A Delayed Promotion Policy for Parity Games

Massimo Benerecetti
Università degli Studi di Napoli Federico II

Daniele Dell'Erba
Università degli Studi di Napoli Federico II

Fabio Mogavero
Oxford University

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 behaviours exhibited by the original method in the worst case. The resulting procedure not only often outperforms the original priority promotion approach, but so far no exponential worst case is known.

1 Introduction

The abstract concept of *game* has proved to be a fruitful metaphor in theoretical computer science [1]. 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* [18], a simple two-player turn-based perfect-information game played on directed graphs, whose nodes are labelled 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 synthesis, where it is used as algorithmic back-end of satisfiability and model-checking procedures for temporal logics [6, 8, 16], and as a core for several techniques employed in automata theory [7, 10, 15, 17]. In particular, it has been proved to be linear-time interreducible with the model-checking problem for the *modal* μCALCULUS [8] and it is closely related to other games of infinite duration, as *mean payoff* [5, 11], *discounted payoff* [24], *simple stochastic* [4], and *energy* [3] games. 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 [12]. That result improves the NPTIME $\cap$ CONPTIME membership [8], which easily follows from the property of *memoryless determinacy* [7, 18]. 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* [23] and its *dominion decomposition* [14] and *big step* [19] 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 [13] and the *strategy improvement* approaches [20–22].

Recently, a new *divide et impera* solution algorithm, called *priority promotion* (PP, for short), has been proposed in [2], 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 [14, 19] 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 priority corresponding to an under-approximation of the best value 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. 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 know 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 $O(n \cdot \log k)$ space against the $O(k \cdot n \cdot \log n)$ required by Jurdziński's approach [13], where n and k are, respectively, the numbers of vertexes and priorities of the game. Unfortunately, the PP algorithm also exhibits exponential behaviours 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 paper, 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. For the resulting algorithm no exponential worst case has been found. Experiments on randomly generated games also show that the new approach performs much better than PP in practice, while still preserving the same space complexity.

2 Preliminaries

Let us first briefly recall the notation and basic definitions concerning parity games that expert readers can simply skip. We refer to [1] [23] for a comprehensive presentation of the subject.

Given a partial function $f : A \rightharpoonup B$, by $dom(f) \subseteq A$ and $rng(f) \subseteq B$ we indicate the domain and range of f, respectively. In addition, $\uplus$ denotes the *completion operator* that, taken f and another partial function $g : A \rightharpoonup B$, returns the partial function $f \uplus g \triangleq (f \setminus dom(g)) \cup g : A \rightharpoonup B$, 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 $\mathscr{A} = \langle \text{Ps}^0, \text{Ps}^1, Mv \rangle$, with $\text{Ps}^0 \cap \text{Ps}^1 = \emptyset$ and $\text{Ps} \triangleq \text{Ps}^0 \cup \text{Ps}^1$, such that $\langle \text{Ps}, Mv \rangle$ is a finite directed graph. Ps^0 (*resp.*, Ps^1) is the set of positions of player 0 (*resp.*, 1) and $Mv \subseteq \text{Ps} \times \text{Ps}$ is a left-total relation describing all possible moves. A *path* in $V \subseteq \text{Ps}$ is a finite or infinite sequence $\pi \in \text{Pth}(V)$ 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 $\text{lst}(\pi)$ we denote the last position of π. A positional *strategy* for player $\alpha \in \{0, 1\}$ on $V \subseteq \text{Ps}$ is a partial function $\sigma_\alpha \in \text{Str}^\alpha(V) \subseteq (V \cap \text{Ps}^\alpha) \rightharpoonup V$, mapping each α-position $v \in \text{dom}(\sigma_\alpha)$ to position $\sigma_\alpha(v)$ compatible with the move relation, *i.e.*, $(v, \sigma_\alpha(v)) \in Mv$. With $\text{Str}^\alpha(V)$ we denote the set of all α-strategies on V. A *play* in $V \subseteq \text{Ps}$ from a position $v \in V$ *w.r.t.* a pair of strategies $(\sigma_0, \sigma_1) \in \text{Str}^0(V) \times \text{Str}^1(V)$, called $((\sigma_0, \sigma_1), v)$-*play*, is a path $\pi \in \text{Pth}(V)$ such that $\pi_0 = v$ and, for all $i \in [0, |\pi| - 1[$, if $\pi_i \in \text{Ps}^0$ then $\pi_{i+1} = \sigma^0(\pi_i)$ else $\pi_{i+1} = \sigma^1(\pi_i)$. The *play function* $\text{play} : (\text{Str}^0(V) \times \text{Str}^1(V)) \times V \to \text{Pth}(V)$ returns, for each position $v \in V$ and pair of strategies $(\sigma_0, \sigma_1) \in \text{Str}^0(V) \times \text{Str}^1(V)$, the maximal $((\sigma_0, \sigma_1), v)$-play $\text{play}((\sigma^0, \sigma^1), v)$.

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

The α-predecessor of V, in symbols $\text{pre}^\alpha(V) \triangleq \{v \in \text{Ps}^\alpha : Mv(v) \cap V \neq \emptyset\} \cup \{v \in \text{Ps}^{\overline{\alpha}} : Mv(v) \subseteq V\}$, collects the positions from which player α can force the game to reach some position in V with a single move. The α-attractor $\text{atr}^\alpha(V)$ generalises the notion of α-predecessor $\text{pre}^\alpha(V)$ to an arbitrary number of moves, and corresponds to the least fix-point of that operator. When $V = \text{atr}^\alpha(V)$, 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 $\eth \setminus V$ is precisely $\text{Ps} \setminus V$. Finally, the set $\text{esc}^\alpha(V) \triangleq \text{pre}^\alpha(\text{Ps} \setminus V) \cap V$, 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 $\text{int}^\alpha(V) \triangleq (V \cap \text{Ps}^\alpha) \setminus \text{esc}^\alpha(V)$, contains, instead, the α-positions from which α cannot escape with a single move. All the operators and sets above actually depend on the specific game $\eth$ they are applied in. In the rest of the paper, we shall only add $\eth$ as subscript of an operator, *e.g.*, $\text{esc}^\alpha_\eth(V)$, when the game is not clear from the context.

3   The Priority Promotion Approach

The priority promotion approach proposed in [2] attacks the problem of solving a parity game $\eth$ by computing one of its dominions D, for some player $\alpha \in \{0, 1\}$, at a time. Indeed, once the α-attractor $D^\star$ of D is removed from $\eth$, the smaller game $\eth \setminus D^\star$ 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 to that of iteratively finding its dominions [14].

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 [2])**.** *Let* $\supset \in \mathrm{PG}$ *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* $\supset$ *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* $\mathrm{int}^\alpha(\mathrm{Q}) \subseteq \mathrm{dom}(\sigma_{\overline{\alpha}})$, *and positions* $v \in \mathrm{Q}$, *the induced play* $\pi = \mathrm{play}((\sigma_0, \sigma_1), v)$ *satisfies* $\mathrm{pr}(\pi) \equiv_2 \alpha$, *if* π *is infinite, and* $\mathrm{lst}(\pi) \in \mathrm{esc}^{\overline{\alpha}}(\mathrm{Q})$, *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.*, $\mathrm{esc}^{\overline{\alpha}}(\mathrm{Q}) = \emptyset$, 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 $\mathrm{esc}^{\overline{\alpha}}(\mathrm{Q}) \neq \emptyset$, need not 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.

The priority promotion algorithm explores a partial order, whose elements, called *states*, record information about the open quasi dominions computed along the way. The initial state of the search is the top element of the order, where the quasi dominions are initialised 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 $\mathfrak{R}$, 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 1 implements the do-

---

Algorithm 1: The Searcher. [2]

signature $\mathrm{src}_{\mathscr{D}} : \mathrm{S}_{\mathscr{D}} \to \mathrm{QD}^+_{\supset_{\mathscr{D}}}$

function $\mathrm{src}_{\mathscr{D}}(s)$

1    $(\mathrm{Q}, \alpha) \leftarrow \mathfrak{R}_{\mathscr{D}}(s)$

2   **if** $(\mathrm{Q}, \alpha) \in \mathrm{QD}^+_{\supset_{\mathscr{D}}}$ **then**

3 | **return** (Q, α)

   else

4 | **return** $\mathrm{src}_{\mathscr{D}}(s \downarrow_{\mathscr{D}} (\mathrm{Q}, \alpha))$

---

minion search procedure $\mathrm{src}_{\mathscr{D}}$. 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 operator and the compatibility relation, forms what is called a *dominion space*.

Definition 3.2 (Dominion Space [2])**.** *A dominion space for a game* $\supset \in \mathrm{PG}$ *is a tuple* $\mathscr{D} \triangleq \langle \supset, \mathscr{S}, \succ, \mathfrak{R}, \downarrow \rangle$, *where* (1) $\mathscr{S} \triangleq \langle \mathrm{S}, \top, \prec \rangle$ *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) $\mathfrak{R} : \mathrm{S} \to \mathrm{QD}$ *is the* query operator *mapping each element* $s \in \mathrm{S}$ *to a quasi dominion pair* $(\mathrm{Q}, \alpha) \triangleq \mathfrak{R}(s) \in \mathrm{QD}$ *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 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 [2], indeed, it is shown that the search procedure $\mathrm{src}_{\mathscr{D}}$ is sound and complete on any dominion space $\mathscr{D}$. 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 [2]. Here we propose a different one, starting from a slight optimisation, 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 domains previously computed at higher priorities, is considered. At each priority of parity α, a quasi α-domain 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 position 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 $\eth$ is a special form of quasi α-dominion of $\eth$ with the additional requirement that all the escape positions in $\mathsf{esc}^{\overline{\alpha}}(\mathrm{R})$ have the maximal priority $p \triangleq \mathsf{pr}(\eth) \equiv_2 \alpha$ in $\eth$. In this case, we say that α-region R has priority p. As a consequence, if the opponent $\overline{\alpha}$ can escape from the α-region R, it must visit a position with the highest priority in it, which is of parity α.

Definition 3.3 (Region [2])**.** *A quasi α-dominion R is an α-region in $\eth$ if* $\mathsf{pr}(\eth) \equiv_2 \alpha$ *and all the positions in* $\mathsf{esc}^{\overline{\alpha}}(\mathrm{R})$ *have priority* $\mathsf{pr}(\eth)$, *i.e.* $\mathsf{esc}^{\overline{\alpha}}(\mathrm{R}) \subseteq \mathsf{pr}^{-1}(\mathsf{pr}(\eth))$.

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 [2])**.** *Let $\eth \in \mathrm{PG}$ be a game and $\mathrm{R} \subseteq \mathrm{Ps}$ an α-region in $\eth$. Then, $\mathrm{R}^{\star} \triangleq \mathsf{atr}^{\alpha}(\mathrm{R})$ is an α-maximal α-region in $\eth$.*

A solution to the second problem, the merging operation, is obtained as follows. Given an α-region R in some game $\eth$ and an α-dominion D in a subgame of $\eth$ 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 [2])**.** *Let $\eth \in \mathrm{PG}$ be a game, $\mathrm{R} \subseteq \mathrm{Ps}$ an α-region, and $\mathrm{D} \subseteq \mathrm{Ps}_{\eth \setminus \mathrm{R}}$ an α-dominion in the subgame $\eth \setminus \mathrm{R}$. Then, $\mathrm{R}^{\star} \triangleq \mathrm{R} \cup \mathrm{D}$ is an α-region in $\eth$. Moreover, if both R and D are α-maximal in $\eth$ and $\eth \setminus \mathrm{R}$, respectively, then $\mathrm{R}^{\star}$ is α-maximal in $\eth$ 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 [2] 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* $\partial^\star \in \mathrm{PG}$ *be a game and* $\mathrm{R}^\star \subseteq \mathrm{Ps}_{\partial^\star}$ *an* α-*maximal* α-*region in* $\partial^\star$. *For any subgame* ∂ *of* $\partial^\star$ *and* α-*region* $\mathrm{R} \subseteq \mathrm{Ps}$ *in* ∂, *if* $\mathrm{R}^\natural \triangleq \mathrm{R} \setminus \mathrm{R}^\star \neq \emptyset$, *then* $\mathrm{R}^\natural$ *is an* α-*region in* $\partial \setminus \mathrm{R}^\star$.

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.

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 {a} of measure 6, which is open because of the two moves leading to d and e. The open 1-region {b,f,h} of measure 5 is, then, formed by attracting both f and h to b, which is open in the subgame where {a} is removed. Similarly, the 0-region {c,j} of measure 4 and the 1-region {d} of measure 3 are open, once removed {a,b,f,h} and {a,b,c,f,h}, respectively, from the game. At priority 2, the 0-region {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 {e} to 4 is performed, resulting in the new 0-region {c,e,j} that resets 1-region {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 {d} of measure 3 is recomputed and, then, priority 1 is processed to build the 1-region {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 {i} to priority 6 is performed, however, 0-region {c,e,j} of measure 4 is not reset. This leads directly to the configuration in Column 6, after the maximisation of 0-region 6, which attracts b, d, g, and j. Notice that, as prescribed by Proposition 3.3, the set {c,e}, highlighted by the grey area, is still a 0-region. On the other hand, the set {f,h}, highlighted by the dashed line and originally included in 1-region {b,d,f,g,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 {a,b,c,d,e,g,i,j}, which is a dominion for player 0.

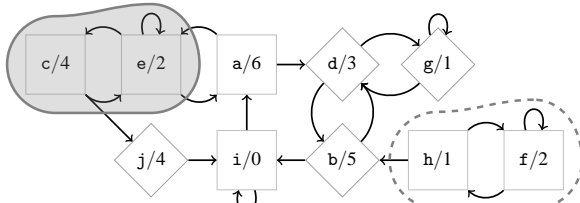**Figure 1:** Running example.

		1	2	3	4	5	6
6		a↓	⋯	⋯	⋯	⋯	a,b,d,g,i,j↓
5		b,f,h↓	⋯	⋯	b,d,f,g,h↓	⋯	
4		c,j↓	c,e,j↓	⋯	c,j↓	c,e,j↓	c,e↑$_6$
3		d↓	d↓	d,g↑$_5$			
2		e↑$_4$			e↑$_4$		
1			g↑$_3$				
0						i↑$_6$	

Table 1: PP+ simulation.

We can now provide the formal account of the PP+ dominion space. We shall denote with Rg the set of region pairs in ∂ 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 $\mathsf{r} \in \Delta \triangleq$

Ps $\to$ Pr, called *region function*. Given a priority $p \in$ 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, $\mathsf{r}^{(\sim p)} \triangleq \mathsf{r}{\upharpoonright}\{v \in$ Ps $: \mathsf{r}(v) \sim p\}$, for $\sim \in \{\geq, >, <, \equiv_2\}$. By $\eth_\mathsf{r}^{\leq p} \triangleq \eth \setminus \mathrm{dom}\left(\mathsf{r}^{(>p)}\right)$, we denote the largest subgame obtained by removing from $\eth$ all the positions in the domain of $\mathsf{r}^{(>p)}$. The *maximisation* of a priority function $\mathsf{r} \in \Delta$ is the unique priority function $\mathsf{m} \in \Delta$ such that $\mathsf{m}^{-1}(q) = \mathrm{atr}_{\eth_\mathsf{m}^{\leq q}}^\alpha \left(\mathsf{r}^{-1}(q) \cap \mathrm{Ps}_{\eth_\mathsf{m}^{\leq q}}\right)$, for all priorities $q \in \mathrm{rng}(\mathsf{r})$ with $\alpha \triangleq q \bmod 2$. In addition, we say that r is *maximal* above $p \in$ 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 maximisation 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 maximisations m. Formally, $\mathsf{r} \in \mathbb{R} \subseteq \Delta$ is a *region function* iff, for all priorities $q \in \mathrm{rng}(\mathsf{m})$ with $\alpha \triangleq q \bmod 2$, it holds that $\mathsf{r}^{-1}(q) \cap \mathrm{Ps}_{\eth_\mathsf{m}^{\leq q}}$ is an α-region in the subgame $\eth_\mathsf{m}^{\leq q}$, where m is the maximisation 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 $\eth$. Initially, r coincides with the priority function pr of the entire game $\eth$, while p is set to the maximal priority $\mathrm{pr}(\eth)$ available in the game. To each of such states $s \triangleq (\mathsf{r}, p)$, we then associate the *subgame at s* defined as $\eth_s \triangleq \eth_\mathsf{r}^{\leq p}$, 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 $\mathscr{S} \triangleq \langle \mathrm{S}, \top, \prec \rangle$, 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 $p \in$ Pr such that* (a) *r is maximal above p and* (b) *$p \in \mathrm{rng}(\mathsf{r})$;*

2. $\top \triangleq (\mathsf{pr}, \mathsf{pr}(\eth))$;

3. *two states $s_1 \triangleq (\mathsf{r}_1, p_1), s_2 \triangleq (\mathsf{r}_2, p_2) \in$ 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 $q \in \mathrm{rng}(\mathsf{r}_1)$ 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 $\mathsf{r}^{-1}(q) \subseteq \mathrm{Ps}_{\eth_\mathsf{r}^{\leq q}}$. 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 $\eth_s$. 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 $s \triangleq (\mathsf{r}, p) \in$ S, in symbols $s \succ (\mathrm{R}, \alpha)$, iff* (1) *$(\mathrm{R}, \alpha) \in \mathrm{Rg}_{\eth_s}$ and* (2) *if R is α-open in $\eth_s$ then $\mathrm{R} = \mathrm{atr}_{\eth_s}^\alpha(\mathsf{r}^{-1}(p))$.*

Algorithm 2 provides the implementation for the query function compatible with the priority-promotion mechanism. Line 1 simply computes the parity α of the priority to process in the state $s \triangleq (r, p)$. Line 2, instead, computes the attractor *w.r.t.* player α in subgame $\eth_s$ 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 $\eth_s$ containing $r^{-1}(p)$.

Algorithm 2: Query Function.
signature $\mathfrak{R} : \mathrm{S} \to 2^{\mathrm{Ps}} \times \{0,1\}$
function $\mathfrak{R}(s)$
let $(r, p) = s$ **in**
1       $\alpha \leftarrow p \bmod 2$
2       $\mathrm{R} \leftarrow \mathrm{atr}^{\alpha}_{\eth_s}(r^{-1}(p))$
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 $I \triangleq Mv \cap ((\mathrm{R} \cap \mathrm{Ps}^{\overline{\alpha}}) \times (\mathrm{dom}(r) \backslash \mathrm{R}))$ 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, $\mathrm{bep}^{\overline{\alpha}}(\mathrm{R}, r)$ is set to the minimal measure of those regions that contain positions reachable by a move in I. Formally, $\mathrm{bep}^{\overline{\alpha}}(\mathrm{R}, r) \triangleq \min(\mathrm{rng}(r \upharpoonright \mathrm{rng}(I)))$. 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 $\eth_s$, then $\mathrm{bep}^{\overline{\alpha}}(\mathrm{R}, r)$ 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 $\mathrm{R} = \{\mathrm{e}\}$ with measure 2 in Column 1 of Figure 1, we have that $I = \{(\mathrm{e}, \mathrm{a}), (\mathrm{e}, \mathrm{c})\}$ and $r \upharpoonright \mathrm{rng}(I) = \{(\mathrm{a}, 6), (\mathrm{c}, 4)\}$. Hence, $\mathrm{bep}^1(\mathrm{R}, r) = 4$.

Algorithm 3 reports the pseudo-code of the successor function, which differs from the one proposed in [2] 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 $s^{\star} \triangleq (r^{\star}, p^{\star})$ in the dominion space. It first checks whether R is open also in the subgame $\eth_s$ (Line 1). If this is the case, it assigns the measure p to region R and stores it in the new region function $r^{\star}$ (Line 2). The new current priority $p^{\star}$ is, then, computed as the highest priority lower than p in $r^{\star}$ (Line 3). If, on the other hand,

Algorithm 3: Successor Function.
signature $\downarrow\, :\, \succ\, \to\, \Delta \times \mathrm{Pr}$
function $s \downarrow (\mathrm{R}, \alpha)$
let $(r, p) = s$ **in**
1   **if** $(\mathrm{R}, \alpha) \in \mathrm{Rg}^{-}_{\eth_s}$ **then**
2         $r^{\star} \leftarrow r[\mathrm{R} \mapsto p]$
3         $p^{\star} \leftarrow \max(\mathrm{rng}(r^{\star(<p)}))$
else
4         $p^{\star} \leftarrow \mathrm{bep}^{\overline{\alpha}}(\mathrm{R}, r)$
5         $r^{\star} \leftarrow \mathrm{pr} \uplus r^{(\geq p^{\star}) \vee (\equiv_2 p^{\star})}[\mathrm{R} \mapsto p^{\star}]$
6   **return** $(r^{\star}, p^{\star})$

R is closed in $\eth_s$, 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 $\eth$ *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 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 3 is a dominion space.

Theorem 3.1 (PP+ Dominion Space)**.** *For a game $\eth$, the PP+ structure $\mathscr{D} \triangleq \langle \eth, \mathscr{S}, \succ, \mathfrak{R}, \downarrow \rangle$, where $\mathscr{S}$ is given in Definition 3.4, $\succ$ is the relation of Definition 3.5, and $\mathfrak{R}$ and $\downarrow$ are the functions computed by Algorithms 2 and 3 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 [2] 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 $\eth_h$ 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 $F(1) = 2$.

The search procedure on ∂_4 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 corre-

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

sponds 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 configuration 1000.

An estimate of the depth of the PP+ dominion space on the game ∂_h 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 $\mathscr{D}_h^{\text{PP+}}$ with $k = 2h + 1$ positions and priorities, whose execution depth is* $\text{Fib}(2(h+4))/\text{Fib}(h+4) - (h+6) = \Theta\left(((1+\sqrt{5})/2)^{k/2}\right)$.

4   Delayed Promotion Policy

At the beginning of the previous section, we have observed that the time complexity of the dominion search procedure $\text{src}_{\mathscr{D}}$ linearly depends on the execution depth of the underlying dominion space $\mathscr{D}$. 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 behaviours 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 suffices in avoiding some pathological cases and we do not have any finer criteria 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 in advance to minimise the necessity of application
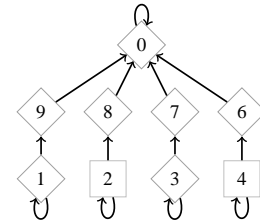
of the reset mechanism. The new solution procedure is based on delaying the promotions of regions, called *locked promotions*, that require the reset of previously performed promotions of the opponent parity, until a complete knowledge of the current search phase is reached. Once only locked promotions are left, the search phase terminates by choosing the highest measure $p^\star$ among those associated with the locked promotions and 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 $\widetilde{r}$ 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 $\widetilde{r}$, by assigning to R in $\widetilde{r}$ 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 $\widetilde{r}$ is selected and all promotions of the same parity are applied at once. This is done by first moving all regions from $\widetilde{r}$ 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 $\widetilde{r}$, 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 $\widetilde{r}$ 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 same parity are preserved. If this was not the case, we would have no criteria to determine which promotions need to be locked and which, instead, can be freely performed.

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 $\{g\}$ 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 $\{d,g\}$ could be promoted to 5. However, since its target is greater than $4 \in$ P, it is delayed and recorded in $\widetilde{r}$, where it is assigned priority 5. At priority 0, a delayed promotion of 0-region $\{i\}$ to priority 6 is encountered and registered,

	1	2	3	4
6	a↓	...	...	a,b,d,g,i,j↓
5	b,f,h↓	...	...	
4	c,j↓	c,e,j↓	...	c,e↑$_6$
3	d↓	d↓	d,g $\not\uparrow_5$	
2	e↑$_4$			
1		g↑$_3$		
0			i↑$_6$	

Table 2: DP simulation.

since it would overtake priority $3 \in$ 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 $\{i\}$ is promoted and both the auxiliary priority function $\widetilde{r}$ and the set of performed promotions P are emptied. Previously computed 0-region $\{c,e,j\}$ has the same parity and, therefore, it is not reset, while the positions in both 1-regions $\{b,f,h\}$ and $\{d,g\}$ are reset to their original priorities. After maximisation of the newly created 0-region $\{a,i\}$, positions b, d, g, and j get attracted as well. This leads to the first cell of Column 4, where 0-region $\{a,b,d,g,i,j\}$ is open. The next priority to process is then 4, where 0-region $\{c,e\}$, the previous 0-region $\{c,e,j\}$ purged of position 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 $\{d,g\}$ allowed a reduction in the number of operations. Indeed, the redundant maximisation of 1-region $\{b,f,h\}$ is avoided.

It is worth noting that this procedure only requires a linear number of promotions, precisely $\left\lfloor\frac{h+1}{2}\right\rfloor$, on the lower bound game $\partial_h^{\mathrm{PP+}}$ for PP+. This is due to the fact that all resets are performed on regions that are not destination of any promotion. Also, the procedure appears to be much more robust, in terms of preventing resets, than the PP+ technique alone, to the point that it does not seem obvious whether an exponential lower bound even exists. Further investigation is, however, needed for a definite answer on its actual time complexity.

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 $\widetilde{r}$, 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 $\widetilde{r}$ is to record the delayed promotions obtained by moving the corresponding positions from their priority p in r to the new measure q in $\widetilde{r}$. Therefore, as dictated by Proposition 3.2, the union of $r^{-1}(q)$ and $\widetilde{r}^{-1}(q)$ must always be a region in the subgame $\partial_r^{\leq q}$. In addition, $\widetilde{r}^{-1}(q)$ 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 $\widetilde{r} \in \Delta^{\rightharpoonup} \triangleq \mathrm{Ps} \rightharpoonup \mathrm{Pr}$ is *aligned* with a region function r *w.r.t.* p if, for all priorities $q \in \mathrm{rng}(\widetilde{r})$, it holds that *(a)* $\widetilde{r}^{-1}(q) \subseteq \mathrm{dom}\big(r^{(>p)\wedge(<q)\wedge(\equiv_2 q)}\big)$ and *(b)* $r^{-1}(q)\cup\widetilde{r}^{-1}(q)$ is an α-region in $\partial_r^{\leq q}$ 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 $\mathscr{S} \triangleq \langle \mathrm{S}, \top, \prec \rangle$, where:*

1. *$\mathrm{S} \subseteq \mathrm{S}^{\mathrm{PP+}} \times \Delta^{\rightharpoonup} \times 2^{\mathrm{Pr}}$ is the set of all triples $s \triangleq ((r,p),\widetilde{r},\mathrm{P})$, called* states, *composed by a PP+ state $(r,p) \in \mathrm{S}^{\mathrm{PP+}}$, a partial priority function $\widetilde{r} \in \Delta^{\rightharpoonup}$ aligned with r w.r.t. p, and a set of priorities $\mathrm{P} \subseteq \mathrm{Pr}$.*

2. *$\top \triangleq (\top^{\mathrm{PP+}}, \varnothing, \emptyset)$;*

3. *$s_1 \prec s_2$ iff $\widehat{s_1} \prec^{\mathrm{PP+}} \widehat{s_2}$, 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 $\widetilde{r}$, but greater than the corresponding priority set in r. Condition *(a)* is the one characterising 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.

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

Hence, an α-region R is called α-*locked w.r.t.* a state $s \triangleq ((r,p),\widetilde{r},\mathrm{P})$ if the predicate $\phi_{Lck}(q,s) \triangleq \phi_a(q,\mathrm{P}) \vee \phi_b(q,r,\widetilde{r})$ is satisfied, where $q = \mathrm{bep}^{\overline{\alpha}}(\mathrm{R},r \uplus \widetilde{r})$. 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 contain the region $r^{-1}(p)$ associated to the priority p of the current state.

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

Algorithm 4 implements the successor function for DP. The pseudo-code on the right-hand side consists of three macros, namely *#Assignment*, *#InstantPromotion*, and *#DelayedPromotion*, used by the algorithm. Macro *#Assignment*(ξ) 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 f, the region is also recorded in $\widetilde{r}$ at the target priority of the promotion. 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 $\widetilde{r}$ and P. Macro *#DelayedPromotion*, instead, is responsible for the delayed promotion operation specific to DP.

Algorithm 4: Successor Function.	Assignment & Promotion Macros.
signature $\downarrow : \succ \to (\Delta \times Pr) \times \Delta^{\to} \times 2^{Pr}$	**macro** *#Assignment*(ξ)
function $s \downarrow (R, \alpha)$	1 $r^\star \leftarrow r[R \mapsto p]$
let $((r, p), \widetilde{r}, P) = s$ **in**	2 $p^\star \leftarrow \max(rng(r^{\star(<p)}))$
1 **if** $(R, \alpha) \in Rg_{\partial_s}^-$ **then**	3 $\widetilde{r}^\star \leftarrow$ **#if** ξ **#then** $\widetilde{r}$ **#else** $\widehat{r}$
2-5 *#Assignment*(t)	4 $P^\star \leftarrow P$
else	**macro** *#DelayedPromotion*
6 $q \leftarrow bep^{\overline{\alpha}}(R, r \uplus \widetilde{r})$	1 $p^\star \leftarrow \max(rng(\widehat{r}))$
7 **if** $\phi_{Lck}(q, s)$ **then**	2 $r^\star \leftarrow pr \uplus (r \uplus \widehat{r})^{(\geq p^\star) \vee (\equiv_2 p^\star)}$
8 $\widehat{r} \leftarrow \widetilde{r}[R \mapsto q]$	3 $\widetilde{r}^\star \leftarrow \emptyset$
9 **if** $R \neq Ps_{\partial_s}$ **then**	4 $P^\star \leftarrow \emptyset$
10-13 *#Assignment*(f)	**macro** *#InstantPromotion*
else	1 $p^\star \leftarrow q$
14-17 *#DelayedPromotion*	2 $r^\star \leftarrow pr \uplus r^{(\geq p^\star) \vee (\equiv_2 p^\star)}[R \mapsto p^\star]$
else	3 $\widetilde{r}^\star \leftarrow \widetilde{r}^{(>p^\star)}$
18-21 *#InstantPromotion*	4 $P^\star \leftarrow P \cap rng(r^\star) \cup \{p^\star\}$
22 **return** $((r^\star, p^\star), \widetilde{r}^\star, P^\star)$	

If the current region R is open in the subgame ∂_s, the main algorithm proceeds, similarly to Algorithm 3, at assigning to it the current priority p in r. This is done by calling macro *#Assignment* with parameter 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 $r \uplus \widetilde{r}$. 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 $\widetilde{r}$. This is done by executing *#Assignment* with parameter f. Finally, in case the entire game has been processed, the delayed promotion to the highest priority recorded in $\widetilde{r}$ is selected and applied. Macro *#DelayedPromotion* is executed, thus merging r with $\widetilde{r}$. Function $\widetilde{r}$ 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 $\eth$, the DP structure $\mathscr{D} \triangleq \langle \eth, \mathscr{S}, \succ, \mathfrak{R}, \downarrow \rangle$, where $\mathscr{S}$ is given in Definition 4.1, $\succ$ is the relation of Definition 4.2, and $\mathfrak{R}$ and $\downarrow$ are the functions computed by Algorithms 2 and 4, where in the former the assumption "**let** $(r, p) = s$" is replaced by "**let** $((r, p), \_, \_) = s$" is a dominion space.*

It is immediate to observe that the following mapping $h : ((r, p), \_, \_) \in S^{DP} \mapsto (r, p) \in S^{PP+}$, which takes DP states to PP+ states by simply forgetting the additional elements $\widetilde{r}$ 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 $\mathscr{D}_{\eth}$ for a game $\eth \in \mathrm{PG}$ 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 $\mathscr{D}_{\eth}^{PP+}$ for the same game.*

5   Experimental Evaluation

The technique proposed in the paper has been implemented in the tool PGSOLVER [9], which collects implementations of several parity game solvers proposed in the literature and provides benchmarking tools that can be used to evaluate the solver performances.[1]

Figure 3 compares the running times of the new algorithms, PP+ and DP, against the original version PP[2] and the well-known solvers *Rec* and *Str*, implementing the recursive algorithm [23] and the strategy improvement technique [22], respectively. This first pool of benchmarks is taken from [2] and involves 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 behaviour of the solvers is typically highly variable, even on games of the same size and priorities, to summarise 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. All the other solvers provided in PGSOLVER, including the Dominion Decomposition [14] and the Big Step [19] algorithms, perform quite poorly on those games, hitting the time-out already for very small instances. Figure 3 shows only the best performing ones on the considered games, namely *Rec* and *Str*. Similar experiments were also conducted on random games with a higher number of moves per position and up to 100000 positions. The resulting games turn out to be very easy to solve by all the priority promotion based approaches. 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. We reserve the presentation of the results for the extended version.

To further stress the DP technique in comparison with PP and PP+, we also generated a second pool of much harder benchmarks, containing more than 500 games, each with 50000 positions, 12000 priorities and 2 moves per positions. We selected as benchmarks only random games whose solution

---

[1]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.

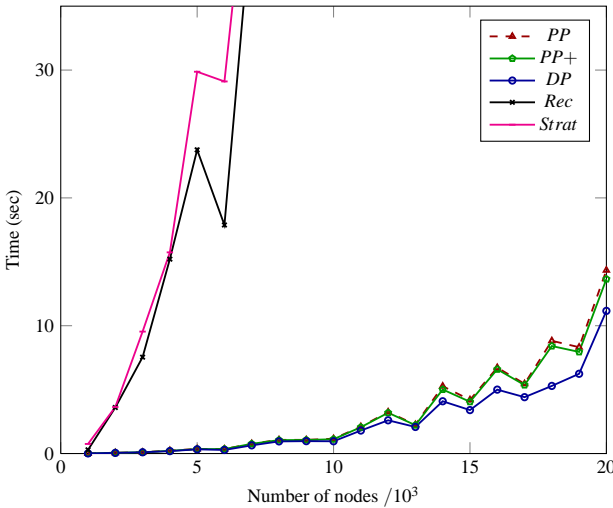[2]The version of PP used in the experiments is actually an improved implementation of the one described in [2].

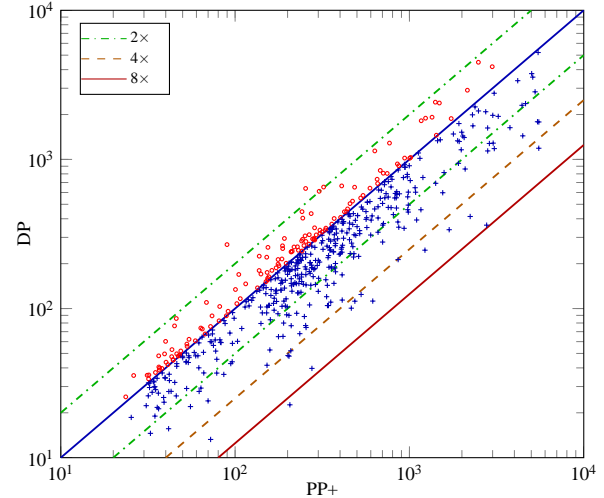**Figure 3:** Solution times on random games from [2].

Figure 4: Comparison between PP+ and DP on random games with 50000 positions.

requires PP+ between 30 and 6000 seconds. The results comparing PP+ and DP are reported in Figure 4 on a logarithmic scale. The figure shows that, in few cases, PP+ actually performs better than DP. This is due to the fact that the two algorithms follow different solution paths within the dominion space and that delaying promotions may also defer the discovery of a closed dominion. Nonetheless, the DP policy does pay off significantly on the vast majority of the benchmarks, often solving a game between two to eight times faster than PP+, as witnessed by the points below the dash-dotted line labeled $2\times$ in Figure 4.

In [2] it is shown that PP solves all the known exponential worst cases for the other solvers without promotions and, clearly, the same holds of DP as well. As a consequence, DP only requires polynomial time on those games and the experimental results coincide with the ones for PP.

6 Discussion

Devising efficient algorithms that can solve parity games well in practice is a crucial endeavour 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 recently proposed in [2]. While the technique seems very effective in practice, the approach still admits exponential behaviours. 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. The actual complexity of the algorithm is, however, currently unknown. Experiments on 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 proposed in the literature.

References

[1] K. Apt & E. Grädel (2011): *Lectures in Game Theory for Computer Scientists.* Cambridge University Press, doi:10.1017/CBO9780511973468.

[2] M. Benerecetti, D. Dell'Erba & F. Mogavero (2016): *Solving Parity Games via Priority Promotion.* In: *CAV'16*, LNCS 9780 (Part II), Springer, pp. 270–290, doi:10.1007/978-3-319-41540-6_15.

[3] K. Chatterjee, L. Doyen, T.A. Henzinger & J.-F. Raskin (2010): *Generalized Mean-Payoff and Energy Games.* In: *FSTTCS'10*, LIPIcs 8, Leibniz-Zentrum fuer Informatik, pp. 505–516, doi:10.4230/LIPIcs.FSTTCS.2010.505.

[4] A. Condon (1992): *The Complexity of Stochastic Games.* IC 96(2), pp. 203–224, doi:10.4230/LIPIcs.FSTTCS.2010.505.

[5] A. Ehrenfeucht & J. Mycielski (1979): *Positional Strategies for Mean Payoff Games.* IJGT 8(2), doi:10.1007/BF01768705.

[6] E.A. Emerson & C.S. Jutla (1988): *The Complexity of Tree Automata and Logics of Programs (Extended Abstract).* In: *FOCS'88*, IEEE Computer Society, pp. 328–337, doi:10.1109/SFCS.1988.21949.

[7] E.A. Emerson & C.S. Jutla (1991): *Tree Automata, muCalculus, and Determinacy.* In: *FOCS'91*, IEEE Computer Society, pp. 368–377, doi:10.1109/SFCS.1988.21949.

[8] E.A. Emerson, C.S. Jutla & A.P. Sistla (1993): *On Model Checking for the muCalculus and its Fragments.* In: *CAV'93*, LNCS 697, Springer, pp. 385–396, doi:10.1016/S0304-3975(00)00034-7.

[9] O. Friedmann & M. Lange (2009): *Solving Parity Games in Practice.* In: *ATVA'09*, LNCS 5799, Springer, pp. 182–196, doi:10.1007/978-3-642-04761-9_15.

[10] E. Grädel, W. Thomas & T. Wilke (2002): *Automata, Logics, and Infinite Games: A Guide to Current Research.* LNCS 2500, Springer, doi:10.1007/3-540-36387-4.

[11] V.A. Gurevich, A.V. Karzanov & L.G. Khachivan (1990): *Cyclic Games and an Algorithm to Find Minimax Cycle Means in Directed Graphs.* USSRCMMP 28(5), pp. 85–91, doi:10.1016/0041-5553(88)90012-2.

[12] M. Jurdziński (1998): *Deciding the Winner in Parity Games is in UP ∩ co-UP.* IPL 68(3), pp. 119–124, doi:10.1016/S0020-0190(98)00150-1.

[13] M. Jurdziński (2000): *Small Progress Measures for Solving Parity Games.* In: *STACS'00*, LNCS 1770, Springer, pp. 290–301, doi:10.1007/3-540-46541-3_24.

[14] M. Jurdziński, M. Paterson & U. Zwick (2008): *A Deterministic Subexponential Algorithm for Solving Parity Games.* SJM 38(4), pp. 1519–1532, doi:10.1137/070686652.

[15] O. Kupferman & M.Y. Vardi (1998): *Weak Alternating Automata and Tree Automata Emptiness.* In: *STOC'98*, Association for Computing Machinery, pp. 224–233, doi:10.1145/276698.276748.

[16] O. Kupferman, M.Y. Vardi & P. Wolper (2000): *An Automata Theoretic Approach to Branching-Time Model Checking.* JACM 47(2), pp. 312–360, doi:10.1145/333979.333987.

[17] A.W. Mostowski (1984): *Regular Expressions for Infinite Trees and a Standard Form of Automata.* In: *SCT'84*, LNCS 208, Springer, pp. 157–168, doi:10.1007/3-540-16066-3_15.

[18] A.W. Mostowski (1991): *Games with Forbidden Positions.* Technical Report, University of Gdańsk, Gdańsk, Poland.

[19] S. Schewe (2007): *Solving Parity Games in Big Steps.* In: *FSTTCS'07*, LNCS 4855, Springer, pp. 449–460, doi:10.1007/978-3-540-77050-3_37.

[20] S. Schewe (2008): *An Optimal Strategy Improvement Algorithm for Solving Parity and Payoff Games.* In: *CSL'08*, LNCS 5213, Springer, pp. 369–384, doi:10.1007/978-3-540-87531-4_27.

[21] S. Schewe, A. Trivedi & T. Varghese (2015): *Symmetric Strategy Improvement.* In: *ICALP'15*, LNCS 9135, Springer, pp. 388–400, doi:10.1007/978-3-662-47666-6_31.

[22] J. Vöge & M. Jurdziński (2000): *A Discrete Strategy Improvement Algorithm for Solving Parity Games.* In: *CAV'00*, LNCS 1855, Springer, pp. 202–215, doi:10.1007/10722167_18.

[23] W. Zielonka (1998): *Infinite Games on Finitely Coloured Graphs with Applications to Automata on Infinite Trees.* *TCS* 200(1-2), pp. 135–183, doi:10.1016/S0304-3975(98)00009-7.

[24] U. Zwick & M. Paterson (1996): *The Complexity of Mean Payoff Games on Graphs.* *TCS* 158(1-2), pp. 343–359, doi:10.1016/0304-3975(95)00188-3.