mathsf{P}(0) = 1$ and inductive cases $\mathsf{P}(h) = 1 + \mathsf{P}(h-1) + \mathsf{G}(h-1)$, where the function G is described below. To be precise, this function computes a value that is exactly one more than the number of promotions. The correctness of the base case is trivial. Indeed, when h is equal to 0, there are no chains and, so, no promotions. We can now focus on the inductive case $h > 0$. Before reaching the promotion of region $\{1\}$ to region $\{2h+1\}$, *i.e.*, to set the most-significant digit, all others digits must be set. To do this, the amount of necessary promotions is equal to those required by the game with $h-1$ chains, *i.e.*, $\mathsf{P}(h-1)$. At this point, an additional promotion is counted when region $\{1\}$ is merged to region $\{2h+1\}$. As shown in the execution of the game depicted in Figure 2, this promotion resets the digits with index $i < h$ such that $i \not\equiv_2 h$. Therefore, $\mathsf{G}(h-1)$ counts exactly the promotions needed to set to 1 the digits reset after this promotion.

We now prove that the function G can be expressed by means of the following recursive definition: $\mathsf{G}(0) = 0$ and $\mathsf{G}(h) = \lceil h/2 \rceil + \sum_{i=0}^{\lceil h/2 \rceil - 1} \mathsf{G}(2i + 1 - h \bmod 2)$. When $h = 0$, we have that $\mathsf{G}(h) = 0$, since no chain is present. For the inductive case, when $h > 0$, if h is odd, the least significant digit is not set. Therefore, the algorithm performs a promotion to set it. After this digit is set, $\mathsf{G}(0) = 0$ promotions are required, since no lower chain is present. Then, the algorithm reaches the third least significant digit, since the second one is already set. In order to set it, another promotion is required. Once the third digit is set, the second one gets reset. So, to set all digits, $\mathsf{G}(2)$ promotions are then required. After this, the algorithm will reach the fifth significant digit and proceeds similarly until the $(h-1)$-th digit is processed. Summing up, $(h+1)/2 + \sum_{i=0}^{(h-1)/2} \mathsf{G}(2i)$ promotions are required in total. Similarly, when h is even, we obtain that $\mathsf{G}(h) = (h/2) + \sum_{i=0}^{(h/2)-1} \mathsf{G}(2i+1)$. Consequently $\mathsf{G}(h) = \lceil h/2 \rceil + \sum_{i=0}^{\lceil h/2 \rceil - 1} \mathsf{G}(2i + 1 - h \bmod 2)$, regardless of the parity of h.

Finally, it can be proved by induction that $\mathsf{G}(h) = \mathsf{Fib}(h+2) - 1$, and, consequently, $\mathsf{P}(h) = \mathsf{Fib}(h+3) - 1$. The proof of these claim requires the application of the equalities $\mathsf{Fib}(2h) = \sum_{i=0}^{h-1} \mathsf{Fib}(2i+1)$, and $\mathsf{Fib}(2h+1) - 1 = \sum_{i=1}^{h} \mathsf{Fib}(2i)$. Observe that the above definition of P is equivalent to the sequence for the number of configuration claimed for ∂_h, *i.e.* $\mathsf{P}(h) = \mathsf{P}(h-1) + \mathsf{P}(h-2) + 1$, since it is easy to see that $\mathsf{G}(h-1) = \mathsf{P}(h-2)$.

A similar reasoning can be exploited to count the number of queries executed by the the PP+ algorithm on game $\partial_h^{\text{PP+}}$. To do this, we use the function Q, having base case $\mathsf{Q}(0) = 1$ and inductive case $\mathsf{Q}(h) = 3 + \mathsf{Q}(h-1) + \mathsf{H}(h-1)$, where $\mathsf{H}(h)$ is defined in the following. The base case is trivial. When h is equal to 0, there are no chains and, so, the only region that can be returned by the query function is $\{0\}$. Let us now focus on the inductive case $h > 0$. The algorithm requires one query to create region $\{2h+1\}$, corresponding to the head of the h-th chain. Before reaching the construction of region $\{1\}$, all remaining digits have to be set to 1. To do this, the amount of necessary queries is equal to $\mathsf{Q}(h-1)$, where the query of region $\{0\}$ actually correspond to the query

of region $\{1\}$. At this point, one query is needed to promote the body $\{1\}$ to the head $\{2h+1\}$ of the h-th chain. This promotion resets the digits with index $i < h$ such that $i \not\equiv_2 h$. Therefore, $\mathsf{H}(h-1)$ queries are needed to set to 1 the digits reset after it. Finally, one query is required to create region $\{0\}$.

We now prove that the function H can be expressed by the following recursive definition: $\mathsf{H}(0) = 0$ and $\mathsf{H}(h) = h + 2\lceil h/2 \rceil + \sum_{i=0}^{\lceil h/2 \rceil - 1} \mathsf{H}(2i + 1 - h \bmod 2)$. Indeed, $\mathsf{H}(h) = 0$, when $h = 0$, since no chain is present. For the inductive case, when $h > 0$, the function requires h queries to create the regions corresponding to the heads of the chains before reaching the first region to promote. Also, it requires two queries for each of the $\lceil h/2 \rceil$ promotions that reset the chains of lower significance.

By induction, it can be proved that $\mathsf{H}(h) = \mathsf{Luc}(h+3) - 4$, and, consequently, $\mathsf{Q}(h) = \mathsf{Luc}(h+4) - h - 6$, where $\mathsf{Luc}(h)$ is the h-th Lucas number with base cases $\mathsf{Luc}(0) = 2$ and $\mathsf{Luc}(1) = 1$. The proof of these claim requires the application of the equalities $\mathsf{Luc}(2h+1) + 1 = \sum_{i=0}^{h} \mathsf{Luc}(2i)$, and $\mathsf{Luc}(2h) - 2 = \sum_{i=0}^{h-1} \mathsf{Luc}(2i+1)$.

In conclusion, since it is known that $\mathsf{Luc}(h) = \mathsf{Fib}(2h)/\mathsf{Fib}(h)$, the function $\mathsf{Q}(h)$ can be expressed as $\mathsf{Fib}(2(h+4)/\mathsf{Fib}(h+4) - (h+6)$, whose asymptotic behavior is a $\Theta\big(((1+\sqrt{5})/2)^{k/2}\big)$, as claimed in the theorem. □

4. Delayed Promotion Policy

At the beginning of the previous section, we have observed that the time complexity of the dominion search procedure $\mathsf{search}_{\mathcal{D}}$ linearly depends on the execution depth of the underlying dominion space $\mathcal{D}$. This, in turn, depends on the number of promotions performed by the associated successor function and is tightly connected with the reset mechanism applied to the regions with measure lower than the one of the target region of the promotion. In fact, it can be proved that, when no resets are performed, the number of possible promotions is bounded by a polynomial in the number of priorities and positions of the game under analysis. Consequently, the exponential behaviors exhibited by the PP algorithm and its enhanced version PP+ are strictly connected with the particular reset mechanism employed to ensure the soundness of the promotion approach. The correctness of the PP+ method shows that the reset can be restricted to only the regions of opposite parity *w.r.t.* the one of the promotion and, as we shall show in the next section, this enhancement is also relatively effective in practice. However, we have already noticed that this improvement does not suffice to avoid some pathological cases and no general finer criteria is available to avoid the reset of the opponent regions. Therefore, to further reduce such resets, in this section we propose a finer promotion policy that tries to reduce the application of the reset mechanism. The new solution procedure is based on delaying the promotions of regions, called *locked promotions*, that require the reset of previously performed promotions of the opponent parity, until a complete knowledge of the current search phase is reached. Once only locked promotions are left, the search phase terminates by choosing the highest measure $p^\star$ among those associated with the locked promotions and, then, performing all the postponed ones of the same parity as $p^\star$ altogether. In order to distinguish between locked and unlocked promotions, the corresponding target priorities of the performed ones, called *instant promotions*, are recorded in a supplementary set P. Moreover, to keep track of the locked promotions,

a supplementary partial priority function $\widetilde{\mathsf{r}}$ is used. In more detail, the new procedure evolves exactly as the PP+ algorithm, as long as open regions are discovered. When a closed one with measure p is provided by the query function, two cases may arise. If the corresponding promotion is not locked, the destination priority q is recorded in the set P and the instant promotion is performed similarly to the case of PP+. Otherwise, the promotion is not performed. Instead, it is recorded in the supplementary function $\widetilde{\mathsf{r}}$, by assigning to R in $\widetilde{\mathsf{r}}$ the target priority q of that promotion and in r its current measure p. Then, the positions in R are removed from the subgame and the search proceeds at the highest remaining priority, as in the case R was open in the subgame. In case the region R covers the entire subgame, all priorities available in the original game have been processed and, therefore, there is no further subgame to analyse. At this point, the delayed promotion to the highest priority $p^\star$ recorded in $\widetilde{\mathsf{r}}$ is selected and all promotions of the same parity are applied at once. This is done by first moving all regions from $\widetilde{\mathsf{r}}$ into r and then removing from the resulting function the regions of opposite parity *w.r.t.* $p^\star$, exactly as done by PP+. The search, then, resumes at priority $p^\star$. Intuitively, a promotion is considered as locked if its target priority is either *(a)* greater than some priority in P of opposite parity, which would be otherwise reset, or *(b)* lower than the target of some previously delayed promotion recorded in $\widetilde{\mathsf{r}}$, but greater than the corresponding priority set in r.

The latter condition is required to ensure that the union of a region in r together with the corresponding recorded region in $\widetilde{\mathsf{r}}$ is still a region. Observe that the whole approach is crucially based on the fact that when a promotion is performed all the regions having lower measure but the same parity are preserved. If this were not the case, we would have no criteria to determine which promotions need to be locked and which, instead, can be freely performed.

	1	2	3	4
6	$\mathtt{a}\downarrow$	...	...	$\mathtt{a},\mathtt{b},\mathtt{d},\mathtt{g},\mathtt{i},\mathtt{j}\downarrow$
5	$\mathtt{b},\mathtt{f},\mathtt{h}\downarrow$	...	...	
4	$\mathtt{c},\mathtt{j}\downarrow$	$\mathtt{c},\mathtt{e},\mathtt{j}\downarrow$	...	$\mathtt{c},\mathtt{e}\uparrow_6$
3	$\mathtt{d}\downarrow$	$\mathtt{d}\downarrow$	$\mathtt{d},\mathtt{g}\not\uparrow_5$	
2	$\mathtt{e}\uparrow_4$			
1		$\mathtt{g}\uparrow_3$		
0			$\mathtt{i}\uparrow_6$	

Table 2: DP simulation.

Example 4.1. *This idea is summarized by Table 2, which contains the execution of the new algorithm on the example in Figure 1. The computation proceeds as for* PP+*, until the promotion of the* 1*-region* $\{\mathtt{g}\}$ *shown in Column 2, occurs. This is an instant promotion to* 3*, since the only other promotion already computed and recorded in* P *has value* 4*. Hence, it can be performed and saved in* P *as well. Starting from priority* 3 *in Column 3, the closed* 1*-region* $\{\mathtt{d},\mathtt{g}\}$ *could be promoted to* 5*. However, since its target is greater than* $4 \in \mathrm{P}$*, it is delayed and recorded in* $\widetilde{\mathsf{r}}$*, where it is assigned priority* 5*. At priority* 0*, a delayed promotion of* 0*-region* $\{\mathtt{i}\}$ *to priority* 6 *is encountered and registered, since it would overtake priority* $3 \in \mathrm{P}$*. Now, the resulting subgame is empty. Since the highest delayed promotion is the one to priority* 6 *and no other promotion of the same parity was delayed,* 0*-region* $\{\mathtt{i}\}$ *is promoted and both the auxiliary priority function* $\widetilde{\mathsf{r}}$ *and the set of performed promotions* P *are emptied. The previously computed* 0*-region* $\{\mathtt{c},\mathtt{e},\mathtt{j}\}$ *has the same parity and, therefore, it is not reset, while the positions in both* 1*-regions* $\{\mathtt{b},\mathtt{f},\mathtt{h}\}$ *and* $\{\mathtt{d},\mathtt{g}\}$ *are reset to their original*

priorities. After maximization of the newly created 0*-region* $\{\mathtt{a},\mathtt{i}\}$*, positions* $\mathtt{b}$*,* $\mathtt{d}$*,* $\mathtt{g}$*, and* $\mathtt{j}$ *get attracted as well. This leads to the first cell of Column 4, where* 0*-region* $\{\mathtt{a},\mathtt{b},\mathtt{d},\mathtt{g},\mathtt{i},\mathtt{j}\}$ *is open. The next priority to process is* 4*, where* 0*-region* $\{\mathtt{c},\mathtt{e}\}$*, the previous* 0*-region* $\{\mathtt{c},\mathtt{e},\mathtt{j}\}$ *purged of position* $\mathtt{j}$*, is now closed in the corresponding subgame and gets promoted to* 6*. This results in a* 0*-region closed in the entire game, hence, a dominion for player* 0 *has been found. Note that postponing the promotion of* 1*-region* $\{\mathtt{d},\mathtt{g}\}$ *allowed a reduction in the number of operations. Indeed, the redundant maximization of* 1*-region* $\{\mathtt{b},\mathtt{f},\mathtt{h}\}$ *is avoided.*

It is worth noting that this procedure only requires a linear number of promotions, precisely $\lfloor \frac{h+1}{2} \rfloor$, on the lower bound game $\mathcal{D}_h^{\text{PP+}}$ for PP+. This is due to the fact that all resets are performed on regions that are not destination of any promotion.

DP *Dominion Space.* As it should be clear from the above informal description, the delayed promotion mechanism is essentially a refinement of the one employed in PP+. Indeed, the two approaches share all the requirements on the corresponding components of a state, on the orderings, and on compatibility relations. However, DP introduces in the state two supplementary elements, a partial priority function $\widetilde{\mathsf{r}}$, which collects the delayed promotions that were not performed on the region function r, and a set of priorities P, which collects the targets of the instant promotions performed. Hence, in order to formally define the corresponding dominion space, we need to provide suitable constraints connecting them with the other components of the search space. The role of function $\widetilde{\mathsf{r}}$ is to record the delayed promotions obtained by moving the corresponding positions from their priority p in r to the new measure q in $\widetilde{\mathsf{r}}$. Therefore, as dictated by Proposition 3.2, the union of $\mathsf{r}^{-1}(q)$ and $\widetilde{\mathsf{r}}^{-1}(q)$ must always be a region in the subgame $\mathcal{D}_{\mathsf{r}}^{\leq q}$. In addition, $\widetilde{\mathsf{r}}^{-1}(q)$ can only contain positions whose measure in r is of the same parity as q and recorded in r at some lower priority greater than the current one p. Formally, we say that a partial function $\widetilde{\mathsf{r}} \in \Delta^{\rightharpoonup} \triangleq \mathrm{Ps} \rightharpoonup \mathrm{Pr}$ is *aligned* with a region function r *w.r.t.* p if, for all priorities $q \in \mathsf{rng}(\widetilde{\mathsf{r}})$, it holds that *(a)* $\widetilde{\mathsf{r}}^{-1}(q) \subseteq \mathsf{dom}\big(\mathsf{r}^{(>p)\wedge(<q)\wedge(\equiv_2 q)}\big)$ and *(b)* $\mathsf{r}^{-1}(q) \cup \widetilde{\mathsf{r}}^{-1}(q)$ is an α-region in $\mathcal{D}_{\mathsf{r}}^{\leq q}$ with $\alpha \equiv_2 q$. The state space for DP is, therefore, defined as follows.

Definition 4.1 (State Space for DP)**.** *A* DP state space *is a tuple* $\mathcal{S} \triangleq \langle \mathrm{S}, \top, \prec \rangle$*, where:*

1. $\mathrm{S} \subseteq \mathrm{S}^{\text{PP+}} \times \Delta^{\rightharpoonup} \times 2^{\mathrm{Pr}}$ *is the set of all triples* $s \triangleq ((\mathsf{r},p),\widetilde{\mathsf{r}},\mathrm{P})$*, called* states*, composed by a* PP+ *state* $(\mathsf{r},p) \in \mathrm{S}^{\text{PP+}}$*, a partial priority function* $\widetilde{\mathsf{r}} \in \Delta^{\rightharpoonup}$ *aligned with* r w.r.t. p*, and a set of priorities* $\mathrm{P} \subseteq \mathrm{Pr}$*.*
2. $\top \triangleq (\top^{\text{PP+}}, \varnothing, \emptyset)$*;*
3. $s_1 \prec s_2$ *iff* $\widehat{s}_1 \prec^{\text{PP+}} \widehat{s}_2$*, for any two states* $s_1 \triangleq (\widehat{s}_1, \_, \_), s_2 \triangleq (\widehat{s}_2, \_, \_) \in \mathrm{S}$*.*

The second property we need to enforce is expressed in the compatibility relation connecting the query and successor functions for DP and regards the closed region pairs that are locked *w.r.t.* the current state. As stated above, a promotion is considered locked if its target priority is either *(a)* greater than some priority in P of opposite parity or *(b)* lower than the target of some previously delayed promotion recorded in $\widetilde{\mathsf{r}}$, but greater than the corresponding priority set in r. Condition *(a)* is the one characterizing the delayed promotion approach, as it reduces the number of resets of previously promoted

regions. The two conditions are expressed by the following two formulas, respectively, where q is the target priority of the blocked promotion.

$$\phi_a(q, \mathrm{P}) \triangleq \exists l \in \mathrm{P} \,.\, l \not\equiv_2 q \wedge l < q$$
$$\phi_b(q, \mathsf{r}, \widetilde{\mathsf{r}}) \triangleq \exists v \in \mathsf{dom}(\widetilde{\mathsf{r}}) \,.\, \mathsf{r}(v) < q \leq \widetilde{\mathsf{r}}(v)$$

Hence, an α-region R is called α-*locked w.r.t.* a state $s \triangleq ((\mathsf{r}, p), \widetilde{\mathsf{r}}, \mathrm{P})$ if the predicate $\phi_{Lck}(q, s) \triangleq \phi_a(q, \mathrm{P}) \vee \phi_b(q, \mathsf{r}, \widetilde{\mathsf{r}})$ is satisfied, where $q = \mathsf{bep}^{\overline{\alpha}}(\mathrm{R}, \mathsf{r} \uplus \widetilde{\mathsf{r}})$ is the best escape priority of the region R for player $\overline{\alpha}$ *w.r.t.* the combined region function $\mathsf{r} \uplus \widetilde{\mathsf{r}}$. In addition to the compatibility constraints for PP+, the compatibility relation for DP requires that any α-locked region, possibly returned by the query function, be maximal and contains the region $\mathsf{r}^{-1}(p)$ associated to the priority p of the current state.

Definition 4.2 (Compatibility Relation for DP)**.** *An open quasi dominion pair* $(\mathrm{R}, \alpha) \in \mathrm{QD}^{-}$ *is* compatible *with a state* $s \triangleq ((\mathsf{r}, p), \_, \_) \in \mathrm{S}$*, in symbols* $s \succ (\mathrm{R}, \alpha)$*, iff* (1) $(\mathrm{R}, \alpha) \in \mathrm{Rg}_{\mathcal{D}_s}$ *and* (2) *if* R *is* α*-open in* $\mathcal{D}_s$ *or it is* α*-locked* w.r.t. s *then* $\mathrm{R} = \mathsf{atr}^{\alpha}_{\mathcal{D}_s}(\mathsf{r}^{-1}(p))$.

Algorithm 5 implements the successor function for DP. The pseudo-code on the right-hand side consists of three macros used by the algorithm, namely $\#Assignment$, $\#InstantPromotion$, and $\#DelayedPromotion$. The first macro $\#Assignment(\xi)$ performs the insertion of a new region R into the region function r. In presence of a blocked promotion, *i.e.*, when the parameter ξ is set to $\mathtt{f}$, the region is also recorded in $\widetilde{\mathsf{r}}$ at the target priority of the promotion. The macro $\#InstantPromotion$ corresponds to the DP version of the standard promotion operation of PP+. The only difference is that it must also take care of updating the supplementary elements $\widetilde{\mathsf{r}}$ and P. The macro $\#DelayedPromotion$, instead, is responsible for the delayed promotion operation specific to DP.

Algorithm 5: Successor Function.

signature $\downarrow : \succ \to (\Delta \times \mathrm{Pr}) \times \Delta^{\rightharpoonup} \times 2^{\mathrm{Pr}}$
function $s\downarrow(\mathrm{R}, \alpha)$
  let $((\mathsf{r}, p), \widetilde{\mathsf{r}}, \mathrm{P}) = s$ **in**
1 **if** $(\mathrm{R}, \alpha) \in \mathrm{Rg}^{-}_{\mathcal{D}_s}$ **then**
2-5 $\#Assignment(\mathtt{t})$
  else
6 $q \leftarrow \mathsf{bep}^{\overline{\alpha}}(\mathrm{R}, \mathsf{r} \uplus \widetilde{\mathsf{r}})$
7 **if** $\phi_{Lck}(q, s)$ **then**
8 $\widehat{\mathsf{r}} \leftarrow \widetilde{\mathsf{r}}[\mathrm{R} \mapsto q]$
9 **if** $\mathrm{R} \neq \mathrm{Ps}_{\mathcal{D}_s}$ **then**
10-13 $\#Assignment(\mathtt{f})$
  else
14-17 $\#DelayedPromotion$
  else
18-21 $\#InstantPromotion$
22 **return** $((\mathsf{r}^{\star}, p^{\star}), \widetilde{\mathsf{r}}^{\star}, \mathrm{P}^{\star})$

Assignment & Promotion Macros.

macro $\#Assignment(\xi)$
1 $\mathsf{r}^{\star} \leftarrow \mathsf{r}[\mathrm{R} \mapsto p]$
2 $p^{\star} \leftarrow \mathsf{max}(\mathsf{rng}\left(\mathsf{r}^{\star(<p)}\right))$
3 $\widetilde{\mathsf{r}}^{\star} \leftarrow$ **#if** ξ **#then** $\widetilde{\mathsf{r}}$ **#else** $\widehat{\mathsf{r}}$
4 $\mathrm{P}^{\star} \leftarrow \mathrm{P}$

macro $\#DelayedPromotion$
1 $p^{\star} \leftarrow \mathsf{max}(\mathsf{rng}(\widehat{\mathsf{r}}))$
2 $\mathsf{r}^{\star} \leftarrow \mathsf{pr} \uplus (\mathsf{r} \uplus \widehat{\mathsf{r}})^{(\geq p^{\star}) \vee (\equiv_2 p^{\star})}$
3 $\widetilde{\mathsf{r}}^{\star} \leftarrow \emptyset$
4 $\mathrm{P}^{\star} \leftarrow \emptyset$

macro $\#InstantPromotion$
1 $p^{\star} \leftarrow q$
2 $\mathsf{r}^{\star} \leftarrow \mathsf{pr} \uplus \mathsf{r}^{(\geq p^{\star}) \vee (\equiv_2 p^{\star})}[\mathrm{R} \mapsto p^{\star}]$
3 $\widetilde{\mathsf{r}}^{\star} \leftarrow \widetilde{\mathsf{r}}^{(> p^{\star})}$
4 $\mathrm{P}^{\star} \leftarrow \mathrm{P} \cap \mathsf{rng}(\mathsf{r}^{\star}) \cup \{p^{\star}\}$

If the current region R is open in the subgame $\mathfrak{d}_s$, the main algorithm proceeds, similarly to Algorithm 4, at assigning to it the current priority p in r. This is done by calling macro $\#Assignment$ with parameter $\mathtt{t}$. Otherwise, the region is closed and a promotion should be performed at priority q, corresponding to the bep of that region *w.r.t.* the composed region function $\mathsf{r} \uplus \widetilde{\mathsf{r}}$. In this case, the algorithm first checks whether such promotion is locked *w.r.t.* s at Line 7. If this is not the case, then the promotion is performed as in PP+, by executing $\#InstantPromotion$, and the target is kept track of in the set P. If, instead, the promotion to q is locked, but some portion of the game still has to be processed, the region is assigned its measure p in r and the promotion to q is delayed and stored in $\widetilde{\mathsf{r}}$. This is done by executing $\#Assignment$ with parameter $\mathtt{f}$. Finally, in case the entire game has been processed, the delayed promotion to the highest priority recorded in $\widetilde{\mathsf{r}}$ is selected and applied. The macro $\#DelayedPromotion$ is executed, thus merging r with $\widetilde{\mathsf{r}}$. Function $\widetilde{\mathsf{r}}$ and set P are, then, erased, in order to begin a new round of the search. Observe that, when a promotion is performed, whether instant or delayed, we always preserve the underlying regions of the same parity, as done by the PP+ algorithm. This is a crucial step in order to avoid the pathological exponential worst case for the original PP procedure.

The soundness of the solution procedure relies on the following theorem.

Theorem 4.1 (DP Dominion Space)**.** *For a game $\mathfrak{d}$, the* DP structure $\mathcal{D} \triangleq \langle \mathfrak{d}, \mathcal{S}, \succ, \Re, \downarrow \rangle$*, where $\mathcal{S}$ is given in Definition 4.1, $\succ$ is the relation of Definition 4.2, and $\Re$ and $\downarrow$ are the functions computed by Algorithms 3 and 5, where in the former the assumption "***let** $(\mathsf{r}, p) = s$*" is replaced by "***let** $((\mathsf{r}, p), \_, \_) = s$*" is a dominion space.*

It is immediate to observe that the following mapping $\mathsf{h} : ((\mathsf{r}, p), \_, \_) \in \mathrm{S}^{\mathrm{DP}} \mapsto (\mathsf{r}, p) \in \mathrm{S}^{\mathrm{PP+}}$, which takes DP states to PP+ states by simply forgetting the additional elements $\widetilde{\mathsf{r}}$ and P, is a homomorphism. This, together with a trivial calculation of the number of possible states, leads to the following theorem.

Theorem 4.2 (DP Size & Depth Upper Bounds)**.** *The size of the* DP *dominion space $\mathcal{D}_{\mathfrak{d}}$ for a game $\mathfrak{d} \in \mathcal{P}$ with $n \in \mathbb{N}_+$ positions and $k \in [1, n]$ priorities is bounded by $2^k k^{2n}$. Moreover, its depth is not greater then the one of the* PP+ *dominion space $\mathcal{D}_{\mathfrak{d}}^{\mathrm{PP+}}$ for the same game.*

Unfortunately, also this improved promotion policy happens to suffer from exponential behaviors. However, the exponential lower-bound family we found has a much more complex structure than the one previously described for PP+. In particular, it requires multiple positions with the same priority in order to fool the delayed promotion criterion and force the algorithm to behave exactly as PP+. There seems to be no obvious way to enforce a similar behavior on games having a single position for each priority. Since it is well known that any parity game can be transformed into an equivalent one with as many priorities as positions [41], the DP approach can still be considered as a candidate to a polynomial-time solution of the problem.

In this family, the game $\mathfrak{d}_h^{\mathrm{DP}}$ of index $h \geq 1$, where h denotes the number of chains it contains, is defined as follows. For all indexes $i \in [1, h]$ and $j \in [i - 1, h - (i \bmod 2)] \cup \{h + i - 1 + (h \bmod 2)\}$, there is a position (i, j), which

we shortly denote by j_i. Position j_i is connected to the i-th chain, belongs to player $i \bmod 2$, *i.e.*, $j_i \in \mathrm{Ps}^{(i \bmod 2)}$, and has priority j, *i.e.*, $\mathsf{pr}(j_i) = j$. For any chain $i \in]1, h]$, the head position $(h + i - 1 + (h \bmod 2))_i$ has a unique move to the head $(h+i-2+(h \bmod 2))_{i-1}$ of the lower chain $i-1$. Moreover, the head $(h+(h \bmod 2))_1$ of the first chain is connected to the tails $(h-1)_i$ of all the chains of odd index, *i.e.*, with $i \equiv_2 1$. The lower position $(i-1)_i$ in each chain has a self-move plus one move to the head $(h + i - 1 + (h \bmod 2))_i$. Finally, each position j_i, with $j \in [i, h - (i \bmod 2)]$, has a unique move to a lower position $(j-1)_i$ in the same chain. Figure 3 depicts the instance of ∂_h^{DP} with $h = 4$.

Similarly to the execution depth of the PP+ dominion space of the game in Figure 2, the execution depth of the DP dominion space for this game is exponential in h. Indeed, the promotion operations performed on each chain, merging its second position to its head, can simulate the increments of a partial binary counter with h digits. The number of configurations of this counter for the DP algorithm can be proved to be $\mathsf{P}(h) + \lceil n/2 \rceil - 2$, where P is the function counting the number of configurations for the PP+ shown in the proof of Theorem 3.2.

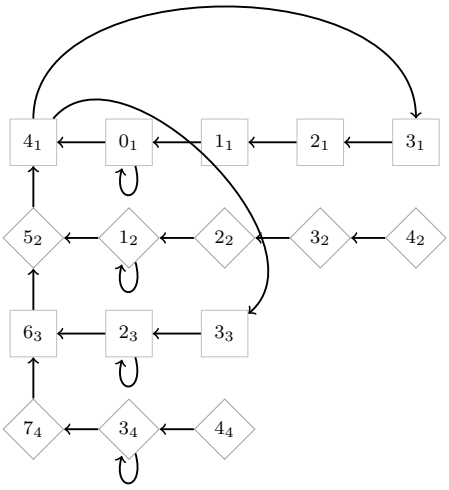

Figure 3: The ∂_4^{DP} game.

The search procedure on ∂_4^{DP} starts by building the following seven open regions: the 1-region $\{7_4\}$, the 0-region $\{6_3\}$, the 1-region $\{5_2\}$, the 0-region $\{4_1, 4_2, 4_4\}$, the 1-region $\{3_1, 3_2, 3_3, 3_4\}$, the 0-region $\{2_1, 2_2, 2_3\}$, and the 1-region $\{1_1, 1_2, \}$. This state represents the configuration of the counter, where all four digits are set to 0. The closed 0-region $\{0_1\}$ is then found and promoted to 4. Now, the counter is set to 0001. After that, the 1-region $\{3_2, 3_3, 3_4\}$ is built again followed by the 0-region $\{2_2, 2_3\}$. Next, the closed 1-region $\{1_2\}$ is computed, and is promoted to 5. Due to the reset criterion, the positions in the 0-region with priority 4 are reset to their original priority, as they belong to the opponent player *w.r.t.* to the promoted region. This releases the chain with head 4, which corresponds to the reset of the least significant digit of the counter caused by the increment of the second one, *i.e.*, the counter displays 0010. The search resumes at priority 5 and the 0-region $\{4_1, 4_4\}$, the 1-region $\{3_1, 3_3, 3_4\}$, the 0-region $\{2_1, 2_3\}$, the 1-region $\{1_1\}$, and the 0-region $\{0_1\}$ are computed once again. A second promotion of $\{0_1\}$ to 4 is performed, resulting in the counter assuming value 0011. When the closed 0-region $\{2_3\}$ is promoted to 6, however, only the 1-region $\{5_2, 1_2, 2_2, 3_2, 4_2\}$ is reset, leading to configuration 0101. Hence, configuration 0100 is skipped. Similarly, when, the counter reaches configuration 0111 and 1-region $\{3_4\}$ is promoted to 7, the 0-regions $\{4_1, 0_1, 1_1, 2_1, 3_1\}$ and $\{6_3, 2_3, 3_3\}$ are reset, leaving 1-region $\{5_2, 1_2, 2_2, 3_2, 4_2\}$ intact. This leads directly to configuration 1010 of the counter, skipping configurations 1000 and 1001.

Observe that for each configuration, but the initial one, a promotion is required. So

the number of promotions to reach the all-1 configuration is $\mathsf{Fib}(h+3)-1$. Once the last configuration is reached, the region of the least significant digit is promoted to the region corresponding to the third significant digit, which is, in turn, promoted to the fifth significant digit and so on, until a region containing all positions of the game, but those of the most significant digit, if h is even, is obtained. Such region is a dominion and the algorithm terminates. The number of these additional promotions is exactly $\lceil n/2 \rceil - 2$. Hence, the total number is $\mathsf{Fib}(h+3) - 1 + \lceil n/2 \rceil - 2$. The observation above allows us to provide an estimation of the depth of the DP dominion space.

Theorem 4.3 (Execution-Depth Lower Bound)**.** *For all $h \in \mathbb{N}$, there exists a* DP *dominion space $\mathcal{D}_h^{\mathrm{DP}}$ with $n = 2h + (h^2 - h \bmod 2)/2$ positions and $k = 2h$ priorities, whose execution depth is at least* $\mathsf{Fib}(h+3) - 3 + \lceil n/2 \rceil = \Theta\Big(((1+\sqrt{5})/2)^{\sqrt{n}}\Big)$.

Note that in a game $\mathcal{D}_h^{\mathrm{DP}}$, the number of positions is quadratic in the number of priorities, hence, $k = \mathrm{O}(\sqrt{n})$. This appears to be a crucial element in order to force the DP algorithm to behave as PP+. Indeed, it is essential for every region of priority $p \in [1, k-1]$ to be open until all the lower ones with priorities $p' < p$ are promoted to the corresponding measure $h + p' + (h \bmod 2)$. The only way to do this seems to require $p+1$ positions with the same parity p.

5. Experimental Evaluation

The technique proposed in the paper has been implemented in PGSOLVER [42], a tool that collects implementations of several parity game solvers proposed in the literature. The tool provides benchmarking tools that can be used to evaluate the performance of the solvers on both concrete and synthetic benchmarks. The concrete ones include three classes of games, namely Towers-of-Hanoi, Elevator-Verification and Language-Inclusion, which encode standard verification problems. The synthetic ones, instead, provide both worst-case exponential families for the solvers included in the tool and random games generators. The solvers considered in the experiments include the original PP algorithm [1], the two versions PP+ and DP presented in this paper, the Recursive algorithm *Rec* [29], the two Dominion Decomposition algorithms *DomDec* [30] and *BigStp* [31], and the Strategy Improvement algorithm *StrImp* [33].

We also experimented with Small Progress Measure [32] and with the two quasi-polynomial solvers recently proposed in [36] and [37]. These last three solvers, however, were unable to solve any of the benchmarks considered within the time limit and have been left out from the following experimental evaluation.

Table 3 reports the results of the solvers on the benchmark families available in PGSOLVER. We only report on the biggest instances we could deal with, given the available computational resources [2]. The parameter Positions gives the number of

[1] The version of PP used in the experiments is actually an improved implementation of the one described in [38].

[2] All the experiments were carried out on a 64-bit 3.1GHz INTEL® quad-core machine, with i5-2400 processor and 8GB of RAM, running UBUNTU 12.04 with LINUX kernel version 3.2.0. PGSOLVER was compiled with OCaml version 2.12.1.

Benchmark	Positions	*DomDec*	*BigStp*	*StrImp*	*Rec*	PP	PP+	DP
Hanoi	$6.3 \cdot 10^6$	21.4	21.4	‡	17.4	**7.0**	**7.0**	7.4
Elevator	$7.7 \cdot 10^6$	†	‡	‡	‡	19.8	**19.7**	20.4
Lang. Incl.	$5 \cdot 10^6$	†	‡	‡	145.5	**16.5**	**16.5**	**16.5**
Ladder	$4 \cdot 10^6$	†	‡	‡	35.0	**7.9**	**7.9**	8.1
Str. Imp.	$4.5 \cdot 10^6$	81.0	82.8	†	71.0	**57.0**	57.1	57.3
Clique	$8 \cdot 10^3$	†	‡	†	†	**10.8**	10.9	**10.8**
MC. Lad.	$7.5 \cdot 10^6$	†	‡	‡	**4.3**	4.4	**4.3**	4.5
Rec. Lad.	$5 \cdot 10^4$	†	‡	‡	‡	**62.8**	63.0	64.9
Jurdziński	$4 \cdot 10^4$	†	†	188.2	†	**69.6**	69.7	71.8
WC. Rec.	$3 \cdot 10^4$	†	†	**9.4**	†	10.2	10.2	18.2

Table 3: Execution times in seconds on several benchmark families. Time out (†) is set to 600 seconds and memory out (‡) to 7.5Gb.

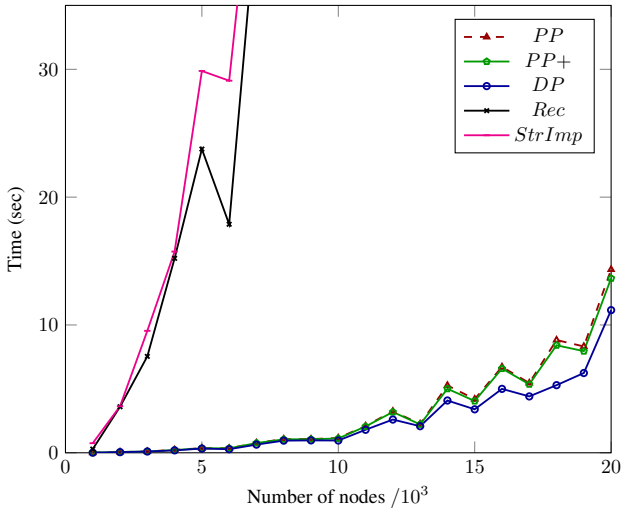

Figure 4: Solution times on random games from [39].

positions in the games and the best performance are emphasized in bold.[3] The first three rows consider the concrete verification problems mentioned above. On the Tower-of-Hanoi problem all the solvers perform reasonably well, except for *StrImp* due its high memory requirements. The Elevator-Verification problem proved to be very demanding in terms of memory for all the solvers, except for the priority-promotion based algorithms and *DomDec*, which, however, could not solve it within the time limit of 10 minutes. Our solvers perform extremely well on both this benchmark and on Language Inclusion, which could be solved only by *Rec* among the other solvers.

On the worst case benchmarks, they all perform quite well on Ladder, Strategy

[3]The instances were generated by issuing the following PGSOLVER commands: `towersofhanoi 13`, `elevatorgame 8`, `langincl 500 100`, `cliquegame 8000`, `laddergame 4000000`, `stratimprgen -pg friedmannsubexp 1000`, `modelcheckerladder 2500000`, `recursiveladder 10000`, and `jurdzinskigame 100 100`.

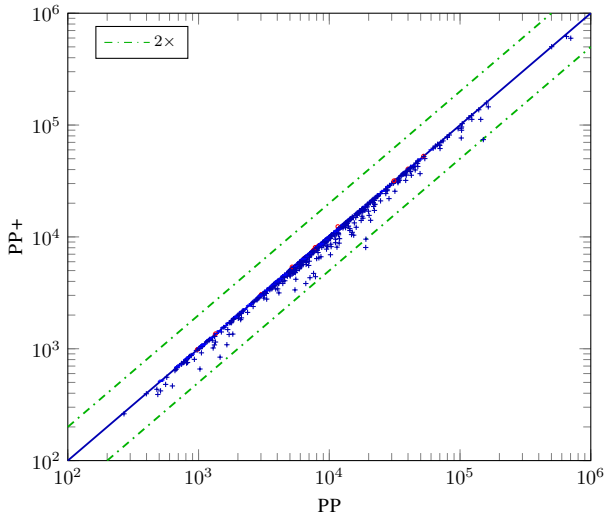

Figure 5: Number of promotions: comparison between PP and PP+ on random games with 50000 positions.

Improvement, Jurdziński, Recursive Ladder, and even on Clique, which proved to be considerably difficult for all the other solvers. The Modelchecker game was essentially a tie with *Rec*. The only family on which they were slightly outperformed by the Strategy Improvement algorithm is the new worst case family *WC-Rec*, which was introduced as a robust worst-case for various solvers in [43]. On these benchmarks the new algorithms exhibit the most consistent behavior overall. Indeed, on all those families the priority-promotion based algorithms perform no promotions, regardless of the input parameters, except for the Elevator-Verification problem, where they require only two promotions. This constant bound on the number of promotions implies that, for those games, the three solvers, PP, PP+, and DP only require time linear in the size of the game and the number of priorities, which explains the similarity of their performance. It is worth noting that, when few promotions are involved, DP is likely to suffer from some overhead *w.r.t.* both PP and PP+. When, on the other hand, the number of promotions increases, the overhead is often compensated by the reduction of the number of promotions required to solve the game, as witnessed by the experiments on random games described below.

Figure 4 compares the running times of the new algorithms, PP+ and DP, against the original PP and the solvers *Rec* and *StrImp* on randomly generated games. The other two solvers *DomDec* and *BigStep* perform quite poorly on those games, hitting the time-out already for very small instances. This first pool of benchmarks contains the same games used in the experimental evaluation presented in [39]. It contains 2000 random games of size ranging from 1000 to 20000 positions and 2 outgoing moves per position. Interestingly, random games with very few moves prove to be much more challenging for the priority promotion based approaches than those with a higher number of moves per position, and often require a much higher number of promotions.

Since the behavior of the solvers is typically highly variable, even on games of

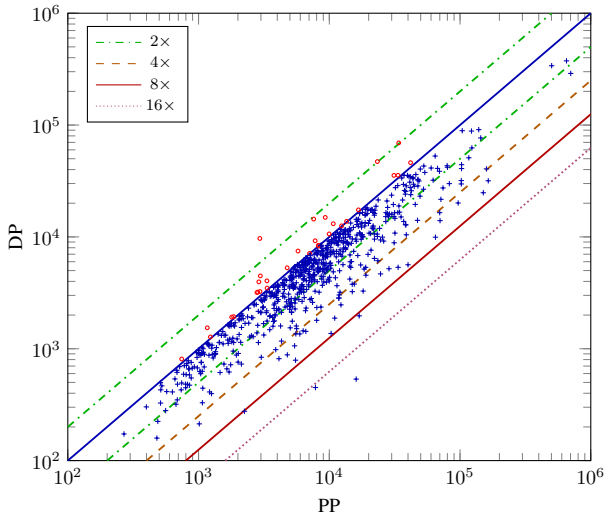

Figure 6: Number of promotions: comparison between PP and DP on random games with 50000 positions.

the same size and priorities, to summaries the results we took the average running time on clusters of games. Therefore, each point in the graph shows the average time over a cluster of 100 different games of the same size: for each size value n, we chose the numbers $k = n \cdot i/10$ of priorities, with $i \in [1, 10]$, and 10 random games were generated for each pair n and k. We set a time-out to 180 seconds (3 minutes). Solver PP+ performs slightly better than PP, while DP shows a much more convincing improvement on the average time.

Similar experiments were also conducted on random games with a higher number of moves per position and up to 1000000 positions. The resulting games turn out to be very easy to solve by all the priority promotion based approaches, requiring few seconds only. The reason seems to be that the higher number of moves significantly increases the dimension of the computed regions and, consequently, also the chances to find a closed one. Indeed, the number of promotions required by PP+ and DP on all those games is typically zero, and the whole solution time is due exclusively to a very limited number of attractors needed to compute the few regions contained in the games. The only other solver that can easily solve these games is *Rec*, whose performance is only slightly worse than that of the priority-promotion-based approaches.

Since the exponential behaviors of the priority-promotion-based algorithms are tightly connected with the number of promotions needed to solve a game and the main aim is to reduce such a number, we devised specific benchmarks towards analyzing the behavior of the algorithms *w.r.t.* this measure. The second pool of benchmarks contains 740 games, each with 50000 positions, 2 moves per positions and priorities varying from 8000 to 12000. These games are much harder than the previous ones and have been selected among random games, whose solution requires PP more than 500 and up to 700000 promotions to be solved. On these games we measured both the number of promotions performed and the solution times.

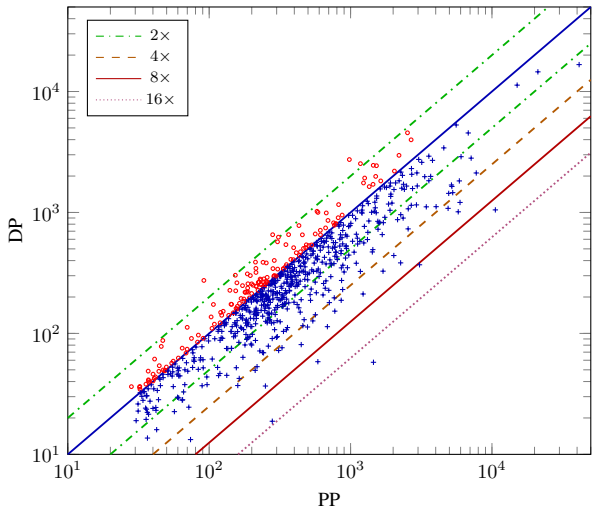

Figure 7: Solution time: comparison between PP and DP on random games with 50000 positions.

Figure 5 reports the experimental results comparing the number of promotions performed by PP and PP+ on these games and provides experimental support to the expectation that PP+ almost always requires less effort than PP, even though the benefits appear quite limited. This essentially confirms the results reported in Figure 4.

On the other hand, Figure 6, comparing DP and PP, reveals that DP does reduce the number promotions considerably. The benefits of the new promotion policy appear to be substantial as, in many cases, PP requires between two to eight times as many promotions as DP to solve a game. Notice that the scale in the figures is logarithmic and the diagonal lines are labeled with the corresponding multiplication factor. The figure also shows that, in very few cases, PP does require less promotions than DP. This is due to the fact that the two algorithms typically follow different solution paths within the dominion space and that delaying promotions may defer the discovery of a closed dominion. Nonetheless, the DP policy does pay off significantly on the vast majority of the benchmarks. Finally, Figure 7 shows how the reduction on the number of promotions translates into solution times. The results, once again reported on a logarithmic scale, confirm that, despite the additional overhead in the computation of DP that sometimes outweighs the gain in terms of promotions shown in Figure 6, DP outperforms PP on most of the benchmarks. Recently, the new tool OINK [44] has been presented. The tool, written in C++, provides efficient implementations of several parity games algorithms. Experiments performed with OINK on the same benchmarks used here essentially confirm the results obtained with PGSOLVER reported above.

6. Discussion

Devising efficient algorithms that can solve parity games well in practice is a crucial endeavor towards enabling formal verification techniques, such as model checking of expressive temporal logics and automatic synthesis, in practical contexts. To this

end, a promising new solution technique, called *priority promotion*, was proposed in [38]. While the technique seems very effective in practice, the approach still admits exponential behaviors. This is due to the fact that, to ensure correctness, it needs to forget previously computed partial results after each promotion. In this work we presented a new promotion policy that delays promotions as much as possible, in the attempt to reduce the need to partially reset the state of the search. Not only the new technique, like the original one, solves in polynomial time all the exponential worst cases known for other solvers, but requires polynomial time for the worst cases of the priority promotion approach as well.

Even though a family of games requiring exponential time does exist, the exponential behavior seems to rely heavily on the fact the positions are in a quadratic relationship with the priorities and it does not seem obvious how to lift it to the cases where those two parameters are linearly dependent. Therefore, the actual complexity of the algorithm remains an open problem.

Experiments on benchmarks families and randomly generated games also show that the new technique often outperforms the original priority promotion technique, as well as the state-of-the-art solvers known in the literature, including the two recent quasi-polynomial algorithms proposed in [36] and [37].

Appendix A. Proof of Theorem 3.1

In this appendix we report the proof of Theorem 3.1 stated in Section 3, providing soundness and completenessof the PP+ search procedure.

Theorem 3.1 (PP+ Dominion Space)**.** *For a game $\mathcal{D}$, the* PP+ structure $\mathcal{D} \triangleq \langle \mathcal{D}, \mathcal{S}, \succ, \Re, \downarrow \rangle$*, where $\mathcal{S}$ is given in Definition 3.4, $\succ$ is the relation of Definition 3.5, and $\Re$ and $\downarrow$ are the functions computed by Algorithms 3 and 4 is a dominion space.*

To prove Theorem 3.1, we have to show that the three components $\mathcal{S}$, $\Re$, and $\downarrow$ of the structure $\mathcal{D}$ satisfy the properties required by Definition 3.2 of dominion space. We do this through the Lemmas Appendix A.1, Appendix A.2, and Appendix A.3.

Lemma Appendix A.1 (State Space)**.** *The* PP+ *state space $\mathcal{S} = \langle \mathrm{S}, \top, \prec \rangle$ for a game $\mathcal{D} \in \mathcal{P}$ is a well-founded partial order* w.r.t. $\prec$ *with designated element* $\top \in \mathrm{S}$.

Proof. Since S is a finite set, to show that $\prec$ is a well-founded partial order on S, it is enough to prove that it is simply a strict partial order on the same set, *i.e.*, an irreflexive and transitive relation.

For the irreflexive property, by Item 3 of Definition 3.4, it is immediate to see that $s \not\prec s$, for all states $s \triangleq (\mathsf{r}, p) \in \mathrm{S}$, since neither there exists a priority $q \in \mathsf{rng}(\mathsf{r})$ such that $\mathsf{r}^{-1}(q) \subset \mathsf{r}^{-1}(q)$ nor $p < p$.

For the transitive property, instead, consider three states $s_1 \triangleq (\mathsf{r}_1, p_1)$, $s_2 \triangleq (\mathsf{r}_2, p_2)$, $s_3 \triangleq (\mathsf{r}_3, p_3) \in \mathrm{S}$ for which $s_1 \prec s_2$ and $s_2 \prec s_3$ hold. Due to Items 3.a and 3.b of the same definition, four cases may arise.

- Item 3.a for both $s_1 \prec s_2$ and $s_2 \prec s_3$: there exist two priorities $q_1 \in \mathsf{rng}(\mathsf{r}_1)$ and $q_2 \in \mathsf{rng}(\mathsf{r}_2)$ with $q_1 \geq p_1$ such that ${\mathsf{r}_1}^{(>q_1)} = {\mathsf{r}_2}^{(>q_1)}$, ${\mathsf{r}_2}^{(>q_2)} = {\mathsf{r}_3}^{(>q_2)}$,

$\mathsf{r}_2^{-1}(q_1) \subset \mathsf{r}_1^{-1}(q_1)$, and $\mathsf{r}_3^{-1}(q_2) \subset \mathsf{r}_2^{-1}(q_2)$. Let $q \triangleq \mathsf{max}\{q_1, q_2\} \geq p_1$. If $q = q_1 = q_2$ then $\mathsf{r}_1{}^{(>q)} = \mathsf{r}_2{}^{(>q)} = \mathsf{r}_3{}^{(>q)}$ and $\mathsf{r}_3^{-1}(q) \subset \mathsf{r}_2^{-1}(q) \subset \mathsf{r}_1^{-1}(q)$. If $q = q_1 > q_2$ then $\mathsf{r}_1{}^{(>q)} = \mathsf{r}_2{}^{(>q)} = (\mathsf{r}_2{}^{(>q_2)})^{(>q)} = (\mathsf{r}_3{}^{(>q_2)})^{(>q)} = \mathsf{r}_3{}^{(>q)}$ and $\mathsf{r}_3^{-1}(q) = \mathsf{r}_2^{-1}(q) \subset \mathsf{r}_1^{-1}(q)$. Finally, if $q = q_2 > q_1$ then $\mathsf{r}_1{}^{(>q)} = (\mathsf{r}_1{}^{(>q_1)})^{(>q)} = (\mathsf{r}_2{}^{(>q_1)})^{(>q)} = \mathsf{r}_2{}^{(>q)} = \mathsf{r}_3{}^{(>q)}$ and $\mathsf{r}_3^{-1}(q) \subset \mathsf{r}_2^{-1}(q) = \mathsf{r}_1^{-1}(q)$. Moreover, $q \in \mathsf{rng}\big(\mathsf{r}_2{}^{(>q_1)}\big) = \mathsf{rng}\big(\mathsf{r}_1{}^{(>q_1)}\big) \subseteq \mathsf{rng}(\mathsf{r}_1)$. Summing up, it holds that $s_1 \prec s_3$.

- Item 3.a for $s_1 \prec s_2$ and Item 3.b for $s_2 \prec s_3$: there exists a priority $q \in \mathsf{rng}(\mathsf{r}_1)$ with $q \geq p_1$ such that $\mathsf{r}_1{}^{(>q)} = \mathsf{r}_2{}^{(>q)}$ and $\mathsf{r}_2^{-1}(q) \subset \mathsf{r}_1^{-1}(q)$; moreover, $\mathsf{r}_2 = \mathsf{r}_3$. Thus, $\mathsf{r}_1{}^{(>q)} = \mathsf{r}_2{}^{(>q)} = \mathsf{r}_3{}^{(>q)}$ and $\mathsf{r}_3^{-1}(q) = \mathsf{r}_2^{-1}(q) \subset \mathsf{r}_1^{-1}(q)$. Consequently, $s_1 \prec s_3$.

- Item 3.a for $s_2 \prec s_3$ and Item 3.b for $s_1 \prec s_2$: there exists a priority $q \in \mathsf{rng}(\mathsf{r}_2)$ with $q \geq p_2$ such that $\mathsf{r}_2{}^{(>q)} = \mathsf{r}_3{}^{(>q)}$ and $\mathsf{r}_3^{-1}(q) \subset \mathsf{r}_2^{-1}(q)$; moreover, $\mathsf{r}_1 = \mathsf{r}_2$ and $p_1 < p_2$. Thus, $\mathsf{r}_1{}^{(>q)} = \mathsf{r}_2{}^{(>q)} = \mathsf{r}_3{}^{(>q)}$, $\mathsf{r}_3^{-1}(q) \subset \mathsf{r}_2^{-1}(q) = \mathsf{r}_1^{-1}(q)$, $q \in \mathsf{rng}(\mathsf{r}_1)$, and $q > p_1$. Consequently, $s_1 \prec s_3$.

- Item 3.b for both $s_1 \prec s_2$ and $s_2 \prec s_3$: $\mathsf{r}_1 = \mathsf{r}_2$, $\mathsf{r}_2 = \mathsf{r}_3$, $p_1 < p_2$, and $p_2 < p_3$. Hence, $\mathsf{r}_1 = \mathsf{r}_3$ and $p_1 < p_3$, which implies that $s_1 \prec s_3$ also in this case.

To conclude the proof, we have to show that $\top \triangleq (\mathsf{pr}, \mathsf{pr}(\mathcal{D}))$ belongs to S. Indeed, Item 1.b follows from the fact that $p = \mathsf{pr}(\mathcal{D}) = \mathsf{max}(\mathsf{rng}(\mathsf{pr})) \in \mathsf{rng}(\mathsf{pr}) = \mathsf{rng}(\mathsf{r})$. Obviously, Item 1.a also holds, since r is vacuously maximal above p. Finally, $\mathsf{pr}_{\mathcal{D}}$ is a region function as, for all priorities $q \in \mathsf{rng}(\mathsf{m})$ with $\alpha \triangleq q \bmod 2$, where m is the maximisation of pr, it holds that $\mathsf{pr}^{-1}(q) \cap \mathrm{Ps}_{\mathcal{D}^{\leq q}_{\mathsf{m}}}$ is an α-region in the subgame $\mathcal{D}^{\leq q}_{\mathsf{m}}$. Indeed, the set $\mathsf{pr}^{-1}(q) \cap \mathrm{Ps}_{\mathcal{D}^{\leq q}_{\mathsf{m}}}$ can only contain positions of priority q, which is the maximal one in the corresponding subgame $\mathcal{D}^{\leq q}_{\mathsf{m}}$. Therefore, player α has an obvious strategy that forces every infinite play inside this set to be winning for it. □

Lemma Appendix A.2 (Query Function)**.** *The function $\Re$ is a query function,* i.e.*, for all states $s \in \mathrm{S}$, it holds that* (1) $\Re(s) \in \mathrm{QD}$ *and* (2) *if* $\Re(s) \in \mathrm{QD}^{-}$ *then* $s \succ \Re(s)$.

Proof. Let $s \triangleq (\mathsf{r}, p) \in \mathrm{S}$ be a state and $(\mathrm{R}, \alpha) \triangleq \Re(s)$ the pair consisting of the set of positions $\mathrm{R} \subseteq \mathrm{Ps}$ and the player $\alpha \in \{0, 1\}$ returned by the function $\Re$ on input s. Due to Line 1 of Algorithm 3, it follows that $\alpha \equiv_2 p$. By Item 1 of Definition 3.4, we have that $p \in \mathsf{rng}(\mathsf{r})$, so $\mathsf{r}^{-1}(p) \neq \emptyset$, and $\mathsf{r}^{-1}(p) \subseteq \mathrm{Ps}_{\mathcal{D}_s}$. Thus, by definition of region function, it holds that $\mathsf{r}^{-1}(p)$ is an α-region in $\mathcal{D}_s$. Now, by Line 2 of Algorithm 3 and Proposition 3.1, we obtain that $\mathrm{R} = \mathsf{atr}^{\alpha}_{\mathcal{D}_s}(\mathsf{r}^{-1}(p)) \subseteq \mathrm{Ps}_{\mathcal{D}_s}$ is an α-region in $\mathcal{D}_s$, *i.e.*, $(\mathrm{R}, \alpha) \in \mathrm{Rg}_{\mathcal{D}_s}$, and, so a quasi dominion in $\mathcal{D}$, *i.e.*, $(\mathrm{R}, \alpha) \in \mathrm{QD}$. In addition, $s \succ (\mathrm{R}, \alpha)$, since all requirements of Definition 3.5 are satisfied. □

Lemma Appendix A.3 (Successor Function)**.** *The function $\downarrow$ is a successor function,* i.e.*, for all states $s \in \mathrm{S}$ and quasi dominion pairs $(\mathrm{R}, \alpha) \in \mathrm{QD}^{-}$ with $s \succ (\mathrm{R}, \alpha)$, it holds that* (1) $s \downarrow (\mathrm{R}, \alpha) \in \mathrm{S}$ *and* (2) $s \downarrow (\mathrm{R}, \alpha) \prec s$.

Proof. Let $s \triangleq (\mathsf{r}, p) \in \mathrm{S}$ be a state, $(\mathrm{R}, \alpha) \in \mathrm{QD}^{-}$ an open quasi dominion pair in $\mathcal{D}$ compatible with s, and $s^\star = (\mathsf{r}^\star, p^\star) \triangleq s \downarrow (\mathrm{R}, \alpha)$ the result obtained by computing the function $\downarrow$ on s and (R, α). Due to Item 1.a of Definition 3.4, we have that *(1)* r is maximal above p. Moreover, by Item 1 of Definition 3.5, it holds that $\mathrm{R} \subseteq \mathrm{Ps}_{\mathcal{D}_s}$, which implies *(2)* $\mathsf{dom}\big(\mathsf{r}^{(>p)}\big) \cap \mathrm{R} = \emptyset$.

On the one hand, suppose that R is α-open in $\mathcal{D}_s$, *i.e.*, $(\mathrm{R}, \alpha) \in \mathrm{Rg}^{-}_{\mathcal{D}_s}$. By Lines 1-3 of Algorithm 4, we have that *(3)* $\mathsf{r}^\star = \mathsf{r}[\mathrm{R} \mapsto p]$ and *(4)* $p^\star = \mathsf{max}(\mathsf{rng}\Big(\mathsf{r}^{\star(<p)}\Big))$. In addition, by Item 2 of Definition 3.5, it holds that *(5)* $\mathrm{R} = \mathsf{atr}^{\alpha}_{\mathcal{D}_s}(\mathsf{r}^{-1}(p))$. Now, by Point *(4)*, it immediately follows that *(6)* $p^\star \in \mathsf{rng}(\mathsf{r}^\star)$, *(7)* $p^\star < p$, and *(8)* there is no priority $q \in \mathsf{rng}(\mathsf{r}^\star)$ such that $p^\star < q < p$. Also, by Points *(2)*, *(3)* and *(5)*, we have that *(9)* $\mathsf{r}^{\star(>p)} = \mathsf{r}^{(>p)}$, *(10)* $\mathsf{r}^{\star -1}(p) = \mathrm{R}$, and *(11)* $\mathsf{r}^{\star(<p)} = \mathsf{r}^{(<p)} \restriction (\mathrm{Ps} \setminus \mathrm{R})$. Thus, by Points *(1)*, *(5)*, *(8)*, *(9)*, and *(10)*, it holds that *(12)* $\mathsf{r}^\star$ is maximal above $p^\star$. Finally, to prove that *(13)* $\mathsf{r}^\star$ is a region function, we need to show that $\mathsf{r}^{\star -1}(q) \cap \mathrm{Ps}_{\mathcal{D}^{\leq q}_{\mathsf{m}^\star}}$ is a β-region in the subgame $\mathcal{D}^{\leq q}_{\mathsf{m}^\star}$, for all priorities $q \in \mathsf{rng}(\mathsf{m}^\star)$ with $\beta \triangleq q \bmod 2$, where $\mathsf{m}^\star$ is the maximisation of $\mathsf{r}^\star$. To this aim, we observe that, by definition of state space, r is a region function, so, the above property is surely satisfied by r *w.r.t.* its maximisation m.

- If $q > p$, by Points *(1)* and *(9)*, we have that $\mathsf{r}^{\star -1}(q) = \mathsf{r}^{-1}(q) \subseteq \mathrm{Ps}_{\mathcal{D}^{\leq q}_{\mathsf{m}}} = \mathrm{Ps}_{\mathcal{D}^{\leq q}_{\mathsf{m}^\star}}$. Therefore, the thesis immediately follows, since $\mathsf{r}^{-1}(q)$ is a β-region in the subgame $\mathcal{D}^{\leq q}_{\mathsf{m}}$, where $\beta \triangleq q \bmod 2$.
- If $q = p$, first observe that $\mathcal{D}_s = \mathcal{D}^{\leq q}_{\mathsf{m}} = \mathcal{D}^{\leq q}_{\mathsf{m}^\star}$. By Points *(2)*, *(5)*, and *(10)*, we have that $\mathsf{r}^{\star -1}(q) \cap \mathrm{Ps}_{\mathcal{D}^{\leq q}_{\mathsf{m}^\star}} = \mathrm{R}$. So, the property is ensured by the fact that R is an α-region in $\mathcal{D}_s$.
- For the last case $q < p$, notice that $\mathrm{Ps}_{\mathcal{D}^{\leq q}_{\mathsf{m}}} \cap \mathrm{R} = \emptyset$. Therefore, by Point *(11)*, $\mathsf{r}^{-1}(q) \cap \mathrm{Ps}_{\mathcal{D}^{\leq q}_{\mathsf{m}}} = \mathsf{r}^{\star -1}(q) \cap \mathrm{Ps}_{\mathcal{D}^{\leq q}_{\mathsf{m}^\star}}$. Hence, the thesis follows, since $\mathsf{r}^{-1}(q) \cap \mathrm{Ps}_{\mathcal{D}^{\leq q}_{\mathsf{m}}}$ is a β-region in the subgame $\mathcal{D}^{\leq q}_{\mathsf{m}}$, where $\beta \triangleq q \bmod 2$.

Summing up, Points *(13)*, *(12)*, and *(6)* ensure that $(\mathsf{r}^\star, p^\star) \in \mathrm{S}$. At this point, it remains just to show that $(\mathsf{r}^\star, p^\star) \prec (\mathsf{r}, p)$. By Point *(5)*, two cases may arise. If $\mathsf{r}^{-1}(p) \subset \mathrm{R}$, due to Points *(9)* and *(10)*, the thesis follows from Item 3.a of Definition 3.4, where the priority q is set to p. On the contrary, if $\mathrm{R} = \mathsf{r}^{-1}(p)$, by Point *(3)*, we have that $\mathsf{r}^\star = \mathsf{r}$. Therefore, due to Point *(7)*, the thesis is derived from Item 3.b of the same definition.

On the other hand, suppose that R is α-closed in $\mathcal{D}_s$, *i.e.*, $(\mathrm{R}, \alpha) \notin \mathrm{Rg}^{-}_{\mathcal{D}_s}$. By Lines 1, 4, and 5 of Algorithm 4, we have that *(14)* $p^\star = \mathsf{bep}^{\overline{\alpha}}(\mathrm{R}, \mathsf{r})$ and *(15)* $\mathsf{r}^\star = \mathsf{pr} \uplus \mathsf{r}^{(\geq p^\star) \vee (\equiv_2 p^\star)}[\mathrm{R} \mapsto p^\star]$. By Points *(2)* and *(14)*, the definition of bep, and the fact that R is closed, it is not hard to see that *(16)* $p^\star > p$. Now, by Points *(15)* and *(2)*, it follows that *(17)* $\mathsf{r}^{\star(>p^\star)} = \mathsf{r}^{(>p^\star)}$, *(18)* $\mathsf{r}^{\star -1}(p^\star) = \mathsf{r}^{-1}(p^\star) \cup \mathrm{R}$, *(19)* $\mathsf{r}^{\star(<p^\star) \wedge (\not\equiv_2 p^\star)} \subseteq \mathsf{pr}^{(<p^\star) \wedge (\not\equiv_2 p^\star)}$. Hence, by Points *(1)*, *(16)*, and *(17)*, it holds that *(20)* $\mathsf{r}^\star$ is maximal above $p^\star$. Moreover, by Point *(18)*, we have that *(21)* $p^\star \in \mathsf{rng}(\mathsf{r}^\star)$. Finally, we have to prove that *(22)* $\mathsf{r}^\star$ is a region function, *i.e.*, $\mathsf{r}^{\star -1}(q) \cap \mathrm{Ps}_{\mathcal{D}^{\leq q}_{\mathsf{m}^\star}}$ is a β-region in the subgame $\mathcal{D}^{\leq q}_{\mathsf{m}^\star}$, for all priorities $q \in \mathsf{rng}(\mathsf{m}^\star)$ with $\beta \triangleq q \bmod 2$, where $\mathsf{m}^\star$ is the maximisation of $\mathsf{r}^\star$.

- If $q > p^\star$, by Points *(1)*, *(16)* and *(17)*, we have that $\mathsf{r}^{\star -1}(q) = \mathsf{r}^{-1}(q) \subseteq \mathrm{Ps}_{\partial^{\leq q}_{\mathsf{m}}} = \mathrm{Ps}_{\partial^{\leq q}_{\mathsf{m}^\star}}$. Therefore, the thesis immediately follows, since $\mathsf{r}^{-1}(q)$ is a β-region in the subgame $\partial^{\leq q}_{\mathsf{m}}$, where $\beta \triangleq q \bmod 2$.
- If $q = p^\star$, by Points *(1)*, *(2)*, *(16)* and *(18)*, we have that $\mathsf{r}^{\star -1}(q) \subseteq \mathrm{Ps}_{\partial^{\leq q}_{\mathsf{m}}}$. Now, by Point *(1)* and the fact that r is a region function, it follows that $\mathsf{r}^{-1}(q)$ is an α-region in $\partial^{\leq q}_{\mathsf{m}} = \partial^{\leq q}_{\mathsf{m}^\star}$. Moreover, due to Point *(14)*, R is an α-dominion in $\partial^{\leq q}_{\mathsf{m}} \setminus \mathsf{r}^{-1}(q)$. Therefore, due to Proposition 3.2, it holds that $\mathsf{r}^{\star -1}(q)$ is an α-region in $\partial^{\leq q}_{\mathsf{m}^\star}$.
- If $q < p^\star$ and $q \not\equiv_2 p^\star$, by Point *(19)*, we immediately have that $\mathsf{r}^{\star -1}(q) \cap \mathrm{Ps}_{\partial^{\leq q}_{\mathsf{m}^\star}}$ is a $\overline{\alpha}$-region in $\partial^{\leq q}_{\mathsf{m}^\star}$, since it only contains positions of priority q, which is the maximal one in the subgame $\partial^{\leq q}_{\mathsf{m}^\star}$.
- If $q < p^\star$ and $q \equiv_2 p^\star$, it can be easily shown that $\mathsf{r}^{\star -1}(q) \cap \mathrm{Ps}_{\partial^{\leq q}_{\mathsf{m}^\star}} = (\mathsf{r}^{-1}(q) \cap \mathrm{Ps}_{\partial^{\leq q}_{\mathsf{m}}} \setminus \mathrm{R}^\star) \cup \mathrm{P}$, for some α-maximal α-region $\mathrm{R}^\star$ in ∂_s and some set of positions $\mathrm{P} \subseteq \mathsf{pr}^{-1}(q)$ of priority q. Now, by applying Proposition 3.3, we have that $\mathsf{r}^{-1}(q) \cap \mathrm{Ps}_{\partial^{\leq q}_{\mathsf{m}}}$ is an α-region in $\partial^{\leq q}_{\mathsf{m}}$. Therefore, also $\mathsf{r}^{\star -1}(q) \cap \mathrm{Ps}_{\partial^{\leq q}_{\mathsf{m}^\star}}$ is an α-region in $\partial^{\leq q}_{\mathsf{m}^\star}$, since it only contains further positions of maximal priority q in the corresponding subgame.

Summing up, Points *(20)*, *(21)*, and *(22)* ensure that $(\mathsf{r}^\star, p^\star) \in \mathrm{S}$. At this point, as for the previous case, it remains to show that $(\mathsf{r}^\star, p^\star) \prec (\mathsf{r}, p)$. This fact easily follows from Item 3.a of Definition 3.4, where the priority q is set to $p^\star$, since, by Points *(2)* and *(18)*, we have that $\mathsf{r}^{-1}(p^\star) \cap \mathrm{R} = \emptyset$, so $\mathsf{r}^{-1}(p^\star) \subset \mathsf{r}^{-1}(p^\star) \cup \mathrm{R} = \mathsf{r}^{\star -1}(p^\star)$. □

Appendix B. Proof of Theorem 4.1

In this appendix we report the proof of Theorem 4.1 stated in Section 4, providing soundness and completenessof the DP search procedure.

Theorem 4.1 (DP Dominion Space)**.** *For a game ∂, the* DP structure $\mathcal{D} \triangleq \langle \partial, \mathcal{S}, \succ, \Re, \downarrow \rangle$*, where $\mathcal{S}$ is given in Definition 4.1, $\succ$ is the relation of Definition 4.2, and $\Re$ and $\downarrow$ are the functions computed by Algorithms 3 and 5, where in the former the assumption "***let** $(\mathsf{r}, p) = s$*" is replaced by "***let** $((\mathsf{r}, p), \_, \_) = s$*" is a dominion space.*

To prove Theorem 4.1, we have to show that the three components $\mathcal{S}$, $\Re$, and $\downarrow$ of the structure $\mathcal{D}$ satisfy the properties required by Definition 3.2 of dominion space. We do this through the Lemmas Appendix B.1, Appendix B.2, and Appendix B.3.

Lemma Appendix B.1 (State Space)**.** *The* DP *state space $\mathcal{S} = \langle \mathrm{S}, \top, \prec \rangle$ for a game $\partial \in \mathcal{P}$ is a well-founded partial order* w.r.t. $\prec$ *with designated element* $\top \in \mathrm{S}$.

Proof. Due to Definition 4.1 and Lemma Appendix A.1, to prove that $\mathcal{S}$ satisfies the required properties, we have only to observe that the distinguished element $\top \triangleq (\top^{\text{PP+}}, \varnothing, \emptyset)$ is a state. This immediately follows from the fact that $\top^{\text{PP+}} = (\mathsf{pr}, \mathsf{pr}(\partial))$ is a PP+ state and the empty supplementary function $\varnothing$ is trivially aligned with pr *w.r.t.* $\mathsf{pr}(\partial)$. □

Lemma Appendix B.2 (Query Function)**.** *The function $\Re$ is a query function,* i.e.*, for all states $s \in \mathrm{S}$, it holds that* (1) $\Re(s) \in \mathrm{QD}$ *and* (2) *if* $\Re(s) \in \mathrm{QD}^{-}$ *then* $s \succ \Re(s)$.

Proof. Due to Definition 4.2 and Lemma Appendix A.2, the thesis directly follows once observed that $\mathrm{R} = \mathsf{atr}^{\alpha}_{\Im_s}(\mathsf{r}^{-1}(p))$ independently from the fact that it is α-open in $\Im_s$ or α-locked *w.r.t.* s. □

Lemma Appendix B.3 (Successor Function)**.** *The function $\downarrow$ is a successor function,* i.e.*, for all states $s \in \mathrm{S}$ and quasi dominion pairs $(\mathrm{R}, \alpha) \in \mathrm{QD}^{-}$ with $s \succ (\mathrm{R}, \alpha)$, it holds that* (1) $s\downarrow(\mathrm{R}, \alpha) \in \mathrm{S}$ *and* (2) $s\downarrow(\mathrm{R}, \alpha) \prec s$.

Proof. Let $s \triangleq ((\mathsf{r}, p), \widetilde{\mathsf{r}}, \mathrm{P}) \in \mathrm{S}$ be a state, $(\mathrm{R}, \alpha) \in \mathrm{QD}^{-}$ an open quasi dominion pair in $\Im$ compatible with s, and $s^{\star} = ((\mathsf{r}^{\star}, p^{\star}), \widetilde{\mathsf{r}}^{\star}, \mathrm{P}^{\star}) \triangleq s\downarrow(\mathrm{R}, \alpha)$ the result obtained by computing the function $\downarrow$ on s and (R, α).

It is quite easy to see that $(\mathsf{r}^{\star}, p^{\star}) \in \mathrm{S}^{\mathrm{PP+}}$ and $(\mathsf{r}^{\star}, p^{\star}) \prec^{\mathrm{PP+}} (\mathsf{r}, p)$. Indeed, in case one of the two macros $\#Assignment(\xi)$ with $\xi \in \{\mathtt{f}, \mathtt{t}\}$ and $\#InstantPromotion$ of Algorithm 5 is executed, one can derive the thesis by applying exactly the same reasoning used in the proof of Lemma Appendix A.3, since the instructions concerning the two components $\mathsf{r}^{\star}$ and $p^{\star}$ are those used in Algorithm 4. If the $\#DelayedPromotion$ macro is considered, instead, due to Item 1 of Definition 4.1, $\widetilde{\mathsf{r}}$ is aligned with r *w.r.t.* p, so, the set of positions $\mathsf{r}^{-1}(q) \cup \widetilde{\mathsf{r}}^{-1}(q)$ is a β-region in $\Im^{\leq q}_{\mathsf{r}}$, for all $q \in \mathsf{rng}(\widetilde{\mathsf{r}})$ with $\beta \equiv_2 q \equiv_2 p^{\star}$. Hence, by applying the same reasoning used for the $\#InstantPromotion$ macro, the thesis follows in this case as well.

To conclude the proof, we need to show that $\widetilde{\mathsf{r}}^{\star}$ is aligned with $\mathsf{r}^{\star}$ *w.r.t.* $p^{\star}$. Again, we do this by means of a case analysis on the four possible macros.

- $\#Assignment(\mathtt{f})$. Lines 2 and 3 of the macro ensure that $p^{\star} < p$ and $\widetilde{\mathsf{r}}^{\star} = \widetilde{\mathsf{r}}$. Hence, the property follows from the fact that $\widetilde{\mathsf{r}}$ is aligned with r *w.r.t.* p.

- $\#Assignment(\mathtt{t})$. By Lines 2 and 3, $p^{\star} < p$ and $\widetilde{\mathsf{r}}^{\star} = \widetilde{\mathsf{r}}[\mathrm{R} \mapsto q]$. Since $\widetilde{\mathsf{r}}$ is aligned with r *w.r.t.* p, we have that $\widetilde{\mathsf{r}}^{-1}(z) \subseteq \mathsf{dom}\big(\mathsf{r}^{(>p)\wedge(<z)\wedge(\equiv_2 z)}\big)$, and $\mathsf{r}^{-1}(z) \cup \widetilde{\mathsf{r}}^{-1}(z)$ is an α-region in $\Im^{\leq z}_{\mathsf{r}}$, for all priorities $z \in \mathsf{rng}(\widetilde{\mathsf{r}})$. Now, if $z \neq q \triangleq \mathsf{bep}^{\overline{\alpha}}(\mathrm{R}, \mathsf{r} \uplus \widetilde{\mathsf{r}})$, we have $\widetilde{\mathsf{r}}^{\star -1}(z) = \widetilde{\mathsf{r}}^{-1}(z)$. So, the required properties on $\widetilde{\mathsf{r}}^{\star -1}(z)$ are immediately satisfied. If $z = q$, instead, the α-region R is added to both r and $\widetilde{\mathsf{r}}$ with measures p and z, respectively. Hence, $\widetilde{\mathsf{r}}^{\star -1}(z) \subseteq \mathsf{dom}\big(\mathsf{r}^{\star(>p^{\star})\wedge(<z)\wedge(\equiv_2 z)}\big)$ is verified, since $p \equiv_2 z$. Finally, we need to show that $\mathsf{r}^{\star -1}(z) \cup \widetilde{\mathsf{r}}^{\star -1}(z)$ is an α-region in $\Im^{\leq z}_{\mathsf{r}^{\star}}$. This follows from Proposition 3.2 applied to $\mathsf{r}^{-1}(z) \cup \widetilde{\mathsf{r}}^{-1}(z)$ and R, since R is an α-dominion in the subgame $\Im^{\leq z}_{\mathsf{r}^{\star}} \setminus (\mathsf{r}^{-1}(z) \cup \widetilde{\mathsf{r}}^{-1}(z))$, due to the fact that $\widetilde{\mathsf{r}}^{-1}(z) \subseteq \mathsf{dom}\big(\mathsf{r}^{(>p)\wedge(<z)\wedge(\equiv_2 z)}\big)$.

- $\#DelayedPromotion$. The property is trivially satisfied, since $\widetilde{\mathsf{r}}^{\star} = \varnothing$.

- $\#InstantPromotion$. Since $\widetilde{\mathsf{r}}^{\star(\leq p^{\star})} = \varnothing$ and $\widetilde{\mathsf{r}}^{\star(>p^{\star})} = \widetilde{\mathsf{r}}^{(>p^{\star})}$, the property follows from the fact that $\widetilde{\mathsf{r}}$ is aligned with r *w.r.t.* p.

□

References

[1] M. Benerecetti, D. Dell'Erba, F. Mogavero, A Delayed Promotion Policy for Parity Games., in: Games, Automata, Logics, and Formal Verification16, EPTCS 226, 2016, pp. 30–45.

[2] K. Apt, E. Grädel, Lectures in Game Theory for Computer Scientists., Cambridge University Press, 2011.

[3] A. Mostowski, Games with Forbidden Positions., Tech. rep., University of Gdańsk, Gdańsk, Poland (1991).

[4] E. Emerson, C. Jutla, The Complexity of Tree Automata and Logics of Programs (Extended Abstract)., in: Foundation of Computer Science'88, IEEE Computer Society, 1988, pp. 328–337.

[5] E. Emerson, C. Jutla, A. Sistla, On Model Checking for the muCalculus and its Fragments., in: Computer Aided Verification'93, LNCS 697, Springer, 1993, pp. 385–396.

[6] O. Kupferman, M. Vardi, P. Wolper, An Automata Theoretic Approach to Branching-Time Model Checking., Journal of the ACM 47 (2) (2000) 312–360.

[7] R. Alur, T. Henzinger, O. Kupferman, Alternating-Time Temporal Logic., Journal of the ACM 49 (5) (2002) 672–713.

[8] S. Schewe, An Optimal Strategy Improvement Algorithm for Solving Parity and Payoff Games., in: Computer Science Logic'08, LNCS 5213, Springer, 2008, pp. 369–384.

[9] F. Mogavero, A. Murano, M. Vardi, Relentful Strategic Reasoning in Alternating-Time Temporal Logic., in: Logic for Programming Artificial Intelligence and Reasoning'10, LNAI 6355, Springer, 2010, pp. 371–387.

[10] F. Mogavero, A. Murano, G. Perelli, M. Vardi, What Makes ATL* Decidable? A Decidable Fragment of Strategy Logic., in: Concurrency Theory'12, LNCS 7454, Springer, 2012, pp. 193–208.

[11] F. Mogavero, A. Murano, G. Perelli, M. Vardi, Reasoning About Strategies: On the Model-Checking Problem., Transactions On Computational Logic 15 (4) (2014) 34:1–42.

[12] F. Mogavero, A. Murano, G. Perelli, M. Vardi, Reasoning About Strategies: On the Satisfiability Problem., Logical Methods in Computer Science 13 (1:9) (2017) 1–37.

[13] M. Benerecetti, F. Mogavero, A. Murano, Substructure Temporal Logic., in: Logic in Computer Science'13, IEEE Computer Society, 2013, pp. 368–377.

[14] M. Benerecetti, F. Mogavero, A. Murano, Reasoning About Substructures and Games., Transactions On Computational Logic 16 (3) (2015) 25:1–46.

[15] A. Mostowski, Regular Expressions for Infinite Trees and a Standard Form of Automata., in: Symposium on Computation Theory'84, LNCS 208, Springer, 1984, pp. 157–168.

[16] E. Emerson, C. Jutla, Tree Automata, muCalculus, and Determinacy., in: Foundation of Computer Science'91, IEEE Computer Society, 1991, pp. 368–377.

[17] O. Kupferman, M. Vardi, Weak Alternating Automata and Tree Automata Emptiness., in: Symposium on Theory of Computing'98, Association for Computing Machinery, 1998, pp. 224–233.

[18] E. Grädel, W. Thomas, T. Wilke, Automata, Logics, and Infinite Games: A Guide to Current Research., LNCS 2500, Springer, 2002.

[19] A. Ehrenfeucht, J. Mycielski, Positional Strategies for Mean Payoff Games., International Journal of Game Theory 8 (2).

[20] V. Gurvich, A. Karzanov, L. Khachivan, Cyclic Games and an Algorithm to Find Minimax Cycle Means in Directed Graphs., USSR Computational Mathematics and Mathematical Physics 28 (5) (1990) 85–91.

[21] U. Zwick, M. Paterson, The Complexity of Mean Payoff Games on Graphs., Theoretical Computer Science 158 (1-2) (1996) 343–359.

[22] A. Condon, The Complexity of Stochastic Games., Information and Computation 96 (2) (1992) 203–224.

[23] K. Chatterjee, L. Doyen, T. Henzinger, J.-F. Raskin, Generalized Mean-Payoff and Energy Games., in: Foundations of Software Technology and Theoretical Computer Science'10, LIPIcs 8, Leibniz-Zentrum fuer Informatik, 2010, pp. 505–516.

[24] F. Horn, W. Thomas, N. Wallmeier, Optimal Strategy Synthesis in Request-Response Games., in: Automated Technology for Verification and Analysis'08, LNCS 5311, Springer, 2008, pp. 361–373.

[25] K. Chatterjee, T. Henzinger, F. Horn, Finitary Winning in omega-Regular Games., Transactions On Computational Logic 11 (1) (2010) 1:1–26.

[26] N. Fijalkow, M. Zimmermann, Cost-Parity and Cost-Streett Games., Logical Methods in Computer Science 10 (2) (2014) 1–29.

[27] F. Mogavero, A. Murano, L. Sorrentino, On Promptness in Parity Games., Fundamenta Informaticae 139 (3) (2015) 277–305.

[28] M. Jurdziński, Deciding the Winner in Parity Games is in UP $\cap$ co-UP., Information Processing Letters 68 (3) (1998) 119–124.

[29] W. Zielonka, Infinite Games on Finitely Coloured Graphs with Applications to Automata on Infinite Trees., Theoretical Computer Science 200 (1-2) (1998) 135–183.

[30] M. Jurdziński, M. Paterson, U. Zwick, A Deterministic Subexponential Algorithm for Solving Parity Games., SIAM Journal on Computing 38 (4) (2008) 1519–1532.

[31] S. Schewe, Solving Parity Games in Big Steps., in: Foundations of Software Technology and Theoretical Computer Science'07, LNCS 4855, Springer, 2007, pp. 449–460.

[32] M. Jurdziński, Small Progress Measures for Solving Parity Games., in: Symposium on Theoretical Aspects of Computer Science'00, LNCS 1770, Springer, 2000, pp. 290–301.

[33] J. Vöge, M. Jurdziński, A Discrete Strategy Improvement Algorithm for Solving Parity Games., in: Computer Aided Verification'00, LNCS 1855, Springer, 2000, pp. 202–215.

[34] S. Schewe, A. Trivedi, T. Varghese, Symmetric Strategy Improvement., in: International Colloquium on Automata, Languages, and Programming'15, LNCS 9135, Springer, 2015, pp. 388–400.

[35] C. Calude, S. Jain, B. Khoussainov, W. Li, F. Stephan, Deciding Parity Games in Quasipolynomial Time., in: Symposium on Theory of Computing'17, Association for Computing Machinery, 2017, accepted for publication.

[36] J. Fearnley, S. Jain, S. Schewe, F. Stephan, D. Wojtczak, An Ordered Approach to Solving Parity Games in Quasi Polynomial Time and Quasi Linear Space., in: SPIN Symposium on Model Checking of Software'2017, Association for Computing Machinery, 2017, accepted for publication.

[37] M. Jurdziński, R. Lazic, Succinct Progress Measures for Solving Parity Games., in: Logic in Computer Science'17, Association for Computing Machinery, 2017, accepted for publication.

[38] M. Benerecetti, D. Dell'Erba, F. Mogavero, Solving Parity Games via Priority Promotion., in: Computer Aided Verification'16, LNCS 9780 (Part II), Springer, 2016, pp. 270–290.

[39] M. Benerecetti, D. Dell'Erba, F. Mogavero, Solving Parity Games via Priority Promotion., Formal Methods in System Design. To appear.

[40] T. van Dijk, Attracting Tangles to Solve Parity Games., in: Computer Aided Verification'18, 2018, accepted for publication at CAV 2018, Oxford, UK, July 14-17 2018.

[41] C. Stirling, Modal and Temporal Properties of Processes., Texts in Computer Science., Springer, 2001.

[42] O. Friedmann, M. Lange, Solving Parity Games in Practice., in: Automated Technology for Verification and Analysis'09, LNCS 5799, Springer, 2009, pp. 182–196.

[43] M. Benerecetti, D. Dell'Erba, F. Mogavero, Robust Exponential Worst Cases for Divide-et-Impera Algorithms for Parity Games., in: Games, Automata, Logics, and Formal Verification17, EPTCS 256, 2017, pp. 121–135.

[44] T. van Dijk, Oink: an Implementation and Evaluation of Modern Parity Game Solvers. (2018) 291–308.