Simple and tight complexity lower bounds for solving Rabin games*

Antonio Casares[1], Marcin Pilipczuk[2], Michał Pilipczuk[2], Uéverton S. Souza[3], and K. S. Thejaswini[4,5]

[1] *LaBRI, Université de Bordeaux, France*
[2] *University of Warsaw, Poland*
[3] *Universidade Federal Fluminense, Niterói, Brazil*
[4] *University of Warwick, United Kingdom*
[5] *Institute of Science and Technology, Austria*

Abstract

We give a simple proof that assuming the Exponential Time Hypothesis (ETH), determining the winner of a Rabin game cannot be done in time $2^{o(k \log k)} \cdot n^{\mathcal{O}(1)}$, where k is the number of pairs of vertex subsets involved in the winning condition and n is the vertex count of the game graph. While this result follows from the lower bounds provided by Calude et al [SIAM J. Comp. 2022], our reduction is considerably simpler and arguably provides more insight into the complexity of the problem. In fact, the analogous lower bounds discussed by Calude et al, for solving Muller games and multidimensional parity games, follow as simple corollaries of our approach. Our reduction also highlights the usefulness of a certain pivot problem — Permutation SAT — which may be of independent interest.

1 Introduction

We study Rabin games defined as follows. The arena of a Rabin game is a (finite) directed graph D whose vertices are divided among the two players involved: Steven and Audrey[1]. There is an initial vertex u_1 on which a token is initially placed. The game proceeds in turns. Each turn, the player controlling the vertex u on which the token is currently placed chooses any outneighbour v of u and moves the token from u to v. Thus, by moving the token, the players construct an infinite walk $\rho = (u_1, u_2, u_3, \ldots)$ in D, called a *play*. To determine the winner, the play ρ is compared against the winning condition consisting of k pairs of vertex subsets $(G_1, B_1), (G_2, B_2), \ldots, (G_k, B_k)$ as follows: Steven wins if there exists $i \in \{1, \ldots, k\}$ such that along ρ, G_i is visited infinitely often while B_i is visited only a finite number of times; Audrey wins otherwise. The computational question associated with the game is to determine which player has a winning strategy.

Rabin conditions were first introduced by Rabin in his proof of decidability of S2S (monadic second order with two successors) [Rab69]. They also naturally appear in the determinization of Büchi automata [Saf88, Pit06, Sch09], a key step in the synthesis problem for reactive systems with specifications given in Linear

*This work is a part of projects CUTACOMBS (Ma. Pilipczuk), BOBR (Mi. Pilipczuk), and VAMOS (K. S. Thejaswini) that have received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreements No 714704, 948057, and 101020093, respectively. Ma. Pilipczuk is also partially supported by Polish National Science Centre SONATA BIS-12 grant number 2022/46/E/ST6/00143.

[1] The right way to memorize the player names is St**even** and **Odd**rey; the naming comes from the context of parity games.

Temporal Logic. Since then, algorithms for solving Rabin games have been extensively studied [EJ99, KV98, PP06, BMM+22]. They generalise the more well-known *parity games*, which differ by altering the winning condition as follows. Each vertex of the graph bears a *colour*, which is an integer from $\{1, \ldots, k\}$. Steven wins a play ρ if the largest colour seen infinitely often in ρ is even, and otherwise Audrey wins. Indeed, to reduce a parity game with colours $\{1, \ldots, k\}$ to a Rabin game with $\lfloor k/2 \rfloor$ pairs in the winning condition, it suffices to take the same graph D and set

$$G_i = \{\text{vertices with colours } \geqslant 2i\} \qquad \text{and} \qquad B_i = \{\text{vertices with colours } \geqslant 2i+1\},$$

for all $i \in \{1, \ldots, \lfloor k/2 \rfloor\}$. Further, both parity games and Rabin games are generalised by *Muller games*, where again vertices have colours from $\{1, \ldots, k\}$ (each vertex may bear multiple colours), and the winning condition is defined by simply providing a family $\mathcal{F}$ of subsets of $\{1, \ldots, k\}$ that are winning for Steven in the following sense: Steven wins a play ρ if the set of colours seen infinitely often in ρ belongs to $\mathcal{F}$.

In a breakthrough paper, Calude, Jain, Khoussainov, Li, and Stephan [CJK+22] proved that solving all the three games discussed above is fixed-parameter tractable when parameterised by k (the number of colours, respectively the number of pairs in the winning condition). More precisely, determining the winner of the game can be done in $k^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ time, where n is the number of vertices of the arena. The recent work of Majumdark, Sağlam and Thejaswini [MST23] provides a more precise analysis which results in an algorithm solving Rabin games in polynomial space and time $k!^{1+o(1)} \cdot nm$, where m is the number of edges. While the work of Calude et al. also provided a quasipolynomial-time algorithm to solve parity games, it is known that solving Rabin games is already NP-complete [EJ88, EJ99], while solving Muller games is PSPACE-complete [HD05]. Hence, for those games, the existence of (quasi)polynomial-time algorithms is unlikely.

In their work, Calude et al. [CJK+22] provided also complexity lower bounds based on the Exponential Time Hypothesis (ETH, the assumption that there exists $\delta > 0$ such that 3SAT problem cannot be solved in time $\mathcal{O}(2^{\delta n})$) for some of the games discussed above. They proved that assuming ETH, there are no algorithms with running time $2^{o(k \log k)} \cdot n^{\mathcal{O}(1)}$ for solving Muller games with priorities in $\{1, \ldots, k\}$ or d-dimensional k-parity games (see preliminaries for a definition of the latter variant). Since every k-dimensional parity game can be reduced in polynomial time to a Rabin game with k pairs in the winning condition (see [CHP07]), one can also derive, as a corollary, the same lower bound for solving Rabin games. The reduction provided by Calude et al. starts with the Dominating Set problem and is rather involved.

Our contribution. We provide a simple reduction that reproves the tight complexity lower bound for solving Rabin games that follows from the work of Calude et al. More precisely, we prove that assuming ETH, there is no algorithm for this problem with running time $2^{o(k \log k)} \cdot n^{\mathcal{O}(1)}$. The same lower bound for (the more general) Muller games follows as a direct corollary. By a minor twist of our construction, we can also reprove the lower bound for k-dimensional parity games reported by Calude et al.

We believe that our reduction is significantly simpler and more transparent than that of Calude et al. but more importantly, it gives a better insight into the origin of the $2^{o(k \log k)}$ factor in the complexity of the problem. Analyzing the algorithms of [CJK+22, MST23], this factor stems from considering all possible permutations of the k pairs of vertex subsets involved in the winning condition. In our reduction, those permutations form the space of potential solutions of a carefully chosen pivot problem — Permutation SAT, a special case of a temporal constraint satisfaction problem — which we discuss below.

Temporal CSPs and Permutation SAT. A constraint satisfaction problem (CSP) is the problem of deciding if there exists a variable assignment that satisfies a given set of constraints. *Temporal problems*

is a rich family of CSPs that model planning various events on a timeline. In a basic form, every variable corresponds to an event that needs to be scheduled at some point of time and constraints speak about some events being in specific order (e.g., one preceding another), at the same time, or at different times. This is usually modeled with $\mathbb{Q}$ as the domain and constraints having access to predicates $<$, $\leqslant$, $=$, and $\neq$. A P vs NP dichotomy for finite languages within this formalism has been provided by Bodirsky and Kára [BK10].

An instance of such a temporal CSP with k variables and n constraints can be solved in time $k^k \cdot (k+n)^{\mathcal{O}(1)}$ as follows: since variables are accessed only via comparisons $<$, $\leqslant$, $=$, and $\neq$, without loss of generality one can restrict to assignments with values in $\{1, 2, \ldots, k\}$, and there are k^k such assignments that can be all checked. An interesting and challenging question is: For which languages this running time can be significantly improved?

In this paper, we focus in a particular temporal CSP: Permutation SAT. An instance of this problem is given by a boolean combination of literals of the form $x_1 < x_2 < \ldots < x_\alpha$; a solution for it is an assignment of variables to integers making it a valid formula. We say that such a problem is an instance of (α, β)-Permutation SAT if its constraints use at most β literals, and each of these literals involves at most α variables. Observe that, without loss of generality, in Permutation SAT one can restrict attention to assignments being surjective functions from variables $\{x_1, \ldots, x_k\}$ to $\{1, \ldots, k\}$, which can be interpreted as permutations of $\{1, \ldots, k\}$; this justifies the choice of the problem name and yields a brute-force algorithm with running time $k! \cdot (k+n)^{\mathcal{O}(1)}$.

Bonamy et al. [BKN+18] proved that $(3, \infty)$-Permutation SAT admits no $2^{o(k \log k)} n^{\mathcal{O}(1)}$ algorithm unless the Exponential Time Hypothesis (ETH) fails. Our main technical contribution is a similar lower bound for $(2, 4)$-Permutation SAT (Theorem 3.1). The proof of this result is a simple reduction from the $k \times k$-Clique problem considered by Lokshtanov, Marx, and Saurabh [LMS18]. It is our belief that (α, β)-Permutation SAT is a problem with a very easy and robust formulation, hence its usefulness may extend beyond the application to Rabin games discussed in this work.

2 Preliminaries on games

For a positive integer p, we denote $[p] \coloneqq \{1, \ldots, p\}$.

Rabin and Muller games are turn-based two-player games played on an *arena* that is a directed graph $D = (V, E)$ together with a partition of the vertices into those owned by player Steven and those owned by player Audrey. A token is initially placed on a designated starting vertex u_1. In each consecutive turn, the owner of the vertex bearing the token moves the token along an edge of D. Thus, the players jointly form an infinite sequence of vertices in consecutive turns, referred to as a *play*. An *objective* is a representation of a subset of the set of all possible plays. We will consider three different objectives discussed below.

Muller objectives. In a Muller game, each vertex is labelled with a subset of colours from $[k]$ via a mapping $c\colon V \to 2^{[k]}$, where V is the set of vertices of the arena D. The Muller objective is specified by a family of subsets of colours $\mathcal{F} \subseteq 2^{[k]}$. A play ρ is winning for Steven if the set of colours visited infinitely often, belongs to $\mathcal{F}$, that is, if

$$\bigcup_{v \in \mathrm{Inf}(\rho)} c(v) \in \mathcal{F},$$

where $\mathrm{Inf}(\rho)$ is the set of vertices appearing infinitely often in the play.

Rabin objective. A Rabin objective of *degree* k consists of k pairs of vertex subsets $(G_1, B_1), \ldots, (G_k, B_k)$; G_i is said to be the *good* subset for index i, and B_i is the *bad* subset. A play ρ is winning for Steven if there exists an index $i \in \{1, \ldots, k\}$ such that ρ visits G_i infinitely often and B_i only a finite number of times.

Rabin objectives of degree k can be encoded as a Muller objective using $2k$ colours. Indeed, for each $0 \leqslant i < k$, we associate $2i$ with the subset G_i and $2i+1$ with the subset B_i. We define $c\colon V \to 2^{[2k]}$ and $\mathcal{F}$ as:

$$c(v) = \{2i\colon v \in G_i\} \cup \{2i+1\colon v \in B_i\} \quad \text{and} \quad \mathcal{F} = \{C \subseteq [2k]\colon \exists i\ 2i \in C \text{ and } 2i+1 \notin C\}.$$

Generalised parity objective. Generalised parity games were first considered in the work of Chaterjee, Henzinger, and Piterman [CHP07]. In a d-dimensional k-parity condition, each vertex is labelled with a d-dimensional vector of integers from $\{1, \ldots, k\}$. An infinite play satisfies this objective for Steven if and only if there is some coordinate such that the highest number that occurs infinitely often at this coordinate is even. Audrey wins otherwise.

These games are inter-reducible with Rabin games, as shown by [CHP07]. For one direction, since a d-dimensional k-parity objective is a disjunction of d distinct parity objectives, and each parity objective can be expressed as a Rabin objective of degree $\lceil k/2 \rceil$, the d-dimensional k-parity objective can therefore similarly be transformed into a Rabin objective of degree $d\lceil k/2 \rceil$, with $\lceil k/2 \rceil$ Rabin pairs for *each* of the d parity objectives. Conversely, a Rabin objective with d pairs can be represented as a d-dimensional 3-parity objective. Indeed, we use each pair (G_i, B_i) to define the component p_i that assigns colour 3 to v when $v \in B_i$, colour 2 if $v \in G_i \setminus B_i$ and 1 otherwise.

Calude et al. [CJK+22] showed that generalised parity games cannot be solved in time $2^{o(k \log k)} \cdot n^{\mathcal{O}(1)}$ assuming the ETH, even when the dimensions is $d = 2$.

Strategies and winners. For a given game, with any of the objectives discussed above, a *strategy* of Steven is a function from the set of plays ending at a Steven vertex to the set of vertices. A play $v_0, v_1, \ldots, v_i, \ldots$ is said to respect this strategy if for every vertex v_i which belongs to Steven, the vertex v_{i+1} is the one proposed by the strategy on the finite prefix of this play ending at v_i. For a fixed objective, a game is said to be winning for Steven if he has a strategy such that plays respecting this strategy satisfy the objective.

Positional strategies. We say that a strategy (for Steven) is *positional* (or *memoryless*) if it can be represented by a function assigning an outgoing edge to each vertex owned by Steven. That is, a positional strategy always makes the same decision over the same vertex, and this decision depends only on the current vertex and not on the history of the play. It is well known that Rabin games are positional for Steven in the following sense.

Lemma 2.1 ([EJ88, EJ99]). *Rabin games are positional for Steven. That is, if Steven wins a Rabin game, then he has a positional winning strategy.*

Exponential Time Hypothesis. The Exponential Time Hypothesis is a complexity assumption introduced by Impagliazzo, Paturi and Zane [IPZ01] that postulates the following: there exists $\delta > 0$ such that the 3-SAT problem cannot be solved in time $\mathcal{O}(2^{\delta n})$, where n is the number of variables of the input formula. We refer the reader to [CFK+15, Chapter 14] for an introduction to the applications of ETH for lower bounds within parameterized complexity.

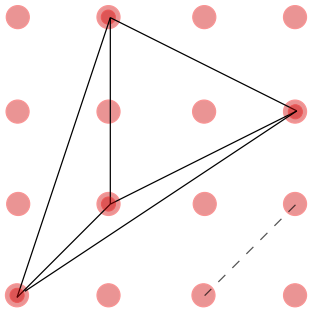

Figure 1: The construction in Section 3. The highlighted clique corresponds to permutation $x_1 < y_4 < x_2 < y_1 < y_3 < x_3 < x_4 < y_2 < x_5$ (with y_1 and y_3 possibly swapped). The dashed non-edge $((4,3),(3,4))$ is disallowed by the clause $\neg\left((x_4 < y_3 < x_5) \wedge (x_3 < y_4 < x_4)\right)$ which ensures if y_4 appears between x_3 and x_4, then y_3 does not appear between x_4 and x_5.

3 Permutation SAT

Fix integers $\alpha \geqslant 2$ and $\beta \geqslant 1$ and let X be a finite set of variables. An *α-literal* is a predicate of the form $x_1 < x_2 < \ldots < x_{\alpha'}$ (being a shorthand for $(x_1 < x_2) \wedge (x_2 < x_3) \wedge \ldots \wedge (x_{\alpha'-1} < x_{\alpha'})$) for some $2 \leqslant \alpha' \leqslant \alpha$ and variables $x_1, x_2, \ldots, x_{\alpha'}$ belonging to X; a *literal* is a 2-literal (i.e., a predicate of the form $x_1 < x_2$). An *(α,β)-clause* is a disjunction of at most β α-literals, and an *(α,β)-formula* is a conjunction of (α,β)-clauses. By β-clauses and β-formulas we mean $(2,\beta)$-clauses and $(2,\beta)$-formulas, respectively.

If ϕ is a formula with variable set X, then for a permutation π of X we define the satisfaction of (literals and clauses of) ϕ by π in the obvious manner. In the (α,β)-Permutation SAT problem we are given an (α,β)-formula ϕ and the task is to decide whether there exists a permutation of the variables of ϕ that satisfies ϕ. β-Permutation SAT is a shorthand for $(2,\beta)$-Permutation SAT.

In this section we prove the following hardness result.

Theorem 3.1. *Assuming ETH, there is no algorithm for 4-Permutation SAT that would work in time $2^{o(k \log k)} \cdot n^{\mathcal{O}(1)}$, where k is the number of variables and n is the number of clauses.*

To prove Theorem 3.1 we use the problem $k \times k$-Clique considered by Lokshtanov, Marx, and Saurabh [LMS18]. They showed that, unless ETH fails, this problem cannot be solved in $2^{o(k \log k)}$-time. We first define $k \times k$-Clique below and then reduce $k \times k$-Clique to 4-Permutation SAT.

An instance of the $k \times k$-Clique problem is an undirected graph G with the vertex set $\{1,\ldots,k\} \times \{1,\ldots,k\}$ (which we can represent as a grid). This graph G is a positive instance of $k \times k$-Clique if there is one vertex from each *row* of the grid that forms a k-clique, that is, a k-clique in which no two vertices share the same first component.

Theorem 3.2 ([LMS18, Theorem 2.4])**.** *Assuming ETH, there is no $2^{o(k \log k)}$-time algorithm for $k \times k$-Clique.*

The reduction. We now reduce $k \times k$-Clique to 4-Permutation SAT. Suppose G is an instance of $k \times k$-Clique. We construct a 4-formula ϕ_G over variable set $X := \{x_1,\ldots,x_k,x_{k+1},y_1,\ldots,y_k\}$ as follows.

Recall that the vertices of the graph G are of the form (i,j) for $i,j \in \{1,\ldots,k\}$. We say that vertex

(i, j) is in the i^{th} row and j^{th} column. To construct ϕ_G, we first write the following $3k$ many 1-clauses:

$$\begin{aligned} x_1 < x_2, \quad x_2 < x_3, \quad \ldots, \quad x_k < x_{k+1}, \\ x_1 < y_1, \quad x_1 < y_2, \quad \ldots, \quad x_1 < y_k \\ y_1 < x_{k+1}, \quad y_2 < x_{k+1}, \quad \ldots, \quad y_k < x_{k+1} \end{aligned}$$

The conjunction of these clauses ensures that in any permutation satisfying ϕ_G, the variables $x_1, \ldots, x_{k+1}$ are ordered exactly in this way, while variables $y_1, \ldots, y_k$ are sandwiched between x_1 and x_{k+1}. In other words, the y-variables that are placed between x_j and x_{j+1} indicate the rows that choose their clique vertices from the j^{th} column; and for some j's, this set may be empty as well.

Next, we introduce clauses that restrict the placement of variables $y_1, \ldots, y_k$ within the chain $x_1 < x_2 < \ldots < x_{k+1}$. The intention is the following: placing y_i between x_j and x_{j+1} corresponds to choosing the vertex (i, j) to the clique. Hence, it remains to introduce clauses ensuring that vertices chosen in this way in consecutive rows are pairwise adjacent. To this end, for every pair $(a, b), (c, d)$ of vertices non-adjacent in G, we construct the following 4-clause:

$$(y_a < x_b) \vee (x_{b+1} < y_a) \vee (y_c < x_d) \vee (x_{d+1} < y_c).$$

Note that logically, this 4-clause is equivalent to the following:

$$\neg \left((x_b < y_a < x_{b+1}) \wedge (x_d < y_c < x_{d+1}) \right).$$

Thus, intuitively speaking, the 4-clause forbids simultaneously choosing (a, b) and (c, d) to the clique.

This concludes the construction of the formula ϕ_G. It remains to verify the correctness of the reduction.

Lemma 3.3. *The graph G admits a k-clique with one vertex from each row if and only if ϕ_G is satisfiable.*

Proof. First, suppose G contains a k-clique $K = \{(1, b_1), \ldots, (k, b_k)\}$. Consider any permutation π of X such that

- $x_1 < x_2 < \cdots < x_k < x_{k+1}$, and
- $x_{b_i} < y_i < x_{b_i+1}$, for all $j \in \{1, \ldots, k\}$.

(Note that π is not defined uniquely, the relative placement of y_i and $y_{i'}$ can be arbitrary whenever $b_i = b_{i'}$.) It can be easily seen that K being a clique, implies that all clauses in ϕ_G are satisfied. The 1-clauses are satisfied trivially, while every 4-clause constructed for a non-adjacent $(a, b), (c, d)$ is satisfied because (a, b) and (c, d) cannot simultaneously belong to K.

Suppose now that there is an ordering of X that satisfies ϕ_G. Clearly, it must be the case that $x_1 < x_2 < \cdots < x_k < x_{k+1}$. Further, for every $i \in \{1, \ldots, k\}$ we have $x_1 < y_i < x_{k+1}$ and therefore, there exists j_i such that $x_{j_i} < y_i < x_{j_i+1}$. We let $K \coloneqq \{(i, j_i) \colon i \in \{1, \ldots, k\}\}$; note that K contains one vertex from each row. We claim that K is a clique in G. Indeed, since in ϕ_G there is a clause disallowing that $((x_b < y_a < x_{b+1}) \wedge (x_d < y_c < x_{d+1}))$ whenever there is no edge between (a, b) and (c, d), all vertices of K must be pairwise adjacent. □

This concludes the proof of Theorem 3.1. We remark that establishing the complexity of 2- and 3-Permutation SAT remains an interesting and challenging open problem. Eriksson in his MSc thesis [Eri19] shows that 2-Permutation SAT can be solved in time $((k/2)!)^2 \cdot (k + n)^{\mathcal{O}(1)}$, which gives roughly a $2^{k/2}$ multiplicative improvement over the naive algorithm.

For a broader context, we also remark that a more general variant of Permutation SAT is Permutation MaxSAT, where we ask for an assignment that satisfies as many constraints as possible (instead of asking

to satisfy all of them). Observe that $(2,1)$-Permutation SAT is equivalent to a problem of checking if a given directed graph is acyclic (and thus solvable in polynomial time) while $(2,1)$-Permutation MaxSAT is equivalent to finding a maximum acyclic subdigraph (which is NP-hard). A simple folklore dynamic programming algorithm solves $(2,1)$-Permutation MaxSAT in $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ time and this algorithm can be generalised to $(3,1)$-Permutation MaxSAT [BFK+12]. Kim and Gonçalves [KG13] proved that $(4,1)$-Permutation MaxSAT admits no $2^{o(k\log k)}n^{\mathcal{O}(1)}$ algorithm unless the Exponential Time Hypothesis fails.

4 Lower bound for Rabin games

Finally, in this section, we prove the main result of this paper, stated as Theorem 4.1 below.

Theorem 4.1. *Assuming the Exponential Time Hypothesis, there is no algorithm that solves Rabin games with n vertices and degree k in time $2^{o(k\log k)}\cdot n^{\mathcal{O}(1)}$.*

As mentioned earlier, we reduce from 4-Permutation SAT.

The reduction. Let $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be an instance of 4-Permutation SAT over k variables $\{y_1,\ldots,y_k\}$, where $C_1,\ldots,C_m$ are 4-clauses. We construct an instance of Rabin Game such that, in this instance, there is a strategy for Steven iff ϕ is satisfiable.

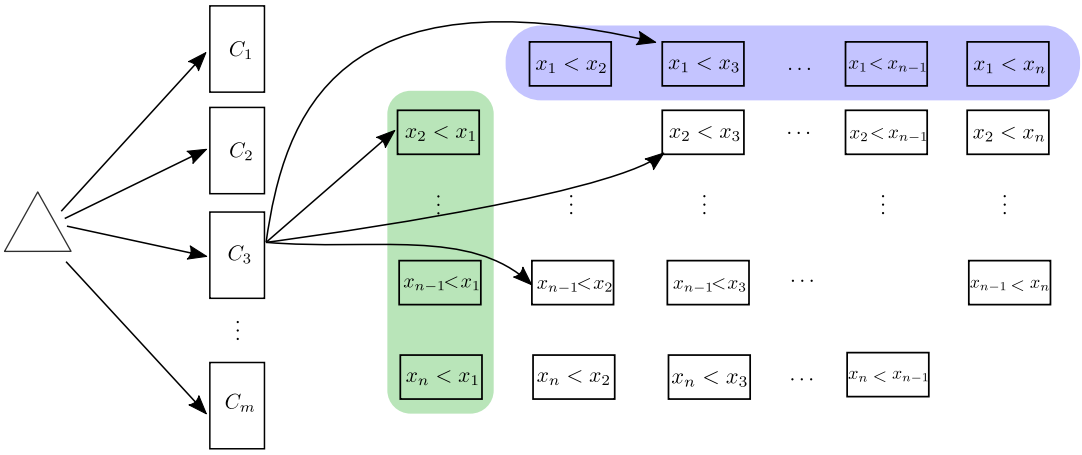

Figure 2: Part of the constructed game graph D. The clause C_3 is $(x_1 < x_3) \vee (x_2 < x_1) \vee (x_2 < x_3) \vee (x_{n-1} < x_2)$. Vertices of G_1 are highlighted in green and vertices of B_1 are highlighted in blue.

We first define the game graph D; see Figure 2. There is an initial vertex Δ, as well as vertices $[C_1],\ldots,[C_m]$, one for each of the m 4-clauses in ϕ. Further, for each possible literal $x_i < x_j$, where $i,j \in \{1,\ldots,k\}$ and $i \neq j$, there is a vertex $[x_i < x_j]$. Vertex Δ belongs to Audrey, while all other vertices belong to Steven.

The intention is that whenever Audrey moves the token currently placed at Δ, she chooses a clause that she wishes to see satisfied. To facilitate this, we add edges $\Delta \to [C_\ell]$ for all $\ell \in \{1,\ldots,m\}$. Once the token is at a vertex $[C_\ell]$, Steven needs to respond with a literal present in C_i; the intention is for it to be a true literal in C_i. Therefore, for every clause C_ℓ and literal $x_i < x_j$ present in C_ℓ, we add the edge $[C_\ell] \to [x_i < x_j]$. Finally, to allow Audrey checking further clauses, we add edges back to Δ: for every literal $x_i < x_j$, there is an edge $[x_i < x_j] \to \Delta$.

Next, we define subset pairs constituting the winning condition. For each $i \in \{1, \ldots, k\}$, we set

$$G_i = \{[x_j < x_i] \colon j \in \{1, \ldots, k\} \setminus \{i\}\} \qquad \text{and} \qquad B_i = \{[x_i < x_j] \colon j \in \{1, \ldots, k\} \setminus \{i\}\}.$$

Before we proceed to the formal verification of the correctness of the reduction, let us give some intuition. It is easy to see that every third turn, the token is placed at vertex Δ. At each such moment, turn Audrey chooses to move the token to any vertex corresponding to a clause C_ℓ, with the intention of challenging Steven about the satisfaction of C_ℓ. Then Steven has to declare the literal that satisfies C_ℓ. If Steven tries to "cheat" by picking literals that cannot be extended to a full ordering of the variables, then the winning condition is designed in such a way that the play will be losing for him. Consider the illustration in Figure 2, where for an instance ϕ of 4-Permutation SAT which consists of m clauses such that the clause C_3 is $(x_1 < x_4) \vee (x_2 < x_1) \vee (x_2 < x_3) \vee (x_{n-1} < x_2)$. The vertices in G_1 are highlighted in green and the vertices in B_1 are highlighted in blue.

Lemma 4.2. *The instance ϕ of 4-Permutation SAT is satisfiable if and only if Steven has a winning strategy in the constructed Rabin game.*

Proof. First suppose ϕ is satisfiable, consider a satisfying permutation π. This gives rise to a (positional) winning strategy for Steven: For each vertex $[C_\ell]$, Steven picks the edge leading to the vertex $[x_i < x_j]$ corresponding to any literal of C_ℓ that is satisfied in π. Consider now any infinite play ρ where Steven obeys this strategy. Let L be the set of literals visited infinitely often by ρ, and let $i_{\max}$ be such that $x_{i_{\max}}$ is the variable that is the largest in π among variables appearing in the literals of L. We argue that ρ satisfies the constructed Rabin condition with the index $i_{\max}$ as a witness. This is because L intersects $G_{i_{\max}}$ as ρ visits $[x_i < x_{i_{\max}}]$ infinitely often for some i, while the intersection of L with $B_{i_{\max}}$ is empty, as ρ never visits any vertex $[x_{i_{\max}} < x_i]$ for any i.

Suppose now ϕ is not satisfiable. Then we need to show that Audrey can win against any positional strategy of Steven. Indeed, consider a fixed positional strategy of Steven: for each Steven vertex $[C_\ell]$ the strategy picks an edge $[C_\ell] \to [x_{a_\ell} < x_{b_\ell}]$ for some literal $x_{a_\ell} < x_{b_\ell}$ appearing in C_ℓ. Since ϕ is not satisfiable, the set $\{x_{a_\ell} < x_{b_\ell} \colon \ell \in [m]\}$ of all selected literals has a cycle. That is, there are variables $x_{c_1}, \ldots, x_{c_p}$ such that literals $x_{c_1} < x_{c_2}, x_{c_2} < x_{c_3}, \ldots, x_{c_{p-1}} < x_{c_p}, x_{c_p} < x_{c_1}$ are among those selected by Steven's strategy. Observe now that for the fixed Steven's positional strategy, Audrey may set up a counter strategy that repeatedly visits each of the vertices $[c_1 < c_2], [c_2 < c_3], \ldots, [c_{p-1} < c_p], [c_p < c_1]$ in a cycle, so that these are exactly the literal vertices visited infinitely often in the play. Then this play does not satisfy the constructed Rabin condition, since for each $i \in \{1, \ldots, k\}$, the set of vertices occurring infinitely often either intersects both B_i and G_i (if $i \in \{c_1, \ldots, c_p\}$), or is disjoint with both B_i and G_i (if $i \notin \{c_1, \ldots, c_p\}$). Hence, Audrey may win against any fixed positional strategy of Steven. □

Using the reductions shown in the preliminaries, we obtain similar corollaries for Muller and generalised parity objectives.

Corollary 4.3. *Assuming the Exponential Time Hypothesis, there is no algorithm that solves Muller games with n vertices and k colours in time $2^{o(k \log k)} \cdot n^{\mathcal{O}(1)}$.*

Corollary 4.4. *Assuming the Exponential Time Hypothesis, there is no algorithm that solves d-dimensional 3-parity games with n vertices in time $2^{o(d \log d)} \cdot n^{\mathcal{O}(1)}$.*

We conclude by remarking that we can also extend our result to 2-dimensional k-parity games. Indeed, consider the following assignment of colours to the same game graph D: for each vertex of the form $[x_j < x_i]$, we assign the two-dimensional colour $(2j + 1, 2i)$. The correctness of this reduction is similar to that for Rabin games presented above, hence we leave the verification to the reader.

Corollary 4.5. *Assuming the Exponential Time Hypothesis, there is no algorithm that solves 2-dimensional k-parity games with n vertices in time $2^{o(k \log k)} \cdot n^{\mathcal{O}(1)}$.*

Acknowledgements. A large part of the results presented in this paper were obtained during Autobóz 2023, an annual research camp on automata theory. The authors thank the organisers and participants of Autobóz for creating a wonderful research atmosphere.

References

[BFK+12] Hans L. Bodlaender, Fedor V. Fomin, Arie M. C. A. Koster, Dieter Kratsch, and Dimitrios M. Thilikos. A note on exact algorithms for vertex ordering problems on graphs. *Theory Comput. Syst.*, 50(3):420–432, 2012.

[BK10] Manuel Bodirsky and Jan Kára. The complexity of temporal constraint satisfaction problems. *J. ACM*, 57(2):9:1–9:41, 2010.

[BKN+18] Marthe Bonamy, Lukasz Kowalik, Jesper Nederlof, Michał Pilipczuk, Arkadiusz Socała, and Marcin Wrochna. On directed feedback vertex set parameterized by treewidth. In *44th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2018*, volume 11159 of *Lecture Notes in Computer Science*, pages 65–78. Springer, 2018.

[BMM+22] Tamajit Banerjee, Rupak Majumdar, Kaushik Mallik, Anne-Kathrin Schmuck, and Sadegh Soudjani. A direct symbolic algorithm for solving stochastic Rabin games. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 81–98. Springer, 2022.

[CFK+15] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[CHP07] Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. Generalized parity games. In *10th International Conference on Foundations of Software Science and Computational Structures, FOSSACS 2007*, volume 4423 of *Lecture Notes in Computer Science*, pages 153–167. Springer, 2007.

[CJK+22] Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasi-polynomial time. *SIAM Journal on Computing*, 51(2):STOC17–152–STOC17–188, 2022.

[EJ88] E. Allen Emerson and Charanjit S. Jutla. The complexity of tree automata and logics of programs (extended abstract). In *29th Annual Symposium on Foundations of Computer Science, FOCS 1988*, pages 328–337. IEEE Computer Society, 1988.

[EJ99] E. Allen Emerson and Charanjit S. Jutla. The complexity of tree automata and logics of programs. *SIAM Journal on Computing*, 29(1):132–158, 1999.

[Eri19] Leif Eriksson. Solving temporal CSPs via enumeration and SAT compilation, 2019. MSc thesis.

[HD05] Paul Hunter and Anuj Dawar. Complexity bounds for regular games. In *30th International Symposium on Mathematical Foundations of Computer Science, MFCS 2005*, volume 3618 of *Lecture Notes in Computer Science*, pages 495–506. Springer, 2005.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

[KG13] Eun Jung Kim and Daniel Gonçalves. On exact algorithms for the permutation CSP. *Theor. Comput. Sci.*, 511:109–116, 2013.

[KV98] Orna Kupferman and Moshe Y. Vardi. Weak alternating automata and tree automata emptiness. In *30th Annual ACM Symposium on the Theory of Computing, STOC 1998*, pages 224–233. ACM, 1998.

[LMS18] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly superexponential parameterized problems. *SIAM Journal on Computing*, 47(3):675–702, 2018.

[MST23] Rupak Majumdar, Irmak Sağlam, and K. S. Thejaswini. Rabin games and colourful universal trees. Unpublished, 2023.

[Pit06] Nir Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. In *21st Annual IEEE Symposium on Logic in Computer Science, LICS 2006*, pages 255–264, 2006.

[PP06] Nir Piterman and Amir Pnueli. Faster solutions of Rabin and Streett games. In *21st Annual IEEE Symposium on Logic in Computer Science, LICS 2006*, pages 275–284, 2006.

[Rab69] Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[Saf88] Shmuel Safra. On the complexity of omega-automata. In *29th Annual Symposium on Foundations of Computer Science, FOCS 1988*, pages 319–327. IEEE Computer Society, 1988.

[Sch09] Sven Schewe. Tighter bounds for the determinisation of Büchi automata. In *12th International Conference on Foundations of Software Science and Computational Structures, FOSSACS 2009*, volume 5504 of *Lecture Notes in Computer Science*, pages 167–181. Springer, 2009.