The Rabin Index of Parity Games

January 17, 2012

Michael Huth and Jim Huan-Pu Kuo
Department of Computing
Imperial College London
London, SW7 2AZ, United Kingdom
{m.huth, jimhkuo}@imperial.ac.uk

Nir Piterman
Department of Computer Science
University of Leicester
Leicester, LE1 7RH, United Kingdom
nir.piterman@leicester.ac.uk

Abstract

We study the descriptive complexity of parity games by taking into account the coloring of their game graphs whilst ignoring their ownership structure. Different colorings of the same graph are identified if they determine the same winning regions and strategies, for *all* ownership structures of nodes. The Rabin index of a parity game is the minimum of the maximal color taken over all equivalent coloring functions. We show that deciding whether the Rabin index is at least k is in P for $k = 1$ but NP-hard for all *fixed* $k \geq 2$. We present an EXPTIME algorithm that computes the Rabin index by simplifying its input coloring function. When replacing simple cycle with cycle detection in that algorithm, its output over-approximates the Rabin index in polynomial time. Experimental results show that this approximation yields good values in practice.

1 Introduction

Parity games (see e.g. [1]) are infinite, 2-person, 0-sum, graph-based games that are hard to solve. Their nodes are colored with natural numbers, controlled by different players, and the winning condition of plays depends on the minimal color occurring in cycles. The condition for winning a node, therefore, is an alternation of existential and universal quantification. In practice, this means that the maximal color of its coloring function is the only exponential source for the worst-case complexity of most parity game solvers, e.g. for those in [1, 2, 3].

One approach taken in analyzing the complexity of parity games, and in so hopefully improving the complexity of their solution, is through the study of the descriptive complexity of their underlying game graph. This method therefore ignores the ownership structure on parity games.

An example of this approach is the notion of DAG-width in [4]. Every directed graph has a DAG-width, a natural number that specifies how well that graph can be decomposed into a directed acyclic graph (DAG). The decision problem for DAG-width, whether the DAG-width of a directed graph is at most k, is NP-complete in k [4]. But parity games whose DAG-width is below a given threshold have polynomial-time solutions [4]. The latter is a non-trivial result since DAG-width also ignores the colors of a parity game.

In this paper we want to develop a similar measure of the descriptive complexity of parity games, their *Rabin index*, a natural number that ignores the ownership of nodes, but does take into account the colors of a parity game. Intuitively, the Rabin index is the number of colors that are *required* to capture the complexity of the game structure. By measuring and reducing the number of colors we hope to improve the complexity of analyzing parity games. [1] The reductions we propose are related to priority compression and propagation in [5] but, in contrast, exploit the *cyclic* structure of game graphs.

The name for the measure developed here is inspired by related work on the Wagner hierarchy for automata on infinite words [6]: Carton and Maceiras use similar ideas to compute and minimize the Rabin index of deterministic parity automata on infinite words [7]. To the best of our knowledge, our work is the first to study this notion in the realm of infinite, 2-person games.

The idea behind our Rabin index is that one may change the coloring function of a parity game to another one if that change neither affects the winning regions nor the choices of winning strategies. This yields an equivalence relation between coloring functions. For the coloring function of a parity game, we then seek an equivalent coloring function with the smallest possible maximal color, and call that minimal maximum the Rabin index of the respective parity game.

The results we report here about this Rabin index are similar in spirit to those developed for DAG-width in [4] but there are important differences:

- We propose a measure of descriptive complexity that is closer to the structure of the parity game as it only forgets ownership of nodes and not their colors.
- We prove that for every *fixed* $k \geq 2$, deciding whether the Rabin index of a parity game is at least k is NP-hard.
- We can characterize the above equivalence relation in terms of the parities of minimal colors on *simple* cycles in the game graph.

[1] We note that if we also were to account for ownership, we could solve the parity game and assign color 0 to nodes won by player 0 and color 1 to nodes won by player 1. Thus, this would reduce the index of *all* games to at most 2. However, this would prevent a more fine-grained analysis of the structural complexity of the game.

- We use that characterization to design an algorithm that computes the Rabin index and a witnessing coloring function in exponential time.
- We show how the same algorithm efficiently computes sound approximations of the Rabin index when simple cycles are abstracted by cycles.
- We derive from that approximation an abstract Rabin index of parity games such that games with bounded abstract Rabin index are efficiently solvable.
- We conduct detailed experimental studies that corroborate the utility of that approximation, also as a preprocessor for solvers.

Outline of paper. Section 2 contains background for our technical develeopments. In Section 3, we define the equivalence between coloring functions, characterize it in terms of simple cycles, and use that characterization to define the Rabin index of parity games. In Section 4 we develop an algorithm that runs in exponential time and computes a coloring function which witnesses the Rabin index of the input coloring function. The complexity of the natural decision problems for the Rabin index is studied in Section 5. An abstract version of our algorithm is shown to soundly approximate that coloring function and Rabin index in Section 6. Section 7 contains our experimental results for this abstraction. And we conclude the paper in Section 8.

2 Background

We write $\mathbb{N}$ for the set $\{0, 1, \dots\}$ of natural numbers. A parity game G is a tuple (V, V_0, V_1, E, c) where V is a non-empty set of nodes partitioned into possibly empty node sets V_0 and V_1, with an edge relation $E \subseteq V \times V$ (where for all v in V there is a w in V with (v, w) in E), and a coloring function $c \colon V \to \mathbb{N}$.

Throughout, we write s for one of 0 or 1. In a parity game, player s owns the nodes in V_s. A play from some node v_0 results in an infinite play $P = v_0 v_1 \dots$ in (V, E) where the player who owns v_i chooses the successor v_{i+1} such that (v_i, v_{i+1}) is in E. Let $\mathsf{Inf}(P)$ be the set of colors that occur in P infinitely often: $\mathsf{Inf}(P) = \{k \in \mathbb{N} \mid \forall j \in \mathbb{N} \colon \exists i \in \mathbb{N} \colon i > j \text{ and } k = c(v_i)\}$. Player 0 wins play P iff $\min \mathsf{Inf}(P)$ is even; otherwise player 1 wins play P.

A strategy for player s is a total function $\tau \colon V_s \to V$ such that $(v, \tau(v))$ is in E for all $v \in V_s$. A play P is consistent with τ if each node v_i in P owned by player s satisfies $v_{i+1} = \tau(v_i)$. It is well known that each parity game is determined: node set V is the disjoint union of two sets W_0 and W_1, the winning regions of players 0 and 1 (respectively), where one of W_0 and W_1 may be empty. Moreover, strategies $\sigma \colon V_0 \to V$ and $\pi \colon V_1 \to V$ can be computed such that

- all plays beginning in W_0 and consistent with σ are won by player 0; and
- all plays beginning in W_1 and consistent with π are won by player 1.

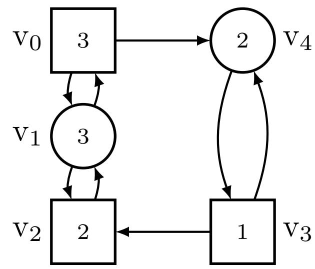


Figure 1: A parity game with winning regions $W_0 = \{v_1, v_2\}$ and $W_1 = \{v_0, v_3, v_4\}$; winning strategies for players 0 and 1 map v_1 to v_2, respectively v_0 and v_3 to v_4

Solving a parity game means computing such data (W_0, W_1, σ, π). We show a parity game and one of its possible solutions in Figure 1.

3 Rabin Index

We now formalize the definition of equivalence for coloring functions, and then use that notion in order to formally define the Rabin index of a parity game.

We want to reduce the complexity of a coloring function c in a parity game (V, V_0, V_1, E, c) by transforming c to some coloring function c'. Since we also want that transformation to ignore ownership of nodes, it needs to be sound for *every possible* ownership structure $V_0 \cup V_1 = V$. Therefore, for *all* such partitions $V_0 \cup V_1 = V$, the two parity games (V, V_0, V_1, E, c) and (V, V_0, V_1, E, c') that differ only in colors need to be equivalent in that they have the same winning regions and the same sets of winning strategies. We formalize this notion.

Definition 1 *Let (V, E) be a directed graph and $c, c' \colon V \to \mathbb{N}$ two coloring functions. We say that c and c' are equivalent, written $c \equiv c'$, iff for all partitions $V_0 \cup V_1$ of V the resulting parity games (V, V_0, V_1, E, c) and (V, V_0, V_1, E, c') have the same winning regions and the same sets of winning strategies for both players.*

Intuitively, changing coloring function c to c' with $c \equiv c'$ is sound: regardless of what the actual partition of V is, we know that this change will neither affect the winning regions nor the choice of their supporting winning strategies. But the definition of $\equiv$ is not immediately amenable to algorithmic simplification of c to some c'. This definition quantifies over exponentially many partitions, and for each such partition it insists that certain sets of strategies be equal.

We need a more compact characterization of $\equiv$ as the basis for designing a static analysis. To that end, we require some concepts from graph theory first.

Definition 2 1. *A path P in a directed graph (V, E) is a sequence $v_0, v_1, \ldots, v_n$ of nodes in V such that (v_i, v_{i+1}) is in E for every i in $\{0, 1, \ldots, n-1\}$.*

2. *A cycle C in a directed graph (V, E) is a path $v_0, \ldots, v_n$ with (v_n, v_0) in E.*
3. *A simple cycle C in a directed graph (V, E) is a cycle $v_0, v_1, \ldots, v_n$ such that for every $i \neq j$ in $\{0, 1, \ldots n\}$ we have $v_i \neq v_j$.*
4. *For (V, E, c), the c-color of a cycle $v_0, \ldots, v_n$ in (V, E) is $\min_{0 \leq i \leq n} c(v_i)$.*

Simple cycles are paths that loop so that no node has more than one outgoing edge on that path. A cycle is defined similarly, except that it is allowed that v_i equals v_j for some $i \neq j$, so a node on that path may have more than one outgoing edge. The color of a cycle is the minimal color that occurs on it.

For example, for the parity game in Figure 1, a simple cycle is v_0, v_4, v_3, v_2, v_1 and its color is 1, a cycle that is not simple is v_0, v_1, v_2, v_1 and its color is 2.

We can now characterize $\equiv$ in terms of colors of simple cycles. Crucially, we make use of the fact that parity games have pure, positional strategies [8].

Proposition 1 *Let (V, E) be a directed graph and $c, c' \colon V \to \mathbb{N}$ two coloring functions. Then $c \equiv c'$ iff for all simple cycles C in (V, E), the c-color of C has the same parity as the c'-color of C.*

Proof. Let us write $c \sim c'$ iff for all simple cycles C in (V, E), the c-color of C has the same parity as the c'-color of C. We have to show that $\sim$ equals $\equiv$.

1. We show that $\sim$ is contained in $\equiv$. Let $c \sim c'$ be given. We want to show $c \equiv c'$. So let $V_0 \subseteq V$ be given. Consider the two derived parity games (V, V_0, V_1, E, c) and (V, V_0, V_1, E, c'). Let W_0 be the winning region of player 0 in the parity games (V, V_0, V_1, E, c) and σ a strategy for player 0 winning for player 0 on W_0 in (V, V_0, V_1, E, c).

Now consider an arbitrary strategy π for player 1. Then π is such a strategy in both parity games (V, V_0, V_1, E, c) and (V, V_0, V_1, E, c'). Let $v \in W_0$ and let P be the play in (V, E) that begins in v and is consistent with σ and π. Since P is consistent with deterministic strategies of both players, its ultimately periodic behavior determines a simple cycle C so that P is composed of a finite prefix and infinitely many repetitions of C. Since v is in W_0 and since σ is winning for player 0 in W_0, we infer that the c-color of C has to be even. Since $c \sim c'$, this means that the c'-color of C is even, too. And so that play is also won by player 0 in (V, V_0, V_1, E, c').

Since π was arbitrary, this shows that σ is also a winning strategy on W_0 in the parity game (V, V_0, V_1, E, c'). Therefore, W_0 is a subset of the winning region W'_0 of player 0 in (V, V_0, V_1, E, c').

A symmetric argument for winning region W_1 in (V, V_0, V_1, E, c) for player 1 and winning strategy π for player 1 on W_1 in that game shows that π is also a winning strategy on W_1 in (V, V_0, V_1, E, c') and that W_1 is contained in W'_1, the winning region of player 1 in (V, V_0, V_1, E, c').

Combining these two insights, and since V equals $W_0 \cup W_1$, it follows that W_0 equals W'_0 and that W_1 equals W'_1. So the winning regions are equal in (V, V_0, V_1, E, c) and (V, V_0, V_1, E, c'), and strategies that are winning on these sets in one of the games (V, V_0, V_1, E, c) and (V, V_0, V_1, E, c') are also winning on these sets in the other game since $c \sim c'$. (We showed this for one player, but the result follows for the other player by symmetry.)

2. We show that $\equiv$ is contained in $\sim$. Let $c \equiv c'$ be given. Let C be a simple cycle in (V, E). Let the parity of the c-color of C be even. (The case when this is odd is proved symmetrically and so we omit that proof.) Consider the parity games $(V, V, \emptyset, E, c)$ and $(V, V, \emptyset, E, c')$ where V_0 is defined to be V, and so V_1 is empty. Since V_0 equals V, player 0 has some strategy σ such that $\sigma(v)$ is again in C for all nodes v from C. Since the c-parity of C is even, it then follows that C is contained in W_0, the winning region of player 0 in (V, V_0, V_1, E, c).

Since $c \equiv c'$ is assumed, we therefore know that W_0 is also the winning region of player 0 in $(V, V, \emptyset, E, c')$, and that σ is also a winning strategy on W_0 in that game. In particular, every play beginning in some node v from C and consistent with σ is won by player 0 in $(V, V, \emptyset, E, c')$. But every such play just repeats the simple cycle C infinitely often (it cannot generate a sub-cycle of C as σ is deterministic and C is simple) and so the outcome of that play is determined by the c'-color of C. Therefore, the c'-color of C has to be even. $\square$

Next, we define the relevant measure of descriptive complexity, which will also serve as a measure of precision for the static analyses we will develop.

Definition 3 *1. For colored arena (V, E, c), its index $\mu(c)$ is $\max_{v \in V} c(v)$.*

2. The Rabin index $\mathsf{RI}(c)$ of colored arena (V, E, c) is $\min\{\mu(c') \mid c \equiv c'\}$.

3. The Rabin index of parity game (V, V_0, V_1, E, c) is $\mathsf{RI}(c)$ for (V, E, c).

The index $\mu(c)$ reflects the maximal color occurring in c. So for a coloring function $c\colon V \to \mathbb{N}$ on (V, E), its Rabin index is the minimal possible maximal color in a coloring function that is equivalent to c. This definition applies to colored arenas and parity games alike.

As an aside, is $\mu(c)$ a good measure, given that $\mu(c + n) = n + \mu(c)$ for $c + n$ with $(c + n)(v) = c(v) + n$ when n is even? And given that c may have large color gaps? Fortunately, this is not a concern for the Rabin index of c. This is so as for all $c' \equiv c$ with $\mu(c') = \mathsf{RI}(c)$ we know that the minimal color of c' is at most 1 and that c' has no color gaps – due to the minimality of the Rabin index.

Intuitively, in order to prove that $\mathsf{RI}(c) < k$ for some $k > 0$ one has to produce a coloring c' and show that all simple cycles in the graph have the same color under c and c'. As we will see below, deciding for a given colored arena (V, E, c) whether $\mathsf{RI}(c)$ is at least k is NP-hard for fixed $k \geq 2$.

Next, we present an algorithm that computes a coloring function which witnesses the Rabin index of a given c.

4 Computing the Rabin Index

We now discuss our algorithm `rabin`, shown in Figure 2. It takes a coloring function as input and outputs an equivalent one whose index is the Rabin index of the input. Formally, `rabin` computes a coloring function c' with $c \equiv c'$ and where there is no $c \equiv c''$ with $\mu(c'') < \mu(c')$. Then, $\mathsf{RI}(c) = \mu(c')$ by definition.

Algorithm `rabin` uses a standard iteration pattern based on a rank function which sums up all colors of all nodes. In each iteration, two methods are called:

- `cycle` analyzes the cyclic structure of (V, E) and so reduces colors of nodes
- `pop` repeatedly lowers all occurrences of maximal colors by 1 until there is a simple cycle whose color is a maximal color.

These iterations proceed until neither `cycle` nor `pop` has an effect on the coloring function. Method `cycle` first sorts all nodes of (V, E, c) in ascending color values for c. It then processes each node v_i in that ascending order. For each node v_i it calls `getAnchor` to find (if possible) a maximal "anchor" for v_i.

If `getAnchor` returns -1, then v_i has no anchor as all simple cycles through v_i have color $c(v_i)$. Therefore, it is sound to change $c(v_i)$ to its parity. Otherwise, `getAnchor` returns an index j to an "anchor" node that is maximal in that

- there is a simple cycle C through v_i whose color j is smaller and of different parity than that of v_i, and
- for all simple cycles C' through v_i, either they have a color that has the same parity as the color of v_i or they have a color that is less than or equal to j.

A node on this simple cycle C with color j is thus a maximal anchor for node v_i. Method `cycle` therefore resets $c(v_i)$ to $j + 1$.

The idea behind `pop` is that one can safely lower maximal color m to $m - 1$ if there is no simple cycle whose color is m. For then all occurrences of m are dominated by smaller colors on simple cycles.

We now prove the soundness of our algorithm `rabin`.

Lemma 1 *Let (V, E, c) be a given colored arena and let c' be the coloring function that is returned by the call* `rabin`(V, E, c). *Then $c \equiv c'$ holds.*

Proof. Let $c = c_0, c_1, \ldots$ be the sequence of coloring functions that reflect the state changes of c in the call `rabin`(V, E, c). By Proposition 1, it suffices to show that $c_n \sim c_{n+1}$ for all such n. So let c_n be given.

1. Consider first the case when c_{n+1} is obtained from c_n by an execution of the for-statement in `pop`. Then m is the maximal color of c_n but there is no simple cycle in (V, E) that has c_n-color m. In other words, color m will never decide the c_n-color of a simple cycle. It is therefore safe to decrease all

```
rabin(V,E,c) {
  rank = ∑_{v∈V} c(v);
  do {
    cache = rank;
    cycle(); pop();
    rank = ∑_{v∈V} c(v);
  } while (cache != rank)
  return c;
}

cycle() {
  sort V in ascending c-color ordering v_1,v_2,...,v_n;
  for (i=1..n) {
    j = getAnchor(v_i);
    if (j == -1) { c(v_i) = c(v_i) % 2; }
    else { c(v_i) = j + 1; }
  }
}

getAnchor(v_i) {
  for (γ = c(v_i) - 1  down to (c(v_i) - 1) % 2;  step size 2) {
     if (∃ simple cycle C with color γ through  v_i) { return γ; }
  }
  return -1;
}

pop() {
  m = max{ c(v) |  v ∈ V};
  while (not ∃ simple cycle C with color m) {
    for (v in { w ∈  V |  c(w)  =  m}) { c(v)  =  m  -  1; }
    m  =  m  -  1;
  }
}
```

Figure 2: Algorithm `rabin` which relies on methods `cycle`, `getAnchor`, and `pop`.

occurrences of m to $m-1$, as this will change the color of no simple cycle in (V, E). Since this change defines c_{n+1}, we have $c_i \sim c_{n+1}$ as desired.

2. Now consider the case when c_{n+1} is the result of c_n through the execution of the if-branch in `cycle`. Then we consider a node v_i for which `getAnchor` returns -1. Therefore, there is no simple cycle C through v_i in (V, E) whose c_n-color is lower than $c_n(v_i)$ and has different parity than $c_n(v_i)$. But the color of cycles through v_i can be at most $c_n(v_i)$. Therefore, all simple cycles through v_i have the same parity as $c_n(v_i)$. It is therefore safe to reduce the color at v_i to that parity, as done in `cycle`. For the resulting c_{n+1} we therefore have $c_n \sim c_{n+1}$.

3. Now consider the case when c_{n+1} is the result of c_n through the execution of the else-branch in `cycle`. If the call to `getAnchor` returns $j \geq 0$ for node v_i, then consider an arbitrary simple cycle C in (V, E) through v_i whose color p has a parity other than that of $c_n(v_i)$. Then it must be that $j \leq p$ by the definition of method `getAnchor`. So every simple cycle through v_i has either a color that has the parity of $c_n(v_i)$ or has a color p with $j \leq p$. Therefore, it is safe to change the color at v_i to $j+1$ (the case $j+1 = c_n(v_i)$ will have no effect), resulting in new coloring function c_{n+1}: this is so since then all simple cycles through v_i have the same parity with respect to c_n and c_{n+1}. (And both coloring functions could only break $c_n \sim c_{n+1}$ by means of simple cycles through v_i.) $\square$

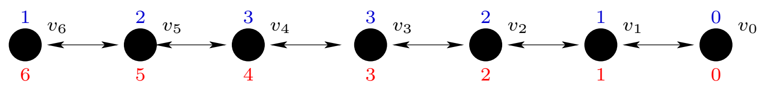
We show some example runs of `rabin`, starting with a detailed worked example, for the parity game in Figure 1. Let the initial sort of `cycle` be $v_3v_4v_2v_0v_1$. Then `cycle` changes no colors at v_3 (as the anchor of v_3 is -1), at v_4 (as the anchor of v_4 is 1 due to simple cycle v_4v_3), at v_2 (as the anchor of v_2 is 1 due to simple cycle $v_2v_1v_0v_4v_3$), but changes $c(v_0)$ to 1 (as the anchor of v_0 is -1). Also, $c(v_1)$ won't change (as the anchor of v_1 is 2 due to simple cycle v_1v_2).

Then `pop` changes $c(v_1)$ to 2 (as there is no simple cycle with color 3). Let the sort of the second call to `cycle` be $v_0v_3v_1v_2v_4$. Then the corresponding list of anchor values is $-1, -1, 1, 1, 1$ and so `cycle` changes no colors. Therefore, the second call to `pop` changes no colors either. Thus the overall effect of `rabin` was to lower the index from 3 to 2 by lowering $c(v_1)$ to 2.

As a second example, in Figure 3, we see a colored arena with $c(v_i) = i$ (in red/bottom), the output `rabin`(V, E, c) (in blue/top), and a table showing how the coloring function changes through repeated calls to `cycle` and `pop`. Each iteration of `rabin` reduces the measure $\mu(c)$ by 1. This illustrates that the number of iterations of `rabin` is unbounded in general.

We note that $\equiv$ cannot be captured by just insisting that the winning regions of all abstracted parity games be the same. In Figure 4, we see a colored arena with two coloring functions c (in blue/top) and c' (in red/bottom).

The player who owns node v will win all nodes as she chooses between z or o the node that has her parity. So c and c' are equivalent in that they always give rise to the same winning regions. But if v is owned by player 1, that player has a winning strategy for c that moves from v to w, but this is not winning for c'.



iteration	`cycle`	`pop`
1	*nil*	$c(v_6) = 5$
2	$c(v_6) = 1$	$c(v_5) = 4$
3	$c(v_5) = 2$	$c(v_4) = 3$

Figure 3: Colored arena (V, E, c) and table showing effects of iterations in $\texttt{rabin}(V, E, c)$

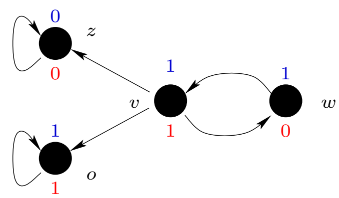


Figure 4: Two coloring functions c (in blue/top) and c' (in red/bottom) that give rise to the same winning regions, but not always to the same winning strategies. Thus $c \not\equiv c'$

In Figure 5, colored arena (V, E, c) has odd index n and Rabin index 2. Although there are cycles from all nodes with color n, e.g., to the node with color $n-1$, there are no *simple* such cycles. So all colors reduce to their parity. As before, the number of iterations is linear in the size of the graph.

Now we can prove that algorithm `rabin` is basically as precise as it could be. First, we state and prove an auxilliary lemma which provides sufficient conditions for a coloring function c to have its index $\mu(c)$ as its Rabin index $\mathsf{RI}(c)$. Then we show that the output of `rabin` meets these conditions.

Lemma 2 *Let (V, E, c) be a colored arena where*

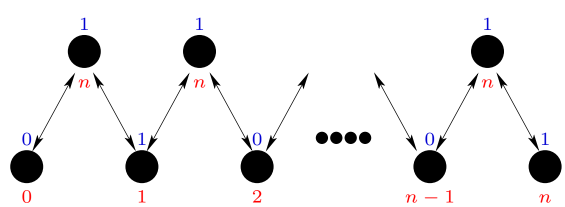


Figure 5: Coloring function c (red/bottom) has Rabin index 2, witnessed by c' (blue/top)

1. *there is a simple cycle in (V, E) whose color is the maximal one of c*
2. *for all v in V with $c(v) > 1$, node v is on a simple cycle C with color $c(v) - 1$.*

Then there is no c' with $c \equiv c'$ and $\mu(c') < \mu(c)$. And so $\mu(c)$ equals $\mathsf{RI}(c)$.

Proof. Let k be the maximal color of c and consider an arbitrary c' with $c \equiv c'$.

Proof by contradiction: Let the maximal color k' of c' satisfy $k' < k$. By the first assumption, there is a simple cycle C_0 whose c-color is k. Since $k' < k$ and $c \equiv c'$, we know that the c'-color of C_0 can be at most $k - 2$. Let v_0 be a node on C_0 such that $c'(v_0)$ is the c'-color of C_0. Then $c'(v_0) \leq k - 2$. As all nodes on C_0 have c-color k, we have also $c(v_0) \geq k$. For $k < 2$, then $c'(v_0) \leq k - 2$ gives us a contradiction $c'(v_0) < 0$. It thus remains to consider the case when $k \geq 2$.

By the second assumption, there is some simple cycle C_1 through v_0 such that the color of C_1 is $k - 1$. In particular, there is some node v'_0 in C_1 with color $k - 1$. But $k - 1$ cannot be the color of C_1 with respect to c' since v_0 is on C_1 and $c'(v_0) \leq k - 2$. Since $c \equiv c'$, the c'-color of C_1 is therefore at most $k - 3$. So there is some v_1 on C_1 such that $c'(v_1) \leq k - 3 < k - 1 \leq c(v_1)$.

If $c(v_1) > 1$, we repeat the above argument at node v_1 to construct a simple cycle C_2 through v_1 with color $c(v_1) - 1$. Again, there then have to be nodes v'_1 and v_2 on C_2 such that the color $c'(v'_1)$ is the c'-color of C_2, and such that $c'(v_2) \leq k - 4 < k - 2 \leq c(v_2)$ holds.

We can repeat the above argument to construct simple cycles $C_0, C_1, C_2, \ldots$ and nodes $v_0, v'_0, v_1, v'_1, v_2, v'_2, \ldots$ such that $c'(v_j) \leq k - j - 2 < k - j \leq c(v_j)$ until $k - j \leq c(v_j) \leq 1$. But then $c'(v_j) \leq k - j - 2 \leq 1 - 2 = -1$, a contradiction. □

We now show that the output of `rabin` satisfies the assumptions of Lemma 2. Since `rabin` is sound for $\equiv$, we therefore infer that it computes a coloring function whose maximal color equals the Rabin index of its input coloring function.

Theorem 1 *Let (V, E, c) be a colored arena. And let c^* be the output of the call* `rabin`(V, E, c)*. Then $c \equiv c^*$ and $\mu(c^*)$ is the Rabin index of c.*

Proof. By Lemma 1, we have $c \equiv c^*$. Since $\equiv$ is clearly transitive, it suffices to show that there is no c' with $c^* \equiv c'$ and $\mu(c') < \mu(c^*)$. By Lemma 2, it therefore suffices to establish the two assumptions of that lemma for c^*. What we do know is that neither `cycle` nor `pop` have an effect on c^* as it was returned by `rabin`.

The first assumption of Lemma 2 is therefore true since `pop` has no effect on c^* and so there must be a simple cycle in (V, E) whose color is the maximal one in c. (This also applies to the boundary case when c^* has only one color, as (V, E) has to contain cycles since it is finite and all nodes have outgoing edges.)

As for the second assumption, let by way of contradiction there be some node v with $c^*(v) > 1$ and no simple cycle through v with color $c^*(v) - 1$. Then `cycle` would have an effect on $c^*(v)$ and would lower it, a contradiction. □

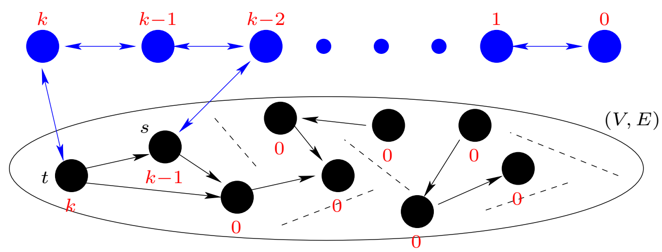


Figure 6: Construction for NP-hardness of deciding whether $\mathsf{RI}(c) \geq k$ for $k \geq 2$

5 Complexity

We now discuss the complexity of algorithm `rabin` and of the decision problems associated with the Rabin index. We turn to the complexity of `rabin` first.

Let us assume that we have an oracle that checks for the existence of simple cycles. Then the computation of `rabin` is efficient modulo polynomially many calls (in the size of the game) to that oracle. Since deciding whether a simple cycle exists between two nodes in a directed graph is NP-complete (see e.g. [9, 10]), we infer that `rabin` can be implemented to run in exponential time.

Next, we study the complexity of deciding the value of the Rabin index. We can exploit the NP-hardness of simple cycle detection to show that the natural decision problem for the Rabin index, whether $\mathsf{RI}(c)$ is at least k, is NP-hard for fixed $k \geq 2$. In contrast, for $k = 1$, we show that this problem is in P.

Theorem 2 *Deciding whether the Rabin index of a colored arena (V, E, c) is at least k is NP-hard for every fixed $k \geq 2$, and is in P for $k = 1$.*

Proof. First consider the case when $k \geq 2$. We use the fact that deciding whether there is a simple cycle through nodes $s \neq t$ in a directed graph (V, E) is NP-complete (see e.g. [10]). Without loss of generality, for all v in V there is some w in V with (v, w) in E (we can add (v, v) to E otherwise). Our hardness reduction uses a colored arena (V', E', c), depicted in Figure 6, which we now describe:

We color s with $k - 1$ and t with k, and color all remaining nodes of V with 0. Then we add $k+1$ many new nodes (shown in blue/top in the figure) to that graph that form a "spine" of descending colors from k down to 0, connected by simple cycles. Crucially, we also add a simple cycle between t and that new k node, and between s and the new $k - 2$ node.

We claim that the Rabin index of (V', E', c) is at least k iff there is a simple cycle through s and t in the original directed graph (V, E).

1. Let there be a simple cycle through s and t in (V, E). Since there is a simple cycle between s and the new $k-2$ node, `cycle` does not change the color at s. As there is a simple cycle through s and t, method `cycle` also does not change the color at t. Clearly, no colors on the spine can be changed by `cycle`. Since there is a simple cycle between t and the new k node, method `pop` also does not change colors. But then the Rabin index of c is k and so at least k.

2. Conversely, assume that there is no simple cycle through s and t in the original graph (V, E). It follows that the anchor j of t has value 0 or, if k is even, has value -1. In this case, `cycle` changes the color at t to the parity of k. Then, `pop` reduces the color of the remaining node colored k to $k-1$. Thus, it cannot be the case that the Rabin index of c is at least k.

This therefore proves the claim. Second, consider the case when $k = 1$. Deciding whether $\mathsf{RI}(c)$ is at least 1 amounts to checking whether $c \equiv \vec{0}$ where $\vec{0}(v) = 0$ for all v in V. This is the case iff all simple cycles in (V, E, c) have even c-parity. But that is the case iff all cycles in (V, E, c) have even c-parity.

To see this, note that the "if" part is true as simple cycles are cycles. As for the "only if" part, this is true since if there were a cycle C with odd c-parity, then some node v on that cycle would have to have that minimal c-color, but v would then be on some simple cycle whose edges all belong to C.

Finally, checking whether all cycles in (V, E, c) have even c-parity is in P. $\square$

The decision problem of whether $\mathsf{RI}(c) = 1$ cannot be in NP, unless NP equals coNP. Otherwise, the decision problem of whether $\mathsf{RI}(c) \leq 1$ would also be in NP, since we can decide in P whether $\mathsf{RI}(c) = 0$ and since NP is closed under unions. But then the complement decision problem of whether $\mathsf{RI}(c) \geq 2$ would be in coNP, and we have shown it to be NP-hard already. Therefore, all problems in NP would reduce to this problem and so be in coNP as well, a contradiction.

In the next section we show how to bypass this complexity by using an abstract version of the equivalence relation.

6 Abstract Rabin index

We now discuss an efficient version of `rabin` which replaces oracle calls for simple cycle detection with over-approximating cycle detection. In fact, this static analysis computes an abstract Rabin index, whose definition is based on an abstract version of the equivalence relation $\equiv$. We define these notions formally.

Definition 4 *1. Let* $\texttt{rabin}^\alpha$ *be* `rabin` *where all existential quantifications over simple cycles are replaced with existential quantifications over cycles.*

2. *Let (V, E) be a directed graph and $c, c' \colon V \to \mathbb{N}$ two coloring functions. Then:*
   - (a) *$c \equiv^\alpha c'$ iff for all cycles C, the parities of their c- and c'-colors are equal.*
   - (b) *The abstract Rabin index $\mathsf{RI}^\alpha(c)$ of (V, E, c) is $\min\{\mu(c') \mid c \equiv^\alpha c'\}$.*

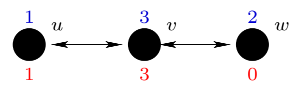


Figure 7: Coloring functions c (blue/top) and c' (red/bottom) with $c \equiv c'$ but $c \not\equiv^\alpha c'$

Thus $\texttt{rabin}^\alpha$ uses the set of cycles in (V, E) to overapproximate the set of simple cycles in (V, E). In particular, $c \equiv^\alpha c'$ implies $c \equiv c'$ but not the other way around, as can be seen in the example in Figure 7.

In that example, we have $c \equiv c'$ since all simple cycles have the same parity of color with respect to c and c'. But there is a cycle that reaches all three nodes and which has odd color for c and even color for c'. Thus, $c \not\equiv^\alpha c'$ follows.

We now show that the overapproximation $\texttt{rabin}^\alpha$ of `rabin` is sound in that its output coloring function is equivalent to its input coloring function. Below, in Theorem 3, we further show that this output yields an abstract Rabin index.

Lemma 3 *Let (V, E, c) be a colored arena and let $\texttt{rabin}^\alpha(V, E, c)$ return c'. Then $c \equiv^\alpha c'$ and $\mu(c') \geq \mathsf{RI}(c)$.*

Proof. Let $c = c_0, c_1, \ldots$ be the sequence of coloring functions that reflect the state changes of c in the call $\texttt{rabin}^\alpha(V, E, c)$. Since $\equiv^\alpha$ is transitive, it suffices to show that $c_n \equiv^\alpha c_{n+1}$ for all such n. So let c_n be given.

1. Consider first the case when c_{n+1} is obtained from c_n by an execution of the for-statement in `pop`. Then m is the maximal color of c_n but there is no cycle in (V, E) that has c_n-color m. In other words, color m will never decide the c_n-color of a cycle. It is therefore safe to decrease all occurrences of m to $m - 1$, as this will change the color of no cycle in (V, E). Since this change defines c_{n+1}, we have $c_i \equiv^\alpha c_{n+1}$ as desired.

2. Now consider the case when c_{n+1} is the result of c_n through the execution of the if-branch in `cycle`. Then we consider a node v_i for which `getAnchor` returns -1. Therefore, there is no cycle C through v_i in (V, E) whose c_n-color is lower than $c_n(v_i)$ and has different parity than $c_n(v_i)$. But the color of cycles through v_i can be at most $c_n(v_i)$. Therefore, all cycles through v_i have the same parity as $c_n(v_i)$. It is therefore safe to reduce the color at v_i to that parity, as done in `cycle`. For the resulting c_{n+1} we therefore have $c_n \equiv^\alpha c_{n+1}$.

3. Now consider the case when c_{n+1} is the result of c_n through the execution of the else-branch in `cycle`. If the call to `getAnchor` returns $j \geq 0$ for node v_i, then consider an arbitrary cycle C in (V, E) through v_i whose color p has a parity other than that of $c_n(v_i)$. Then it must be that $j \leq p$ by the definition of method `getAnchor`. So every cycle through v_i has either a color that has the parity of $c_n(v_i)$ or has a color p with $j \leq p$. Therefore, it is safe to change the color at v_i to $j + 1$ (the case $j + 1 = c_n(v_i)$ will have no effect), resulting in new coloring function c_{n+1}: this is so since then all cycles through v_i have the same parity with respect to c_n and c_{n+1}. (And both coloring functions could only break $c_n \equiv^\alpha c_{n+1}$ by means of cycles through v_i.) $\square$

Note that the definition of $\equiv^\alpha$ is like the characterization of $\equiv$ in Proposition 1, except that the universal quantification over simple cycles is being replaced by a universal quantification over cycles for $\equiv^\alpha$. In proving Lemma 3, we were thus able to reuse the proof for Lemma 1 where we replace $\equiv$ with $\equiv^\alpha$, `rabin` with $\texttt{rabin}^\alpha$, and "simple cycle" with "cycle" throughout in that proof.

We can now adapt the results for `rabin` to this abstract setting.

Lemma 4 *Let (V, E, c) be a colored arena where*

1. *there is a cycle in (V, E) whose color is the maximal one of c*
2. *for all v in V with $c(v) > 1$, node v is on a cycle C with color $c(v) - 1$.*

Then there is no c' with $c \equiv^\alpha c'$ and $\mu(c') < \mu(c)$, and so $\mu(c) = \mathsf{RI}^\alpha(c)$.

Proof. Let k be the maximal color of c and consider an arbitrary c' with $c \equiv^\alpha c'$.

Proof by contradiction: Let the maximal color k' of c' satisfies $k' < k$. By the first assumption, there is a cycle C_0 whose c-color is k. Since $k' < k$ and $c \equiv^\alpha c'$, we know that the c'-color of C_0 can be at most $k - 2$. Let v_0 be a node on C_0 such that $c'(v_0)$ is the c'-color of C_0. Then $c'(v_0) \leq k - 2$. As all nodes on C_0 have c-color k, we have also $c(v_0) \geq k$. Again, if $k < 2$ we have a contradiction right away. So let $k \geq 2$.

By the second assumption, there is some cycle C_1 through v_0 such that the color of C_1 is $k - 1$. In particular, there is some node v'_0 in C_1 with color $k - 1$. But $k - 1$ cannot be the color of C_1 with respect to c' since v_0 is on C_1 and $c'(v_0) \leq k - 2$. Since $c \equiv^\alpha c'$, the c'-color of C_1 is therefore at most $k - 3$. So there is some v_1 on C_1 such that $c'(v_1) \leq k - 3 < k - 1 \leq c(v_1)$.

If $c(v_1) > 1$, we repeat this argument at node v_1 to construct a cycle C_2 through v_1 with color $c(v_1) - 1$. Again, there then have to be nodes v'_1 and v_2 on C_2 such that the color $c'(v'_1)$ is the c'-color of C_2, and such that $c'(v_2) \leq k - 4 < k - 2 \leq c(v_2)$ holds.

In this manner, we can repeat this argument to construct cycles $C_0, C_1, C_2, \ldots$ and nodes $v_0, v'_0, v_1, v'_1, v_2, v'_2, \ldots$ such that $c'(v_j) \leq k - j - 2 < k - j \leq c(v_j)$ until $k - j \leq c(v_j) \leq 1$. But then we obtain $c'(v_j) \leq k - j - 2 \leq 1 - 2 = -1$, a contradiction. $\square$

Similary to the case for algorithm `rabin`, we now show that the output of $\texttt{rabin}^\alpha$ satisfies the assumptions of Lemma 4. Since algorithm $\texttt{rabin}^\alpha$ is sound for $\equiv^\alpha$, we therefore infer that it computes coloring functions whose maximal color equals the abstract Rabin index of their input coloring function.

Theorem 3 *Let (V, E, c) be a colored arena. And let c^* be the output of the call $\texttt{rabin}^\alpha(V, E, c)$. Then $c \equiv^\alpha c^*$ and $\mu(c^*)$ is the abstract Rabin index $\mathsf{RI}^\alpha(c)$.*

Proof. By Lemma 3, we have $c \equiv^\alpha c^*$. Since $\equiv^\alpha$ is transitive, it suffices to show that there is no c' with $c^* \equiv^\alpha c'$ and $\mu(c') < \mu(c^*)$. By Lemma 4, it therefore suffices to establish the two assumptions of that lemma for c^*. What we do know is that neither `cycle` nor `pop` have an effect on c^* as it was returned by $\mathtt{rabin}^\alpha$.

The first assumption is therefore true since `pop` has no effect on c^* and so there must be a cycle in (V, E) whose color is the maximal one in c. (This also applies to the boundary case when c^* has only one color, as (V, E) has to contain cycles since it is finite and all nodes have outgoing edges.)

As for the second assumption, let by way of contradiction be some node v with $c^*(v) > 1$ and no cycle through v with color $c^*(v) - 1$. Then `cycle` would have an effect on $c^*(v)$ and would lower it, a contradiction. □

We now study the sets of parity games whose abstract Rabin index is below a fixed bound. We define these sets formally.

Definition 5 *Let $\mathcal{P}_k^\alpha$ be the set of parity games (V, V_0, V_1, E, c) with $\mathsf{RI}^\alpha(c) < k$.*

We can now show that parity games in these sets are efficiently solvable, also in the sense that membership in such a set is efficiently decidable.

Theorem 4 *Let $k \geq 1$ be fixed. All parity games in $\mathcal{P}_k^\alpha$ can be solved in polynomial time. Moreover, membership in $\mathcal{P}_k^\alpha$ can be decided in polynomial time.*

Proof. For each parity game (V, V_0, V_1, E, c) in $\mathcal{P}_k^\alpha$, we first run $\mathtt{rabin}^\alpha$ on it, which runs in polynomial time. By definition of $\mathcal{P}_k^\alpha$, the output coloring function c^* has index $< k$. Then we solve the parity game (V, V_0, V_1, E, c^*), which we can do in polynomial time as the index is bounded by k. But that solution is also one for (V, V_0, V_1, E, c) since $c \equiv^\alpha c^*$ by Lemma 3, and so $c \equiv c^*$ as well.

That the membership test is polynomial in the running time can be seen as follows: for coloring function c, compute $c' = \mathtt{rabin}^\alpha(V, E, c)$ and return `true` if $\mu(c') < k$ and return `false` otherwise; this is correct by Theorem 3. □

We note that algorithm $\mathtt{rabin}^\alpha$ is precise for colored arenas $A = (V, E, c)$ with Rabin index 0. These are colored arenas that have only simple cycles with even color. Since a colored arena has a cycle with odd color iff it has a simple cycle with odd color, $\mathtt{rabin}^\alpha$ will correctly reduce all colors to 0 for such arenas.

For Rabin index 1, the situation is more subtle. We cannot expect $\mathtt{rabin}^\alpha$ to always be precise, as the decision problem for $\mathsf{RI}(c) \geq 2$ is NP-hard. Algorithm $\mathtt{rabin}^\alpha$ will correctly compute Rabin index 1 for all those arenas that do not have a simple cycle with even color. But for c from Figure 7, e.g., algorithm $\mathtt{rabin}^\alpha$ does not change c with index 3, although the Rabin index of c is 1.

Game Type	$\mu(c)$	$\mu(s(c))$	$\mathsf{RI}^\alpha(c)$	S	R	#I	Sol	Sol.S	Sol.R
`Ladder`[19]	1	1	1	0.06	0.23	0	1.04	1.03	1.05
`Jurdziński`[5 10]	11	11	2	0.07	44.27	1	82.48	81.89	14.14
`Recursive Ladder`[6]	20	18	6	0.03	0.52	1	26.67	23.37	3.42
`Strategy Impr`[7]	210	158	3	0.06	245.39	1	69.36	68.31	15.86
`Model Checker Ladder`[50]	100	100	0	0.06	42.56	1	1.17	1.13	0.45
`Tower of Hanoi`[3]	1	1	1	0.05	0.37	0	2.46	2.42	2.40

Figure 8: Indices and average times (in ms) for 100 runs for game types named in first column. Next three columns: original, statically compressed, and $\texttt{rabin}^\alpha$-compressed index. Next three columns: times of static and $\texttt{rabin}^\alpha$-compression, and the number of iterations within $\texttt{rabin}^\alpha$. Last three columns: Times of solving the original, statically compressed, and $\texttt{rabin}^\alpha$-compressed games with Zielonka's solver

7 Experimental results

We now provide experimental results for our algorithm $\texttt{rabin}^\alpha$.

The objectives of our experiments are to compare the effectiveness of color compression of $\texttt{rabin}^\alpha$ to a known color compression algorithm (called static compression), to observe the performance improvement in solving compressed games using Zielonka's parity game solver [1], and to get a feel for how much the abstract Rabin index reduces the index of random and non-random games.

The applications used in the experiments are written in Scala. Since we are interested in descriptive complexity measures and relative computation time, all game elements are realized as objects for sake of conceptual simplicity.

We programed algorithm `rabin` with simple cycle detection reduced to incremental SAT solving. This did not scale to graphs with more than 40 nodes. But for those games for which we could compute the Rabin index, $\texttt{rabin}^\alpha(V, E, c)$ often computed the Rabin index $\mathsf{RI}(c)$ or did get very close to it.

Our implementation of $\texttt{rabin}^\alpha$ reduced cycle detection to the decomposition of the graph into strongly connected components, using Tarjan's algorithm (which is linear in the number of edges). The rank function is only needed for complexity and termination analysis, we replaced it with Booleans that flag whether `cycle` or `pop` had an effect.

The standard static compression algorithm simply removes gaps between colors, e.g. a set of colors $\{0, 3, 4, 5, 6, 8\}$ is being compressed to $\{0, 1, 2, 3, 4\}$. Below, we write $s(c)$ for the statically compressed version of coloring function c.

The experiments are conducted on non-random and random games separately. Each run of the experiments generates a parity game $G = (V, V_0, V_1, E, c)$ of a selected configuration. Static compression and $\texttt{rabin}^\alpha$ are performed on these games. We report the time taken to execute static compression and $\texttt{rabin}^\alpha$, as well as the number of iterations that $\texttt{rabin}^\alpha$ runs until `cycle` and `pop` have no effect, i.e. the number of iterations needed for $\mu(c)$ to reach $\mathsf{RI}^\alpha(c)$.

Game Configs	$\mu(c)$	$\mu(s(c))$	$\mathsf{RI}^\alpha(c)$	S	R	#I	Sol	Sol.S	Sol.R
100/1/20/100	98.40	46.24	37.49	0.12	43.05	1	7.46	5.82	5.94
200/1/40/200	198.27	92.13	82.29	0.07	537.46	1	21.87	19.72	19.79
400/1/80/400	398.51	183.97	171.86	0.15	8443.94	1	100.45	89.48	88.83
800/1/160/400	398.85	304.07	294.95	0.35	102651.25	1	645.43	621.22	623.34
1000/1/200/400	398.88	337.82	330.66	0.45	223560.69	1	1094.31	1062.86	1067.23

Figure 9: Indices and average times (in ms) for 100 runs of random games of various configurations listed in the first column. Next three columns: average original, statically compressed, and `rabin`$^\alpha$-compressed indices. The remaining columns are as in Figure 8

Finally, we record the wall-clock time required to solve original, statically compressed, and `rabin`$^\alpha$-compressed games, using Zielonka's solver [1].

We use PGSolver to generate non-random games, detailed descriptions on these games can be found in [11]. Each row in Figure 8 shows the average statistics from 100 runs of the experiments on corresponding non-random game. We see that `rabin`$^\alpha$ has significantly reduced the indices of `Jurdziński`, `Recursive Ladder`, `Strategy Impr`, and `Model Checker Ladder`, where $\mathsf{RI}^\alpha(c)$ is 0% to 33% of the index $\mu(s(c))$ of the statically compressed coloring function.

The benefit of `rabin`$^\alpha$ is reflected by the performance increase in solving parity games. For all four game types, we observe 60% to 85% in solver time reduction between solving statically compressed and `rabin`$^\alpha$-compressed games.

The time required to perform static compression is low compared to the time needed for `rabin`$^\alpha$-compression. But `rabin`$^\alpha$-compression followed by solving the game is still faster than solving the original game for the game types `Jurdziński`, and `Recursive Ladder`.

For the game types `Ladder`, and `Tower of Hanoi` further color compression is not possible due to their extremely small indices. Method `cycle` has no effect on `Clique` games, but `pop` manages to reduce its index by 1.

We now discuss our experimental results on random games. The notation used to describe randomly generated parity games is $xx/yy/zz/cc$, where xx is the number of nodes (node ownership is determined by a fair coin flip for each node independently), with between yy to zz out-going edges for each node, and with colors at nodes chosen at random from $\{0, \ldots, cc\}$. Also, the games used in the experiments have 1 as the minimum number of out-going edges. This means that the nodes have no dead-ends. We also disallow self-loops (no (v, v) in E).

Figure 9 shows the average statistics of 100 runs of experiments on five selected game configurations. (Our experiments on larger games are consistent with the data reported here, and so not reported here.) The results indicate that static compression is effective in reducing the colors for randomly generated games, it achieves 24% to 54% index reduction. The `rabin`$^\alpha$-compression achieves further 2% to 19% reduction. Due to the relatively small index reduction by $\texttt{rabin}^\alpha$, we do not see much improvement in solving $\texttt{rabin}^\alpha$-compressed games over solving statically-compressed one. In addition, $\texttt{rabin}^\alpha$ reduces $\mu(c)$ to $\mathsf{RI}^\alpha(c)$ in one iteration for all of the randomly generated games G.

On all random games generated only the first execution of `pop` had an effect. Effect in further iterations of `pop` requires specific game structure, e.g. as for the game in Figure 3. We would not expect to see such structure on random games.

The experimental results show that $\texttt{rabin}^\alpha$ is able to reduce the indices of parity games significantly and quickly, for certain structures such as `Jurdziński`, and `Recursive Ladder`. Hence it effectively improves the overall solver performance for those games.

However, algorithm $\texttt{rabin}^\alpha$ has a negative effect on the overall performance for other non-random games and experimented random games, when we consider $\texttt{rabin}^\alpha$-compression time plus solver time.

8 Conclusions

We now summarize the results and insights of this paper, and mention what questions we mean to pursue in future work.

We have provided a descriptive measure of complexity for parity games that (essentially) measures the number of colors needed in a parity game if we forget the ownership structure of the game but if we do not compromise the winning regions or winning strategies by changing its colors.

We called this measure the Rabin index of a parity game. We then studied this concept in depth. By analyzing the structure of simple cycles in parity games, we arrived at an algorithm that computes this Rabin index in exponential time.

Then we studied the complexity of the decision problem of whether the Rabin index of a parity game is at least k for some fixed $k > 0$. For k equal to 1, we saw that this problem is in P, but we showed NP-hardness of this decision problem for all other values of k. These lower bounds therefore also apply to games that capture these decision problems in game-theoretic terms.

Next, we asked what happens if our algorithm `rabin` abstractly interprets all detection checks for simple cycles through detection checks for cycles. The resulting algorithm $\texttt{rabin}^\alpha$ was then shown to run in polynomial time, and to compute a corresponding abstract and sound interpretation of the Rabin index.

Our experiments were performed on random and non-random games. We observed that $\texttt{rabin}^\alpha$-compression plus Zielonka's solver [1] achieved 29% and 85% time reduction for `Jurdziński` and `Recursive Ladder` games, respectively, over solving the original games. But for other game types and random games, no such reduction was observed. We also saw that for some structured game types, the abstract Rabin index is dramatically smaller than the index of the game.

In future work we mean to investigate properties of the measure $\mathsf{RI}^\alpha(c) - \mathsf{RI}(c)$. Intuitively, it measures the difference of the Rabin index based on the structure of cycles with that based on the structure of simple cycles. From the family of examples in Figure 5 we already know that this measure can be arbitrarily large.

It will also be of interest to study variants of $\mathsf{RI}(c)$ that are targeted for specific solvers. For example, the SPM solver in [2] favors fewer occurrences of odd colors but also favors lower index. This suggests a measure with a lexicographical order of the Rabin index followed by an occurrence count of odd colors.

References

[1] Zielonka, W.: Infinite games on finitely coloured graphs with applications to automata on infinite trees. Theoretical Computer Science **200**(1–2):135–183 (1998).

[2] Jurdziński, M.: Small progress measures for solving parity games. In Proc. of STACS, LNCS 1770, 290–301. Springer-Verlag (2000).

[3] Vöge, J., Jurdziński, M.: A discrete strategy improvement algorithm for solving parity games. In Proc. of Computer Aided Verification, LNCS 1855, 202–215. Springer-Verlag (2000).

[4] Berwanger, D., Dawar, A., Hunter, P., Kreutzer, S.: DAG-width and parity games. In Proc. of STACS, LNCS 3884, 524–436. Springer-Verlag (2006).

[5] Friedmann, O., Lange, M.: Solving parity games in practice. In Proc. of ATVA, LNCS 5799, 182–196. Springer-Verlag (2009).

[6] Wagner, K.: On ω-regular sets. Information and Computation **43**:123–177 (1979).

[7] Carton, O., Maceiras, R.: Computing the Rabin Index of a Parity Automaton. Informatique Théorique et Applications **33**(6):495–506 (1999).

[8] Emerson, E., Jutla, C.: Tree automata, μ-calculus and determinacy. In Proc. of FOCS, 368–377. IEEE (1991).

[9] Even, S., Itai, A., Shamir, A.: On the complexity of timetable and multicommodity flow problems. SIAM J. Comput. **5**(4):691–703 (1976).

[10] Fortune, S., Hofcroft, J., Wyllie, J.: The directed subgraph homeomorphism problem. Theoretical Computer Science **10**:111–121 (1980).

[11] Friedmann, O., Lange, M.: The PGSolver Collection of Parity Game Solvers. Technical report, Institut für Informatik, LMU Munich (Feb 2010).