Improving Priority Promotion for Parity Games

Massimo Benerecetti[1], Daniele Dell'Erba[1], and Fabio Mogavero[2]

[1] Università degli Studi di Napoli Federico II [2] Oxford University

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 *region recovery*, 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 graph*, 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, such 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 ∩ CoUPTime class [12]. That result improves the NPTime ∩ 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 the space complexity of the best know algorithm by an exponential factor, since 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 previously computed quasi dominions at lower ones.

In order to mitigate the problem, we propose in this paper a new algorithm, called RR for *region recovery*, which is built on top of of PP and is based on a form of conservation property of quasi dominions. This property provides sufficient conditions for a subset a quasi α-dominion to be still a quasi α-dominion. By exploiting this property, the RR algorithm can significantly reduce the execution of the resetting phase, which is now limited to the cases when the conservation property is not guaranteed to hold. For the resulting procedure no exponential worst case has been found yet. Experiments on randomly generated games also show that the new approach performs significantly 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 $\mathsf{f} : \mathrm{A} \rightharpoonup \mathrm{B}$, by $\mathsf{dom}(\mathsf{f}) \subseteq \mathrm{A}$ and $\mathsf{rng}(\mathsf{f}) \subseteq \mathrm{B}$ we denote 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. 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 $\eth = \langle \mathcal{A}, \mathrm{Pr}, \mathsf{pr} \rangle$, 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. We denote with PG the class of parity games. The priority function can be naturally extended to games, sets of positions, and paths as follows: $\mathsf{pr}(\eth) \triangleq \mathsf{max}_{v \in \mathrm{Ps}} \mathsf{pr}(v)$; for a set of positions $\mathrm{V} \subseteq \mathrm{Ps}$, we set $\mathsf{pr}(\mathrm{V}) \triangleq \mathsf{max}_{v \in \mathrm{V}} \mathsf{pr}(v)$; 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. The maximal α-dominion in a game, denoted Wn_α, is called winning region of player α. By $\partial \setminus \mathrm{V}$ we denote the maximal subgame of ∂ 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 $\mathsf{atr}^\alpha(\mathrm{V})$ generalises the notion of α-predecessor $\mathsf{pre}^\alpha(\mathrm{V})$ to an arbitrary number of moves. Thus, it 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 this set. For such a V, the set of positions of the subgame $\partial \setminus \mathrm{V}$ is precisely $\mathrm{Ps} \setminus \mathrm{V}$. Finally, the *$\alpha$-escape* of V, formally $\mathsf{esc}^\alpha(\mathrm{V}) \triangleq \mathsf{pre}^\alpha(\mathrm{Ps}\setminus\mathrm{V}) \cap \mathrm{V}$, 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 the α-positions from which α cannot escape with a single move, while the notion of *α-stay*, defined as $\mathsf{stay}^\alpha(\mathrm{V}) \triangleq (\mathrm{V} \cap \mathrm{Ps}^\alpha) \setminus \mathsf{esc}^{\overline{\alpha}}(\mathrm{V})$, denotes the α-positions from which α has a move to remain in V.

3 Quasi Dominion Approach

The priority promotion algorithm proposed in [2] attacks the problem of solving a parity game ∂ 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 ∂, the smaller game $\partial \setminus \mathrm{D}^\star$ 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 to that of iteratively finding its dominions [14].

In order to solve the dominion problem, the idea described in [2] is to introduce a much weaker notion than that of dominion, called *quasi dominion*, which satisfies, under suitable conditions, a composition property that eventually brings to the construction of a dominion. Intuitively, a quasi α-dominion Q is a set of positions on which player α has a *witness strategy* σ_α, whose induced plays either remain inside Q forever and are winning for α or can exit from Q passing through a specific set of escape positions.

Definition 1 (Quasi Dominion [2]). *Let $\partial \in \mathrm{PG}$ 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 ∂ if there exists an α-strategy $\sigma_\alpha \in \mathrm{Str}^\alpha(\mathrm{Q})$, called* α-witness *for 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 plays induced by the witness σ_α remain in the set Q forever, this is actually an α-dominion and, therefore, a subset of the winning region Wn_α of α, with σ_α the projection over Q of some α-winning strategy on the entire game. 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 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, *e.g.*, by visiting a cycle containing a position with maximal priority of parity $\overline{\alpha}$. The set of triples $(\mathrm{Q}, \sigma, \alpha) \in 2^{\mathrm{Ps}} \times \mathrm{Str} \times \{0, 1\}$, where Q is a quasi α-dominion having σ as one of its α-witnesses, is denoted by QD, and is partitioned into the sets QD^- and QD^+ of open and closed quasi α-dominion triples, respectively.

Similarly to the other *divide et impera* techniques proposed in the literature, the one reported in [2], called PP, does not make any algorithmic use of the witness strategy σ_α associated with a quasi dominion Q, as this notion is only employed in the correctness proof. In this work, instead, we strongly exploit the effective computability of such a witness in order to considerably alleviate the collateral effects of a *reset operation* required by PP to ensure the soundness of the approach, which is also responsible for the exponential worst cases. Indeed, this algorithm needs to forget previously computed partial results after each compositions of two quasi-dominions, since the information computed during the entire process cannot ensure that these results can be still correctly used in the search for a dominion. In this work, instead, we exploit the following simple observation on the witness strategies, formally reported in Lemma 1, to determine which partial results can be safely preserved.

In general, quasi α-dominions are not closed under restriction. For example, consider the quasi 1-dominion $\mathrm{Q}^\star = \{\mathtt{a}, \mathtt{c}, \mathtt{d}\}$ of Figure 1 with unique 1-witness strategy $\sigma^\star = \{\mathtt{c} \mapsto \mathtt{a}, \mathtt{d} \mapsto \mathtt{c}\}$ and its subset $\mathrm{Q} = \{\mathtt{d}\}$. It is quite immediate to see that Q is not a quasi 1-dominion, since $\mathsf{esc}^0(\mathrm{Q}) = \emptyset$, but player 1 does not have a strategy that induces infinite 1-winning plays that remain in Q. Indeed, $\sigma^\star$ requires player 1 to exit from Q by going from d to c. On the contrary, the subset $\mathrm{Q} = \{\mathtt{c}, \mathtt{d}\}$ is still a 1-dominion with 1-witness strategy $\sigma = \{\mathtt{d} \mapsto \mathtt{c}\} = \sigma^\star {\restriction} \mathsf{stay}^1(\mathrm{Q})$, since $\mathsf{esc}^0(\mathrm{Q}) = \{\mathtt{c}\}$. Hence, the restriction σ of $\sigma^\star$ to $\mathsf{stay}^1(\mathrm{Q}) = \{\mathtt{d}\}$ is still a well-defined strategy on Q. Inspired by this observation, we provide the following sufficient criterion for a subset Q of a quasi α-dominion $\mathrm{Q}^\star$ to be still a quasi α-dominion.

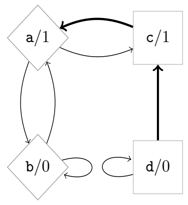

Fig. 1: Witness strategy.

Lemma 1 (Witness Strategy). *Given a quasi α-dominion $\mathrm{Q}^\star$ having $\sigma^\star \in \mathrm{Str}^\alpha(\mathrm{Q}^\star)$ as* α-witness *and a subset of positions $\mathrm{Q} \subseteq \mathrm{Q}^\star$, the restriction $\sigma \triangleq \sigma^\star {\restriction} \mathsf{stay}^\alpha(\mathrm{Q})$ is an* α-witness *for Q iff $\sigma \in \mathrm{Str}^\alpha(\mathrm{Q})$.*

The proof-idea behind this lemma is very simple. Any infinite play induced by the restriction of σ on Q is necessarily winning for player α, since it is coherent with the original α-witness $\sigma^\star$ of $\mathrm{Q}^\star$ as well. Now, if $\sigma \in \mathrm{Str}^\alpha(\mathrm{Q})$, we are also sure that any finite play ends in $\mathsf{esc}^{\overline{\alpha}}(\mathrm{Q})$, as required by the definition of quasi dominion. Therefore, σ is an α-witness for Q, which is, then, a quasi α-dominion. On the other hand, if $\sigma \notin \mathrm{Str}^\alpha(\mathrm{Q})$, there exists a finite play induced by σ that does not terminate in $\mathsf{esc}^{\overline{\alpha}}(\mathrm{Q})$. Hence, σ is not an α-witness. In this case, we cannot ensure that Q is a quasi α-dominion.

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 α-dominion Q together with one of its possible α-witnesses σ 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 Q is open. If, on the other hand, it is closed, the search is over. Algorithm 1 implements the dominion search procedure $\mathsf{src}_{\mathcal{D}}$. A *compatibility relation* $\succ$ connects the query and the successor operators. The relation holds between states of the partial order and the quasi dominions triples 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*.

Algorithm 1: The Searcher.

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

Definition 2 (Dominion Space). *A* dominion space *for a game* $\eth \in \mathrm{PG}$ *is a tuple* $\mathcal{D} \triangleq \langle \eth, \mathcal{S}, \succ, \Re, \downarrow \rangle$*, 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) $\succ \subseteq \mathcal{S} \times \mathrm{QD}^{-}_{\eth}$ *is the* compatibility relation*,* (3) $\Re : \mathrm{S} \to \mathrm{QD}_{\eth}$ *is the* query operator *mapping each element* $s \in \mathrm{S}$ *to a quasi dominion triple* $(\mathrm{Q}, \sigma, \alpha) \triangleq \Re(s) \in \mathrm{QD}_{\eth}$ *such that, if* $(\mathrm{Q}, \sigma, \alpha) \in \mathrm{QD}^{-}_{\eth}$ *then* $s \succ (\mathrm{Q}, \alpha, \sigma)$*, and* (4) $\downarrow : \succ \to \mathrm{S}$ *is the* successor operator *mapping each pair* $(s, (\mathrm{Q}, \sigma, \alpha)) \in \succ$ *to the element* $s^{\star} \triangleq s \downarrow (\mathrm{Q}, \sigma, \alpha) \in \mathrm{S}$ *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 $\mathsf{src}_{\mathcal{D}}$ 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 [2]. In the next section, we propose a different one, called RR *dominion space*, which crucially exploits Lemma 1 in order to prevent a considerable amount of useless reset operations after each quasi dominion composition, to the point that it does not seem obvious whether an exponential lower bound even exists for this new approach.

4 Priority Promotion with Region Recovery

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 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.

Solving problem *(b)* is not trivial, since quasi α-dominions are not, in general, closed under union. Consider the example in Figure 2. Both $Q_1 = \{\mathsf{a}, \mathsf{c}\}$ and $Q_2 = \{\mathsf{b}, \mathsf{d}\}$ are quasi 0-dominions. Indeed, $\sigma_1 = \{\mathsf{c} \mapsto \mathsf{c}\}$ and $\sigma_2 = \{\mathsf{d} \mapsto \mathsf{d}\}$ are the corresponding 0-witnesses. However, their union $Q \triangleq Q_1 \cup Q_2$ is not a quasi 0-dominion, since the 1-strategy $\sigma = \{\mathsf{a} \mapsto \mathsf{b}, \mathsf{b} \mapsto \mathsf{a}\}$ forces player 0 to lose along any infinite play starting from either a or b.

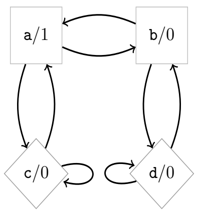

Fig. 2: Quasi dominions.

A solution to both problems 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 quasi α-dominion of ∂ with the additional requirement that all the 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 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 (Region [2]). *A quasi α-dominion* R *is an α-region in ∂ if* $\mathsf{pr}(\partial) \equiv_2 \alpha$ *and all the positions in* $\mathsf{esc}^{\overline{\alpha}}(\mathrm{R})$ *have priority* $\mathsf{pr}(\partial)$*, i.e.,* $\mathsf{esc}^{\overline{\alpha}}(\mathrm{R}) \subseteq \mathsf{pr}^{-1}(\mathsf{pr}(\partial))$*.*

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

A solution to the second problem, the merging operation, is obtained as follows. Given an α-region R in some game $\Game$ and an α-dominion D in a subgame of $\Game$ 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 2 (Region Merging [2]). *Let* $\Game \in \mathrm{PG}$ *be a game,* $\mathrm{R} \subseteq \mathrm{Ps}$ *an* α*-region, and* $\mathrm{D} \subseteq \mathrm{Ps}_{\Game \setminus \mathrm{R}}$ *an* α*-dominion in the subgame* $\Game \setminus \mathrm{R}$*. Then,* $\mathrm{R}^{\star} \triangleq \mathrm{R} \cup \mathrm{D}$ *is an* α*-region in* $\Game$*. Moreover, if both* R *and* D *are* α*-maximal in* $\Game$ *and* $\Game \setminus \mathrm{R}$*, respectively, then* $\mathrm{R}^{\star}$ *is* α*-maximal in* $\Game$ *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. As shown in [2], the reset operation is the main source of the exponential behaviours of the approach. We shall propose here a different approach that, based on the result of Lemma 1, can drastically reduce the number of resets needed.

	1	2	3	4	5
8	a$\downarrow$	=	=	a,b,c,d,h,i,j$\downarrow$	=
7	j,k$\downarrow$	=	=		
5	b$\downarrow$	=	b,c,e,f,g,h$\downarrow$		
4	c,d$\downarrow$	=		k$\downarrow$	=
3	e,f$\downarrow$	e,f,g,h$\uparrow_5$	d$\downarrow$	e,f$\downarrow$	e,f,g
1	g,h$\uparrow_3$			g$\uparrow_3$	
0			i$\uparrow_8$		

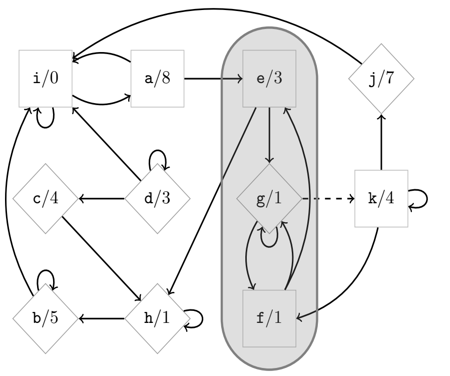

Fig. 3: Running example.

Figure 3 illustrates the dominion search procedure on an example game. Diamond shaped positions belong to player 0 and square shaped ones to opponent 1. Each cell of the table contains a computed region. The downward arrow denotes that the region is open in the subgame where is computed, while the upward arrow means that the region gets to be promoted to the priority in the subscript. The measure of the region correspond to the index of the row in which the region is contained. Empty slots in the table represent empty regions, while a slot with symbol = in it means that the it contains the same region as the corresponding slot in the previous column.

Assume the dashed move $(\mathtt{g},\mathtt{k})$ is not present in the game. Then, following the idea sketched above, the first region obtained is the single-position 0-region $\{\mathtt{a}\}$ at priority 8, which is open because of the two moves leading to e and i. At priority 7, the open 1-region $\{\mathtt{j},\mathtt{k}\}$ is formed, by attracting k to j according to Proposition 1, which is open in the subgame where $\{\mathtt{a}\}$ is removed. The procedures proceeds similarly, processing all the priorities down to 1 and extracting the regions reported in the first column of the table of Figure 3. Those are all open regions in their corresponding subgames, except for the 1-region $\{\mathtt{g},\mathtt{h}\}$ at priority 1, which is closed in its subgames but not in the entire game. This region has a move $(\mathtt{g},\mathtt{f})$ leading to region 3 and Proposition 2 is then applied, which promotes this region to 3, obtaining a new 1-region $\{\mathtt{e},\mathtt{f},\mathtt{g},\mathtt{h}\}$ with measure 3. This one is again closed in its subgames and, due to move $(\mathtt{h},\mathtt{b})$, triggers another application of Proposition 2, which promotes all of its positions to region 5 and resets the positions in region 4 to their original priority. The search resumes at priority 5 and the maximization of that region attracts position c as well, forming region $\{\mathtt{b},\mathtt{c},\mathtt{e},\mathtt{f},\mathtt{g},\mathtt{h}\}$ with measure 5. In the resulting subgame, the procedure now extracts the open 1-region $\{\mathtt{d}\}$ at priority 3. The residual game only contains position i, that forms a closed 0-region with a move leading to region 8. This triggers a new promotion that resets the position of all the regions with measure lower than 8, namely the regions with measures 7 and 5. After maximization of the target region, positions b, c, d, h, and j are all attracted to form the 0-region in the first row of column 4. The reset of previous region 7 releases position k which now forms an open 0-region of priority 4. Similarly, positions e and f, reset by the last promotion, form an open 1-region at priority 3. Finally, at priority 1 the closed 1-region $\{\mathtt{g}\}$ is extracted and promoted, by move $(\mathtt{g},\mathtt{f})$, to region 3, forming the set $\{\mathtt{e},\mathtt{f},\mathtt{g}\}$. Since no move from 0-positions lead outside the set, this region is closed in the entire game and a 1-dominion has been found.

During the simulation above, three resets have been performed. The first one resets 0-region $\{\mathtt{c},\mathtt{d}\}$ with measure 4 in column 2, after promotion of region $\{\mathtt{e},\mathtt{f},\mathtt{g},\mathtt{h}\}$ to priority 5. Indeed, the maximization of the resulting region $\{\mathtt{b},\mathtt{e},\mathtt{f},\mathtt{g},\mathtt{h}\}$ attracts position c, leaving the set $\{\mathtt{d}\}$ with measure 4. However, according to Definition 1, this is not a quasi 0-dominion, since its 1-escape is empty and player 1 can win by remaining in the set forever. After the promotion of region $\{\mathtt{i}\}$ to 8 in column 3, both the regions in rows 7 and 5 get reset. The maximization of the target region of the promotion, *i.e.*,$\{\mathtt{a},\mathtt{i}\}$, attracts positions b, c, d, h, and j. As a consequence, for similar reasons described above for position d, the residual position k at priority 7 must be reset to its original priority. Notice that, in both the considered cases, Lemma 1 does not apply. Indeed, the 0-witness strategy for region $\{\mathtt{c},\mathtt{d}\}$ is $\sigma = \{\mathtt{d} \mapsto \mathtt{c}\}$, the 0-stay set of the residual region $\{\mathtt{d}\}$ is the set itself, and the restriction of σ to $\{\mathtt{d}\}$ leads outside $\{\mathtt{d}\}$, hence, it does not belong to $\mathrm{Str}^{\circ}(\{\mathtt{d}\})$. A similar argument applies to set $\{\mathtt{k}\}$ as well.

As opposed to this case, however, the reset of region 5 can be avoided, thanks to Lemma 1. Indeed, a 1-witness for that region is $\sigma = \{\mathtt{e} \mapsto \mathtt{g}, \mathtt{f} \mapsto \mathtt{e}\}$ and, in this case, the residual set after the promotion and the maximization of the target region 8 is $\{\mathtt{e}, \mathtt{f}, \mathtt{g}\}$, whose 1-stay set is $\{\mathtt{e}, \mathtt{f}\}$. The restriction of σ to that set is, however, contained in $\mathrm{Str}^1(\{\mathtt{e}, \mathtt{f}\})$ and the lemma applies. Note that, avoiding the reset of region with measure 5, containing $\{e, f, g\}$ in column 4, would also avoid the computation of regions 4, 3, and 1, and the promotion of region 1 to 3 that leads to column 5. Indeed, the residual region 5 is a 1-region, according to the lemma, and is also closed in the entire game.

If, however, the dashed move $(\mathtt{g}, \mathtt{k})$ was added to the game, the reset of region 5 would be necessary. The reason is that, in this case, the 0-escape set $\{\mathtt{e}, \mathtt{f}, \mathtt{g}\}$ would contain position $\mathtt{g}$, which can escape to position $\mathtt{k}$. As a consequence, $\{\mathtt{e}, \mathtt{f}, \mathtt{g}\}$ would not be a 1-region as the escape set contains a position with priority non-maximal in the subgame, contrary to what is required by Definition 3.

In summary, we can exploit Lemma 1 and Definition 3 to avoid resetting regions after a promotion whenever *(i)* the witness strategy of the residual region satisfies the condition of the lemma, and *(ii)* its escape set only contains positions of maximal priorities in the subgame. This is the core observation that allows the definition of the RR approach, which is formally defined in the following.

The* RR *Dominion Space. We can now provide the formal account of the RR dominion space. We shall denote with Rg the set of region triples in ∂ and with Rg^- and Rg^+ the sets of open and closed region triples, respectively.

Similarly to the PP 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 $\mathsf{r}^{(\neq p)}$) 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 different from p). Formally, $\mathsf{r}^{(\sim p)} \triangleq \mathsf{r}\restriction\{v \in \mathrm{Ps} : \mathsf{r}(v) \sim p\}$, for $\sim \in \{\geq, >, <, \neq\}$. By $\partial_{\mathsf{r}}^{\leq p} \triangleq \partial \setminus \mathsf{dom}(\mathsf{r}^{(>p)})$, we denote the largest subgame obtained by removing from ∂ all the positions in the domain of $\mathsf{r}^{(>p)}$. In order for the RR procedure to exploit Lemma 1, it also needs to keep track of witness strategies of the computed region. To this end, we introduce the notion of witness core. A strategy $\sigma \in \mathrm{Str}^{\alpha}(\mathrm{Q})$ is an *α-witness core* for an α-region R if *(i)* it is defined on all positions having priority lower than $\mathsf{pr}(\mathrm{R})$, *i.e.*, $\{v \in \mathrm{R} : \mathsf{pr}(v) < \mathsf{pr}(\mathrm{R})\} \subseteq \mathsf{dom}(\sigma)$, and *(ii)* it is a restriction of some α-witness $\varsigma \in \mathrm{Str}^{\alpha}$ for R, *i.e.*, $\sigma \subseteq \varsigma$. Intuitively, a witness core only maintains the essential part of a witness and can be easily transformed into a complete witness by associating every position $v \in \mathsf{stay}^{\alpha}(\mathrm{R}) \setminus \mathsf{dom}(\sigma)$ with an arbitrary successor in R. The result of any such completion is an actual witness, since any infinite path passing through v is forced to visit a maximal priority of parity α.

Definition 4 (Region-Witness Pair). *Let $\mathsf{r} \in \Delta$ be a priority function, $\tau \in \mathrm{Str}$ a strategy, and $p \in \mathrm{Pr}$ a priority. The pair (r, τ) is a* region-witness pair w.r.t. *p if, for all $q \in \mathsf{rng}(\mathsf{r})$ with $\alpha \triangleq q \bmod 2$, $\mathrm{R} \triangleq \mathsf{r}^{-1}(q) \cap \mathrm{Ps}_{\partial_{\mathsf{r}}^{\leq q}} \neq \emptyset$, and $\sigma \triangleq \tau\restriction\mathrm{R}$, the following two conditions hold:*

1. *if $q \geq p$, then R is an α-region in the subgame $\partial_{\mathsf{r}}^{\leq q}$ with α-witness core σ;*

2. *if $q < p$, there exists a quasi α-dominion $\mathrm{Q}^\star \supseteq \mathrm{R}$ with α-witness $\sigma^\star$ such that* (i) $\mathsf{pr}(\mathrm{Q}^\star) = q$, (ii) $\sigma \subseteq \sigma^\star$, *and* (iii) $(\mathrm{R} \cap \mathrm{Ps}^\alpha) \setminus \mathsf{dom}(\sigma) \subseteq \mathsf{pr}^{-1}(q)$.

In addition, r *is* maximal *above* $p \in \mathrm{Pr}$ *iff, whenever* $q > p$*, it holds that* R *is* α*-maximal in* $\eth_{\mathsf{r}}^{\leq q}$ *as well.*

As opposed to the PP approach, where a promotion to a priority p resets all the regions of measure lower than p, the RR algorithm resets lower regions only when it cannot ensure their validity. This is done one region at a time, during the descend phase. If, while reading a set $\mathsf{r}^{-1}(q)$ at a certain priority $q < p$, the conditions of Lemma 1 are not met by $\mathsf{r}^{-1}(q)$ or the escape of that region contains positions of priority lower than q, then $\mathsf{r}^{-1}(q)$ is reset.

Contrary to PP, for which the set contained in r at each measure q must be an α-region, RR requires such a property only for those regions with measure $q \geq p$, as expressed by Item 1 of the previous definition. For each $q < p$, instead, we simply require the set of positions contained in r at that measure to be a subset of some previously computed quasi dominions of the same player. This is done by requiring that the strategies recorded in τ be subsets of witnesses of these dominions, as described in Item 2. In this way, to verify that $\mathsf{r}^{-1}(q)$ is still a quasi α-dominion, RR can apply the property stated in Lemma 1.

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-witness pair (r, τ) and the current priority p reached by the search in $\eth$. Initially, r coincides with the priority function pr of the entire game $\eth$, τ is the empty strategy, and p is set to the maximal priority $\mathsf{pr}(\eth)$ 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 RR procedure can reside and the relative order that the successor function must satisfy.

Definition 5 (State Space). *A* state space *is a tuple* $\mathcal{S} \triangleq \langle \mathrm{S}, \top, \prec \rangle$*, where:*

1. $\mathrm{S} \subseteq \Delta \times \mathrm{Str} \times \mathrm{Pr}$ *is the set of triples* $s \triangleq (\mathsf{r}, \tau, p)$*, called* states*, where* (a) (r, τ) *is a region-witness pair* w.r.t. p, (b) r *is maximal above* p*, and* (c) $p \in \mathsf{rng}(\mathsf{r})$.
2. $\top \triangleq (\mathsf{pr}, \varnothing, \mathsf{pr}(\eth))$;
3. *for any two states* $s_1 \triangleq (\mathsf{r}_1, \_, p_1), s_2 \triangleq (\mathsf{r}_2, \_, p_2) \in \mathrm{S}$*, it holds that* $s_1 \prec s_2$ *iff either* (a) *there exists a priority* $q \in \mathsf{rng}(\mathsf{r}_1)$ *with* $q \geq p_1$ *such that* (a.i) $\mathsf{r}_1{}^{(>q)} = \mathsf{r}_2{}^{(>q)}$ *and* (a.ii) $\mathsf{r}_2^{-1}(q) \subset \mathsf{r}_1^{-1}(q)$*, or* (b) *both* (b.i) $\mathsf{r}_1^{(\geq p_2)} = \mathsf{r}_2^{(\geq p_2)}$ *and* (b.ii) $p_1 < p_2$ *hold.*

Condition 1 requires every region $\mathsf{r}^{-1}(q)$ with measure $q > p$ to 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 must be one of the measures recorded in r. Condition 2 specifies the initial state. Finally, 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.

As reported in Definition 2, the compatibility relation describes which regions are compatibles with a state, *i.e.*, which region triples can be returned by the query operator and used by the successor function. A region triple $(\mathrm{R}, \sigma, \alpha)$ is compatible with a state $s \triangleq (\mathsf{r}, \tau, p)$ if R 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 6 (Compatibility Relation). *An open quasi dominion triple* $(\mathrm{R}, \sigma, \alpha) \in \mathrm{QD}^-$ *is* compatible *with a state* $s \triangleq (\mathsf{r}, \tau, p) \in \mathrm{S}$*, in symbols* $s \succ (\mathrm{R}, \sigma, \alpha)$*, iff* (1) $(\mathrm{R}, \sigma, \alpha) \in \mathrm{Rg}_{\eth_s}$ *and* (2) *if* R *is* α*-open in* $\eth_s$ *then* (2.a) R *is* α*-maximal in* $\eth_s$ *and* (2.b) $\mathsf{r}^{-1}(p) \subseteq \mathrm{R}$.

Algorithm 2: Query Function.

```
   signature $\Re : \mathrm{S} \to 2^{\mathrm{Ps}} \times \mathrm{Str} \times \{0,1\}$
   function $\Re(s)$
       let $(\mathsf{r}, \tau, p) = s$ in
1          $\alpha \leftarrow p \bmod 2$
2          $\mathrm{R}^\star \leftarrow \mathsf{r}^{-1}(p)$
3          $(\mathrm{R}, \sigma) \leftarrow \mathsf{atr}^{\alpha}_{\eth_s}(\mathrm{R}^\star, \tau \upharpoonright \mathrm{R}^\star)$
4      return $(\mathrm{R}, \sigma, \alpha)$
```

Algorithm 2 provides a possible implementation for the query function compatible with the region-recovery mechanism. Given the current state $s \triangleq (\mathsf{r}, \tau, p)$, Line 1 simply computes the parity α of the priority p to process at s. Line 3, instead, computes the attractor *w.r.t.* player α in subgame $\eth_s$ of the region $\mathrm{R}^\star$ contained in r at p, as determined by Line 2. Observe that here we employ a version of the α-attractor that, given an α-witness core for $\mathrm{R}^\star$, also computes the α-witness for R. This can easily be done by first extending $\tau \upharpoonright \mathrm{R}^\star$ with the attraction strategy on the α-positions in $\mathrm{R} \setminus \mathrm{R}^\star$ and, then, by choosing, for any α-positions in $\mathrm{R} \setminus \mathsf{dom}(\tau \upharpoonright \mathrm{R}^\star)$ with a successor in $\mathrm{R} \setminus \mathrm{R}^\star$, any one of those successors. The resulting set R is, according to Proposition 1, an α-maximal α-region of $\eth_s$ containing $\mathsf{r}^{-1}(p)$ with α-witness σ.

Algorithm 3: Successor Function.

```
   signature $\downarrow : \succ \to \Delta \times \mathrm{Str} \times \mathrm{Pr}$
   function $s \downarrow (\mathrm{R}, \sigma, \alpha)$
       let $(\mathsf{r}, \tau, p) = s$ in
1          if $(\mathrm{R}, \sigma, \alpha) \in \mathrm{Rg}^-_{\eth_s}$ then
2              return $\mathsf{N}(\mathsf{r}[\mathrm{R} \mapsto p], \tau \uplus \sigma, p)$
           else
3              $p^\star \leftarrow \mathsf{bep}^{\overline{\alpha}}(\mathrm{R}, \mathsf{r})$
4              return $(\mathsf{r}[\mathrm{R} \mapsto p^\star], \tau \uplus \sigma, p^\star)$
```

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 incident move coming from the α-region, closed in the current subgame, that needs to be promoted. This concept is formally defined as follows. Let $I \triangleq Mv \cap ((\mathrm{R} \cap \mathrm{Ps}^{\overline{\alpha}}) \times (\mathsf{dom}(\mathsf{r}) \setminus \mathrm{R}))$ 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, $\mathsf{bep}^{\overline{\alpha}}(\mathrm{R}, \mathsf{r}) \triangleq \mathsf{min}(\mathsf{rng}(\mathsf{r} \upharpoonright \mathsf{rng}(I)))$. 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 $\eth_s$, 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 1-region $\mathrm{R} = \{\mathtt{g}, \mathtt{h}\}$ with measure 1 in Column 1 of Figure 3, we have that $I = \{(\mathtt{g}, \mathtt{f}), (\mathtt{h}, \mathtt{b})\}$ and $\mathsf{r}{\upharpoonright}\mathsf{rng}(I) = \{(\mathtt{b}, 5), (\mathtt{f}, 3)\}$. Hence, $\mathsf{bep}^{\mathrm{o}}(\mathrm{R}, \mathsf{r}) = 3$.

Algorithm 3 implements the successor function. Given the state $s \triangleq (\mathsf{r}, \tau, p)$ and one of its possible compatible region triples $(\mathrm{R}, \sigma, \alpha)$ open in the original game $\eth$, it produces a successor state $s^\star \prec s$. Line 1 checks if R is open in the subgame $\eth_s$ as well. If this is the case, at Line 2, the next state $s^\star$ is generated by the auxiliary function, called *next state function*, described below, which also applies the required resets. On the other hand, if R is closed in $\eth_s$, the procedure performs the promotion of R, by exploiting Proposition 2. Indeed, Line 3 computes the best escape priority $p^\star$ to which R needs to be promoted, while Line 4 sets the measure of R to $p^\star$ and merges the strategies contained in τ with the α-witness σ of R. Observe that, unlike in the PP successor function, no reset operation is applied to r at this stage.

Algorithm 4: Next State Function.

```
signature $\mathsf{N} : \mathrm{S} \to \Delta \times \mathrm{Str} \times \mathrm{Pr}$
function $\mathsf{N}(s)$
    let $(\mathsf{r}, \tau, p) = s$ in
1       $p^\star \leftarrow \mathsf{max}(\mathsf{rng}(\mathsf{r}^{(<p)}))$
2       if $\phi(\mathsf{r}, \tau, p^\star)$ then
3           return $(\mathsf{r}, \tau, p^\star)$
        else
4           $\mathsf{r}^\star \leftarrow \mathsf{pr} \uplus \mathsf{r}^{(\neq p^\star)}$
5           $\tau^\star \leftarrow \sigma \setminus \mathsf{r}^{-1}(p^\star)$
6           return $\mathsf{N}(\mathsf{r}^\star, \tau^\star, p)$
```

Finally, Algorithm 4 reports the pseudo code of the next state function, the essential core of the RR approach. At a state $s \triangleq (\mathsf{r}, \sigma, p)$, Line 1 computes the priority $p^\star$ of the successive set of positions $\mathsf{r}^{-1}(p^\star)$ occurring in r starting from p in descending order of priority. Then, Line 2 verifies whether this set is actually a region, by computing the truth value of the formula ϕ described below applied to the triple $(\mathsf{r}, \tau, p^\star)$. If this is the case, the successor state $(\mathsf{r}, \tau, p^\star)$ of s is returned at Line 3. On the other hand, if the check fails, the algorithm resets, at Line 4, the positions in $\mathsf{r}^{-1}(p^\star)$ to their original priority stored in the priority function pr of the game, and deletes, at Line 5, the associated strategy contained in τ. Finally, at Line 6, the next state function is recursively applied to the newly obtained state.

To check whether a set of positions $\mathrm{R} \triangleq \mathsf{r}^{-1}(p)$ at a certain priority $q < p$ is an α-region with $\alpha \triangleq q \bmod 2$, we make use of the formula $\phi(\mathsf{r}, \tau, q) \triangleq \phi_i(\mathsf{r}, \tau, q) \wedge \phi_{ii}(\mathsf{r}, q)$, which verifies that *(i)* $\sigma \triangleq \tau{\upharpoonright}\mathrm{R}$ is a witness core for R and *(ii)* the escape only contains positions of maximal priorities in the subgame. The two predicates are formally defined as follows.

$$\begin{aligned} \phi_i(\mathsf{r}, \tau, q) &\triangleq \forall v \in \mathrm{Ps}^{\alpha}_{\eth^\star} \cap \mathsf{dom}(\sigma) \,.\, \sigma(v) \in \mathrm{R} \\ \phi_{ii}(\mathsf{r}, q) &\triangleq \mathsf{esc}^{\overline{\alpha}}_{\eth^\star}(\mathrm{R}) \subseteq \mathsf{pr}^{-1}_{\eth^\star}(\mathsf{pr}(\eth^\star)) \end{aligned} \quad \text{with} \quad \begin{cases} \eth^\star \triangleq \eth^{\leq q}_{\mathsf{r}}, \ \alpha \triangleq q \bmod 2, \\ \mathrm{R} \triangleq \mathsf{r}^{-1}(q), \ \sigma \triangleq \tau{\upharpoonright}\mathrm{R}. \end{cases}$$

Intuitively, if $\phi_i(\mathsf{r}, \tau, q)$ holds, we are sure that $\sigma \in \mathrm{Str}^{\alpha}(\mathrm{R})$. Moreover, due to Item 2 of Definition 4, R is a subset of a quasi α-dominion having a witness containing σ. Therefore, by Lemma 1, we immediately derives that σ is a witness core for R. Additionally, the formula $\phi_{ii}(\mathsf{r}, q)$ just checks that the second condition of the definition of region is also met.

The following theorem establishes the correctness of the RR approach.

Theorem 1 (Dominion Space). *For a game ∂, the* structure $\mathcal{D} \triangleq \langle \partial, \mathcal{S}, \succ, \Re, \downarrow \rangle$*, where $\mathcal{S}$ is given in Definition 5, $\succ$ is the relation of Definition 6, and $\Re$ and $\downarrow$ are the functions computed by Algorithms 2 and 3 is a dominion space.*

The RR procedure drastically reduces the number of resets needed to solve a game *w.r.t.* PP. In particular, the exponential worst-case game presented in [2] does not work any more, since the execution depth of the associated RR dominion space is only quadratic in the parameter of game family. Unfortunately, at the present time, we are not able to provide a better asymptotic upper bound for the time complexity *w.r.t.* the PP one.

5 Experimental Evaluation

The technique proposed in the paper has been implemented in the tool PG-Solver [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 4 compares the running times of the new algorithm RR against the original version PP 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.

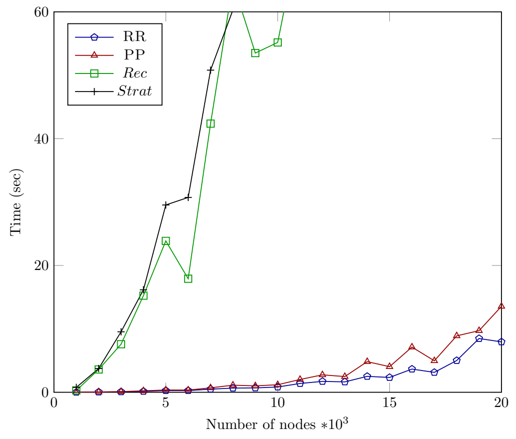

Fig. 4: Comparative results on 2000 random games with up to 20000 positions (from [2]).

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

[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.

$k = n \cdot i/10$ 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). The new solver RR shows a significant improvement on all the benchmarks. 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 4 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 RR 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.

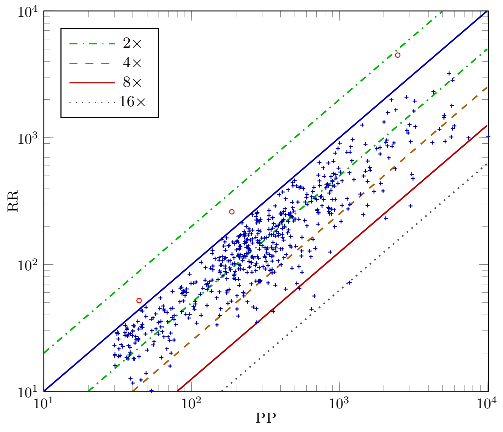

Fig. 5: Comparison between PP and RR on random games with 50000 positions on a logarithmic scale.

To further stress the RR technique in comparison with 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 requires PP between 30 and 6000 seconds. The results comparing PP and RR are reported in Figure 5 on a logarithmic scale. The figure shows that in three cases PP performs better than RR. This is due to the fact that the two algorithms may follow different solution paths within the dominion space and that following the new technique may, in some cases, defer the discovery of a closed dominion. Nonetheless, the RR algorithm does pay off significantly on the vast majority of the benchmarks, often solving a game between two to sixteen times faster than PP.

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 RR as well. As a consequence, RR only requires polynomial time on those games and the experimental results coincide with the ones for PP.

References

1. K. Apt and E. Grädel, *Lectures in Game Theory for Computer Scientists.* Cambridge University Press, 2011.

2. M. Benerecetti, D. Dell'Erba, and F. Mogavero, "Solving parity games via priority promotion," in *CAV'16*, ser. LNCS 9780. Springer, 2016.
3. K. Chatterjee, L. Doyen, T. Henzinger, and J.-F. Raskin, "Generalized Mean-Payoff and Energy Games." in *FSTTCS'10*, ser. LIPIcs 8. Leibniz-Zentrum fuer Informatik, 2010, pp. 505–516.
4. A. Condon, "The Complexity of Stochastic Games." *IC*, vol. 96, no. 2, pp. 203–224, 1992.
5. A. Ehrenfeucht and J. Mycielski, "Positional Strategies for Mean Payoff Games." *IJGT*, vol. 8, no. 2, 1979.
6. E. Emerson and C. Jutla, "The Complexity of Tree Automata and Logics of Programs (Extended Abstract)." in *FOCS'88*. IEEE Computer Society, 1988, pp. 328–337.
7. ——, "Tree Automata, muCalculus, and Determinacy." in *FOCS'91*. IEEE Computer Society, 1991, pp. 368–377.
8. E. Emerson, C. Jutla, and A. Sistla, "On Model Checking for the muCalculus and its Fragments." in *CAV'93*, ser. LNCS 697. Springer, 1993, pp. 385–396.
9. O. Friedmann and M. Lange, "Solving Parity Games in Practice." in *ATVA'09*, ser. LNCS 5799. Springer, 2009, pp. 182–196.
10. E. Grädel, W. Thomas, and T. Wilke, *Automata, Logics, and Infinite Games: A Guide to Current Research.*, ser. LNCS 2500. Springer, 2002.
11. V. Gurevich, A. Karzanov, and L. Khachivan, "Cyclic Games and an Algorithm to Find Minimax Cycle Means in Directed Graphs." *USSRCMMP*, vol. 28, no. 5, pp. 85–91, 1990.
12. M. Jurdziński, "Deciding the Winner in Parity Games is in UP $\cap$ co-Up." *IPL*, vol. 68, no. 3, pp. 119–124, 1998.
13. ——, "Small Progress Measures for Solving Parity Games." in *STACS'00*, ser. LNCS 1770. Springer, 2000, pp. 290–301.
14. M. Jurdziński, M. Paterson, and U. Zwick, "A Deterministic Subexponential Algorithm for Solving Parity Games." *SJM*, vol. 38, no. 4, pp. 1519–1532, 2008.
15. O. Kupferman and M. Vardi, "Weak Alternating Automata and Tree Automata Emptiness." in *STOC'98*. Association for Computing Machinery, 1998, pp. 224–233.
16. O. Kupferman, M. Vardi, and P. Wolper, "An Automata Theoretic Approach to Branching-Time Model Checking." *JACM*, vol. 47, no. 2, pp. 312–360, 2000.
17. A. Mostowski, "Regular Expressions for Infinite Trees and a Standard Form of Automata." in *SCT'84*, ser. LNCS 208. Springer, 1984, pp. 157–168.
18. ——, "Games with Forbidden Positions." University of Gdańsk, Gdańsk, Poland, Tech. Rep., 1991.
19. S. Schewe, "Solving Parity Games in Big Steps." in *FSTTCS'07*, ser. LNCS 4855. Springer, 2007, pp. 449–460.
20. ——, "An Optimal Strategy Improvement Algorithm for Solving Parity and Payoff Games." in *CSL'08*, ser. LNCS 5213. Springer, 2008, pp. 369–384.
21. S. Schewe, A. Trivedi, and T. Varghese, "Symmetric Strategy Improvement." in *ICALP'15*, ser. LNCS 9135. Springer, 2015, pp. 388–400.
22. J. Vöge and M. Jurdziński, "A Discrete Strategy Improvement Algorithm for Solving Parity Games." in *CAV'00*, ser. LNCS 1855. Springer, 2000, pp. 202–215.
23. W. Zielonka, "Infinite Games on Finitely Coloured Graphs with Applications to Automata on Infinite Trees." *TCS*, vol. 200, no. 1-2, pp. 135–183, 1998.
24. U. Zwick and M. Paterson, "The Complexity of Mean Payoff Games on Graphs." *TCS*, vol. 158, no. 1-2, pp. 343–359, 1996.