The Priority Promotion Approach to 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. We consider *parity games*, a special form of two-player infinite-duration games on numerically labeled graphs, whose winning condition requires that the maximal value of a label occurring infinitely often during a play be of some specific parity. We present a new family of algorithms for the solution of the problem, based on the idea of promoting vertices to higher priorities during the search for winning regions. The proposed approach has nice computational properties, exhibiting the best space complexity among the currently known solutions. Experimental results on both random games and benchmark families show that the technique is also very effective in practice.

1 Introduction

Parity games are perfect-information two-player turn-based games of infinite duration, usually played on finite directed graphs. Their vertices, called *positions*, are labeled by natural numbers, called *priorities*, and are assigned to one of two players, named *Even* and *Odd* or, simply, 0 and 1, respectively. The game starts at an arbitrary position and, during its evolution, players can take a move (an outgoing edge) only at their own positions. The moves selected by the players induce an infinite sequence of vertices, called play. If the maximal priority of the vertices occurring infinitely often in the play is *even*, then the play is *winning* for player 0, otherwise, player 1 takes it all. The importance of these games is due to the numerous applications in automata theory and 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 automata theory. In particular, it has been proved to be linear-time interreducible with the model-checking problem for the *modal* μCALCULUS [5]. 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 [7] and still open is the question about the membership in PTIME. The literature on the topic is rich 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* [13] and its *dominion decomposition* [10] and *big step* [11] improvements. The second family, instead, groups together those algorithms that try to compute a winning strategy for the two players on the entire game. Notable members of this category are the *Jurdziński's progress measure* algorithm [8] and the *strategy improvement* approach [12]. Recently, [4] has shown that the problem can be solved in quasi-polynomial time. Despite the theoretical relevance of this result, it does not seem obvious how to turn the employed technique into practical algorithms.

The first two implementations based on that result, described in [9] and [6], are no match for any of the existing exponential algorithms.

This extended abstract presents the ideas underlying a new family of algorithms for solving parity games described in [1–3], based on the notions of *quasi dominion* and *priority promotion*. A quasi dominion Q for player $\alpha \in \{0, 1\}$, called a *quasi α-dominion*, 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 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). Under suitable and easy to check conditions, quasi dominions of the same player can be merged together to form larger quasi α-dominions until, eventually, a dominion for some player is found. The resulting technique, while still exhibiting exponential behaviors, only requires $\mathrm{O}(n \cdot \log k)$ memory (with n and k the number of positions and priorities), which improves on the state-of-the-art. Moreover, experimental results show that the proposed approach performs very well in practice, most often significantly better than existing ones, as extensively shown in [1–3].

2 The Priority Promotion Approach

Preliminaries. A *parity game* is a tuple $\Game = \langle \mathrm{Ps}^{0}, \mathrm{Ps}^{1}, Mv, \mathrm{Pr}, \mathsf{pr} \rangle$, where Ps^{0} and Ps^{1} form a disjoint partitions of the set of *positions* of the game Ps, the left-total relation $Mv \subseteq \mathrm{Ps} \times \mathrm{Ps}$ describes all possible *moves*, $\mathrm{Pr} \subset \mathbb{N}$ is a finite set of *priorities*, and $\mathsf{pr} : \mathrm{Ps} \to \mathrm{Pr}$ is a *priority function* assigning a priority to each position. Player $\alpha \in \{0, 1\}$ controls the positions in Ps^{α}. A *path* in $\mathrm{V} \subseteq \mathrm{Ps}$ is a finite or infinite sequence π of positions in V compatible with the move relation, *i.e.*, $(\pi_i, \pi_{i+1}) \in Mv$. A positional *strategy* σ_α for player α on V is a partial function $\sigma_\alpha \subseteq (\mathrm{V} \cap \mathrm{Ps}^{\alpha}) \rightharpoonup \mathrm{V}$, mapping each α-position v to a position $\sigma_\alpha(v)$, such that $(v, \sigma_\alpha(v)) \in Mv$. A set of positions $\mathrm{V} \subseteq \mathrm{Ps}$ is an *α-dominion*, with $\alpha \in \{0, 1\}$, if there exists a *winning strategy* σ_α in V, *i.e.*, for every infinite path compatible with σ_α, the maximal priority visited infinitely often in V has parity α. By $\Game \setminus \mathrm{V}$ we denote the maximal subgame of $\Game$ 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 collects all positions from which player α can force the game to reach some position in V with a single move. The α-attractor $\mathsf{atr}^{\alpha}(\mathrm{V})$ generalizes the notion of α-predecessor to an arbitrary number of moves. Finally, an *α-escaping position* of V is a position in V from which α can leave the set in one move.

Priority Promotion. The priority promotion approach attacks the problem of solving a parity game $\Game$ by computing one of its dominions D, for some player α, at a time. Indeed, once the α-attractor $\mathrm{D}^{\star}$ of D is removed from $\Game$, the smaller game $\Game \setminus \mathrm{D}^{\star}$ is obtained, whose positions are winning for one player iff they are winning for the same player in the original game. This allows for decomposing the problem of solving a parity game into iteratively finding its dominions [10]. 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 the set of escape positions.

Definition 1 (Quasi Dominion [3]). *A non-empty set of positions Q is a* quasi α-dominion *in $\Game$ for a player α if there exists an α-strategy whose induced plays are either infinite and winning for α or finite and end in an $\overline{\alpha}$-escaping position.*

Observe that, if all induced plays remain in the set Q forever, this is actually an α-dominion and, thus, a subset of the α-winning region. In this case, the escape set of Q is empty and we say that Q is *α-closed*. If this is not the case, we say that it is *α-open*.

The priority promotion algorithm explores a partial order $\langle \mathrm{S}, \prec \rangle$, 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 $\top \in \mathrm{S}$ of the order, in which the quasi dominions are initialized to the sets of positions with the same priority. At each step, a new quasi dominion is extracted from the current state, by means of a *query* operator $\Re$, and used to compute a successor state, by means of a *successor* operator $\downarrow$, if the quasi dominion is open. If, on the other hand, it is closed, the search is over. The partial order $\langle \mathrm{S}, \top, \prec \rangle$ together with the operators $\Re$ and $\downarrow$ form what is called a *dominion space*. The notion of dominion space is quite general and can be instantiated in different ways, by providing specific query and successor operators (see [1–3]). The overall procedure is sound and complete on any dominion space and its time complexity is linear in the *execution depth* of the space, namely the length of the longest chain in the underlying partial order compatible with the successor operator. Its space complexity, instead, is only logarithmic in the *size* of the dominion-space, since only one state at the time needs to be maintained.

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 processes the input game in descending order of priority and, at each step, a subgame of the entire game, obtained by removing the quasi dominions previously computed at higher priorities, is considered. At each priority of parity α: *(i)* a quasi α-domain Q is extracted by the query operator from the current subgame; *(ii)* if Q is closed in the entire game, the search stops and returns Q as result; otherwise, *(iii)* 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, possibly closed, bigger one.

The solution of the first problem relies on the definition of a specific class of quasi dominions, called *regions*. An α-region R of a game ∂ is a special form of quasi α-dominion of ∂ with the additional requirement that all its escape positions 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 2 (Region [3]). *A quasi α-dominion R is an α-region in ∂ if $\mathsf{pr}(\partial) \equiv_2 \alpha$ and all its $\overline{\alpha}$-escape positions have priority $\mathsf{pr}(\partial)$.*

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. A closed α-region in a game is clearly an α-dominion in that game. In addition, the α-attractor of an α-region is always an α-region. 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. 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 $\mathsf{r} : \mathrm{Ps} \rightarrow \mathrm{Pr}$, called *region function*. Given a priority p, we denote by $\mathsf{r}^{(\geq p)}$ (*resp.*, $\mathsf{r}^{(>p)}$ and $\mathsf{r}^{(<p)}$) the function obtained by restricting the domain of r to the positions with measure greater than or equal to p (*resp.*, greater than and lower than p). Moreover, the set of pairs (R, α), where R is an α-region, is denoted by Rg, and is partitioned into the sets Rg^{-} and Rg^{+} of open and closed α-region pairs, respectively.

Algorithm 1 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 $s \triangleq (\mathsf{r}, p)$. Line 2, instead, computes the attractor *w.r.t.* player α in subgame $\Game_s \triangleq \Game \setminus \mathsf{dom}\big(\mathsf{r}^{(>p)}\big)$ of the region contained in r at the current priority p. The resulting set R is an α-region of $\Game_s$ containing $\mathsf{r}^{-1}(p)$, as the following statement asserts.

Algorithm 1: Query Function.

signature $\Re_{\Game} : \mathrm{S}_{\Game} \rightarrow 2^{\mathrm{Ps}_{\Game}} \times \{0, 1\}$
function $\Re_{\Game}(s)$
  let $(\mathsf{r}, p) = s$ **in**
1    $\alpha \leftarrow p \bmod 2$
2    $\mathrm{R} \leftarrow \mathsf{atr}^{\alpha}_{\Game_s}(\mathsf{r}^{-1}(p))$
3 **return** (R, α)

Proposition 1 (Region Extension [3]). *The attractor* $\mathsf{atr}^{\alpha}(\mathrm{R})$ *of an α-region* R *in* $\Game$ *is an 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.

Algorithm 2: Successor Function.

signature $\downarrow_{\Game} : \succ_{\Game} \rightarrow \Delta_{\Game} \times \mathrm{Pr}_{\Game}$
function $s \downarrow_{\Game} (\mathrm{R}, \alpha)$
  let $(\mathsf{r}, p) = s$ **in**
    if $(\mathrm{R}, \alpha) \in \mathrm{Rg}^{-}_{\Game_s}$ **then**
      $\mathsf{r}^{\star} \leftarrow \mathsf{r}[\mathrm{R} \mapsto p]$
      $p^{\star} \leftarrow \max(\mathsf{rng}\Big(\mathsf{r}^{\star(<p)}\Big))$
    else
      $p^{\star} \leftarrow \mathsf{bep}^{\overline{\alpha}}_{\Game}(\mathrm{R}, \mathsf{r})$
      $\mathsf{r}^{\star} \leftarrow \mathsf{pr}_{\Game} \uplus \mathsf{r}^{(\geq p^{\star})}[\mathrm{R} \mapsto p^{\star}]$
  return $(\mathsf{r}^{\star}, p^{\star})$

Proposition 2 (Region Merging [3]). *The union of an α-region* R *in* $\Game$ *and a α-dominion in the subgame* $\Game \setminus \mathrm{R}$ *is an α-region in* $\Game$.

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 [3] 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. The reset is in general unavoidable, since the new promoted region may attract positions from lower ones, thereby potentially invalidating their status as regions. In some cases, indeed, the player who wins by remaining in the region may even change from α to $\overline{\alpha}$.

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 from the α-region R, closed in the current subgame, that needs to be promoted. Algorithm 2 reports the pseudo-code of the successor function. Given the current state s and a region pair (R, α) open in the whole game as inputs, it produces a successor state $s^\star \triangleq (\mathsf{r}^\star, p^\star)$ in the dominion space. It first checks whether R is open also in the subgame $\eth_s$ (Line 1). If this is the case, it assigns measure p to region R and stores it in the new region function $\mathsf{r}^\star$ (Line 2). The new current priority $p^\star$ is, then, computed as the highest priority lower than p in $\mathsf{r}^\star$ (Line 3). If, on the other hand, R is closed in $\eth_s$, a promotion, merging R with some other α-region contained in r, is required. The next priority $p^\star$ is set to the best escape priority 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 the regions with lower measure than $p^\star$ in the region function r are reset by means of the operator $\mathsf{f} \uplus \mathsf{g}$, which complete the missing values for the domain of g with the values from f.

The result is a sound and complete solution procedure, as stated in the following.

Theorem 1. *The Priority Promotion algorithm is correct and solves a game with $n \in \mathbb{N}$ positions and $k \in [1, n]$ priorities in $\mathrm{O}(n \cdot \log k)$ space.*

References

1. M. Benerecetti, D. Dell'Erba, and F. Mogavero, "A Delayed Promotion Policy for Parity Games." in *GANDALF16*, ser. EPTCS 226, 2016, pp. 30–45.
2. ——, "Improving Priority Promotion for Parity Games." in *HVC'16*, ser. LNCS 10028. Springer, 2016, pp. 1–17.
3. ——, "Solving Parity Games via Priority Promotion." in *CAV'16*, ser. LNCS 9780 (Part II). Springer, 2016, pp. 270–290.
4. C. Calude, S. Jain, B. Khoussainov, W. Li, and F. Stephan, "Deciding Parity Games in Quasipolynomial Time." in *STOC'17*. Association for Computing Machinery, 2017, pp. ?–?, accepted for publication.
5. 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.
6. J. Fearnley, S. Jain, S. Schewe, F. Stephan, and D. Wojtczak, "An Ordered Approach to Solving Parity Games in Quasi Polynomial Time and Quasi Linear Space." in *SPIN'2017*. Association for Computing Machinery, 2017, pp. ?–?, accepted for publication.
7. M. Jurdziński, "Deciding the Winner in Parity Games is in UP ∩ co-UP." *IPL*, vol. 68, no. 3, pp. 119–124, 1998.
8. ——, "Small Progress Measures for Solving Parity Games." in *STACS'00*, ser. LNCS 1770. Springer, 2000, pp. 290–301.
9. M. Jurdziński and R. Lazic, "Succinct Progress Measures for Solving Parity Games." in *LICS'17*. Association for Computing Machinery, 2017, pp. ?–?, accepted for publication.
10. M. Jurdziński, M. Paterson, and U. Zwick, "A Deterministic Subexponential Algorithm for Solving Parity Games." *SJM*, vol. 38, no. 4, pp. 1519–1532, 2008.
11. S. Schewe, "Solving Parity Games in Big Steps." in *FSTTCS'07*, ser. LNCS 4855. Springer, 2007, pp. 449–460.
12. 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.
13. 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.