SIAM J. COMPUT.
Vol. 51, No. 2, pp. STOC17-152–STOC17-188
© 2020 Society for Industrial and Applied Mathematics

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

DECIDING PARITY GAMES IN QUASI-POLYNOMIAL TIME*

CRISTIAN S. CALUDE†, SANJAY JAIN‡, BAKHADYR KHOUSSAINOV†, WEI LI§, AND FRANK STEPHAN‡§

Abstract. It is shown that the parity game can be solved in quasi-polynomial time. The parameterized parity game—with n nodes and m distinct values (a.k.a. colors or priorities)—is proven to be in the class of fixed parameter tractable problems when parameterized over m. Both results improve known bounds, from runtime $n^{O(\sqrt{n})}$ to $O(n^{\log(m)+6})$ and from an **XP** algorithm with runtime $O(n^{\Theta(m)})$ for fixed parameter m to a fixed parameter tractable algorithm with runtime $O(n^5 + 2^{m\log(m)+6m})$. As an application, it is proven that colored Muller games with n nodes and m colors can be decided in time $O((m^m \cdot n)^5)$; it is also shown that this bound cannot be improved to $2^{o(m\cdot\log(m))} \cdot n^{O(1)}$ in the case that the exponential time hypothesis is true. Further investigations deal with memoryless Muller games and multidimensional parity games.

Key words. parity game, quasi-polynomial time algorithm, fixed parameter tractability, Muller game, lower bounds

AMS subject classifications. 68Q25, 68Q70

DOI. 10.1137/17M1145288

1. Introduction. A parity game is given by a directed graph (V, E), a starting node $s \in V$, and a function val which attaches to each $v \in V$ an integer value (also called color) from a set $\{1, 2, 3, \ldots, m\}$; the main parameter of the game is n, the number of nodes, and the second parameter is m. Two players, Anke and Boris, move alternately in the graph, with Anke moving first. A move from a node v to another node w is valid if (v, w) is an edge in the graph; furthermore, it is required that from every node one can make at least one valid move. The alternate moves by Anke and Boris and Anke and Boris and ... define an infinite sequence of nodes which is called a play. For the evaluation, it is defined that each value is owned by one player; without loss of generality one player owns the odd numbers and the other player owns the even numbers. Anke wins a play through nodes $a_0, a_1, a_2, \ldots$ iff the limit superior (that is, the largest value appearing infinitely often) of the sequence $\mathrm{val}(a_0), \mathrm{val}(a_1), \mathrm{val}(a_2), \ldots$ is a number she owns, that is, a number of her parity. An

*Received by the editors August 28, 2017; accepted for publication (in revised form) August 29, 2019; published electronically January 14, 2020. A conference version of this work was presented at the Symposium on Theory of Computing, STOC 2017 [11]. The authors were invited to give survey talks on parity games which included the results of this paper at the program Aspects of Computation 2017 of the Institute of Mathematical Sciences in Singapore, the conference Highlights of Logic, Games and Automata 2017 in London, the NII Shonan Meeting Logic and Computational Complexity 2017 in Shonan Village Center in Japan, the conference Developments in Language Theory DLT 2018 in Tokyo, and the International Workshop on Combinatorial Algorithms IWOCA 2018 in Singapore.

https://doi.org/10.1137/17M1145288

Funding: The work of the second author was supported in part by NUS grant C252-000-087-001. The work of the third author was supported in part by the Marsden Fund grant of the Royal Society of New Zealand. The work of the second, third, and fifth authors was supported in part by the Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE2016-T2-1-019 / R146-000-234-112.

†School of Computer Science, University of Auckland, Private Bag 92019, Auckland, New Zealand (cristian@cs.auckland.ac.nz, bmk@cs.auckland.ac.nz).

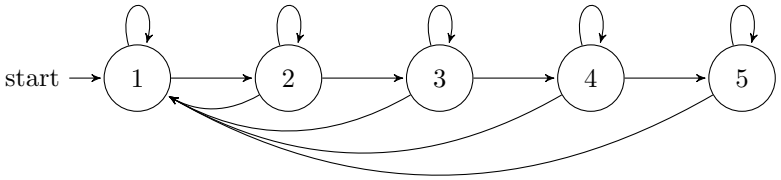
‡Department of Computer Science, National University of Singapore, Singapore 117417, Republic of Singapore (sanjay@comp.nus.edu.sg, fstephan@comp.nus.edu.sg).

§Department of Mathematics, National University of Singapore, Singapore 119076, Republic of Singapore (liwei.sg@gmail.com).

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

example is the following game:



Here the nodes are labeled with their values, which are unique (but this is not obligatory); furthermore, Anke has even and Boris has odd parity. Boris now has the following memoryless (that is, moves are independent of the history) winning strategy for this game: $1 \to 1$, $2 \to 3$, $3 \to 3$, $4 \to 5$, $5 \to 5$. Whenever the play leaves node 1 and Anke moves to node 2, then Boris will move to node 3. In the case that Anke moves to node 4, Boris will move to node 5. Hence, whenever the play is in a node with even value (this only happens after Anke moved it there), in the next step the play will go into a node with a higher odd value. So the largest infinitely often visited node value is odd, and therefore the limit superior of these numbers is an odd number which justifies Boris's win. Hence Boris has a winning strategy for the parity game given above.

Please see the next section for a more formal definition of the games and complexity classes discussed in this introduction.

It is known that for parity games, in general, the winner can always use a *memoryless winning strategy* [6, 29, 30, 32, 61, 62, 80]; see Corollary 21 below. This fact will be one central point in the results obtained in this paper: the parity game will be augmented with a special statistics—using polylogarithmic space—which indicates the winner correctly after a finite time whenever the winner employs a memoryless winning strategy. By the way, the existence of memoryless winning strategies is also a convenient tool to prove that solving parity games is in $\mathbf{NP} \cap \mathbf{coNP}$—fixing a memoryless strategy for one player transforms the parity game into a one-player game with a parity objective, and one can check in polynomial time whether this game can be won.

Parity games are a natural class of games which are not only interesting in their own right, but which are also connected to fundamental notions like μ-calculus, modal logics, tree automata, and Muller games [4, 7, 8, 18, 30, 32, 48, 71, 75, 76, 78, 79]. Faster algorithms for solving parity games could be used to improve the algorithms deciding the theory of certain tree automatic structures [35, 36, 58] and to employ them to understand these structures better.

For investigating the complexity side of the game, it is assumed that the game is given by a description in size polynomial in the number n of nodes and that one can evaluate all relevant parts of the description in logarithmic space. A possibility is to store the following three items for each game (where Anke moves first and starts from node 1):

- two numbers m, n with $1 \leq m \leq n$ and one bit which says whether the values owned by player Anke are the even or the odd numbers;
- the game graph given by a table, that is, for each pair of nodes, a bit which says whether there is a directed edge between the two nodes (which can be the same);
- the values of the nodes given by another table which holds, for each node, a binary number from $\{1, 2, 3, \ldots, m\}$.

An important open problem for parity games is the *time complexity* for finding the winner of a parity game when both players play optimally; the first algorithms took

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

exponential time [61, 80], and subsequent studies searched for better algorithms [51, 53, 55, 64, 70, 71, 72]. Many researchers, including Emerson and Jutla [32] in 1991, asked whether the winner of a parity game can be determined in polynomial time.

Emerson, Jutla, and Sistla [33] showed that the problem is in $\mathbf{NP} \cap \mathbf{coNP}$ and Jurdziński [52] improved this bound to $\mathbf{UP} \cap \mathbf{coUP}$. This indicates that the problem is not likely to be hard for **NP** and might be solvable faster than in exponential time. Indeed, Petersson and Vorobyov [64] devised a subexponential randomized algorithm and Jurdziński, Paterson, and Zwick [55] a deterministic algorithm of similar complexity (more precisely, the subexponential complexity was approximately $n^{O(\sqrt{n})}$).

Besides this main result, there are also various practical approaches to solving special cases [4, 26, 41] or testing out and analyzing heuristics [12, 44, 53]; however, when Friedmann and Lange [39] compared the various parity solving algorithms from the practical side, they found that Zielonka's recursive algorithm [80] was still the most useful one in practice.

McNaughton [61] showed that the winner of a parity game can be determined in time $n^{m+O(1)}$, and this was subsequently improved to $n^{m/2+O(1)}$ [9, 73] and to $n^{m/3+O(1)}$ [70, 72], where n is the number of nodes and m is the maximum value of the nodes.

The consideration of the parameter m is quite important for analyzing the algorithmic complexity of solving parity games; it is furthermore also a very natural choice. Schewe [71, 72] argued that for many applications which are solved using parity games, the parameter m is much smaller than n, often by an exponential gap.

For example, when translating colored Muller games into parity games in the way done by McNaughton [61] and Björklund, Sandberg, and Vorobyov [5], the number of values is, for all but finitely many games, bounded by the logarithm of the number of nodes; see the proof of Theorem 23 below. A similar result holds for the translation of multidimensional parity games into standard parity games.

A further important application of parity games is the area of reactive synthesis. Here one translates linear temporal logic formulas into a Büchi automaton which needs to be determined by translating it into a parity automaton. Building on the work of Safra [68, 69], Piterman [65] showed that one can translate nondeterministic Büchi automata with n states into parity automata with $2 \cdot n^n \cdot n!$ states and $2n$ values. In other words, one can evaluate various conditions on these parity automata by determining the winner in the corresponding parity game. Also Di Stasio et al. [25] investigated in their experiments various scenarios where the number m is logarithmic in n.

The present work therefore takes the parameter m into consideration and improves the time bounds in two ways:

- The overall time complexity is $O(n^{\lceil \log(m) \rceil + 6})$, which provides a quasi-polynomial bound on the runtime, as one can always choose $m \leq n$.
- Furthermore, if $m < \log(n)$, then the overall time complexity is $O(n^5)$, which shows that the problem is fixed parameter tractable when parameterized by m; the parity games are therefore in the lowest time complexity class usually considered in parameterized complexity.

Prior investigations have already established that various other parameterizations of parity games are fixed parameter tractable, but the parameterization by m was left open until now. Chatterjee [14] pointed out to the authors that one can also write the result in product form with parity games being solvable in time $O(2^m \cdot n^4)$ for all m, n; the proof uses just the methods of Theorem 16, but keeping m as a parameter and not using explicitly the bound of $m \leq \log(n)$ which, when invoked into the above

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

formula, would give the bound $O(n^5)$.

An application of the results presented here is that colored Muller games with n nodes and m colors can be decided in time $O((m^m \cdot n)^5)$; Theorem 25 below shows that this bound cannot be improved to $2^{o(m \cdot \log(m))} \cdot n^{O(1)}$, provided that the Exponential Time Hypothesis is true.

Subsequent research [34, 42, 54, 74] has provided the additional runtime bound

$$O(\lceil m/\log(n) \rceil^4 \cdot n^{3.45+\log(\lceil m/\log(n) \rceil+2)}),$$

where the bound cited here stems from Stephan's teaching material [74, Theorem 20.22], while the research papers [34, 42, 54] have obtained slightly better bounds due to some assumptions they make on the game and due to the usage of better bounds for binomials. However, the main contribution of the subsequent research [34, 54] is that the quasi-polynomial time algorithm can be modified such that, in addition to the time bound, the workspace the algorithm uses is only quasi-linear in the number of nodes n. This improves upon the algorithm presented here, which uses quasi-polynomial space. Furthermore, various authors provided their own version of the verification of the algorithm presented in this paper [34, 42, 54]. Before the presentation of the results, the next section summarizes the basic definitions and properties of the games and also provides the basic complexity classes needed. To make the paper self-contained, proofs of some known results, namely, Propositions 17 and 28 as well as Theorems 20, 22, and 23, have been written in a uniform manner and included in this paper.

2. Basic notions used. This section summarizes the basic properties of the two games (parity game and colored Muller game) and also explains related games (multi-dimensional parity game, Rabin game, and Streett game). It furthermore provides the basic complexity-theoretic notions used in this paper.

DEFINITION 1. *A* game *is given by a directed finite graph of n nodes, a starting node, and a set G of sets of nodes which are called the* winning set *of player Anke. The two players, Anke and Boris, alternately move a marker through the graph, where Anke starts from the starting node and the players each time move along an outgoing edge of the current node; here it is required that every node have at least one outgoing edge (which can go to the node itself). A* play *is the infinite sequence of nodes visited by the marker while Anke and Boris are playing. To decide the* winner of a play*, one considers the set of infinitely often visited nodes U. Now Anke wins the play iff $U \in G$.*

In a parity game*, each node v carries a value, denoted* $\mathrm{val}(v)$*. In a* colored Muller game*, each node v carries a set of colors. Note that the general game mentioned above is a (colored) Muller game where each node's color is identified with its name.*

In a parity game*, the set G can be derived from values from* 1 *to m (where $m \leq n$) which are associated with the nodes. For this, one associates with each player Anke and Boris a parity, and a set U is in G iff the maximum value of nodes in U is of Anke's parity. Alternatively one can require that G respects the parity; that is, if U and U' satisfy that the maximum values of nodes in U and in U', respectively, have the same parity, then either U and U' are both inside G or U and U' are both outside G.*

In a colored Muller game*, every node is associated with a set of colors. For a set U of nodes,* $\mathrm{color}(U)$ *is the set of all colors which are associated with at least one node in U. The set G has to respect the colors; that is, if* $\mathrm{color}(U) = \mathrm{color}(U')$*, then either both U and U' are inside G or both U and U' are outside G.*

*In a k-*dimensional parity game*, each node is associated with a k-dimensional vec-*

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

tor of values. Now a set U of nodes is winning for player Anke iff the componentwise maximum of the value vectors of the nodes in U is a vector of k odd numbers.

Rabin games *and* Streett games *have as additional information a list* $(V_1, W_1), (V_2, W_2), (V_3, W_3), \ldots, (V_m, W_m)$ *of pairs such that in the Rabin case a set of nodes is in U iff some pair (V_h, W_h) satisfies $V_h \cap U \neq \emptyset$ and $W_h \cap U = \emptyset$; in the Streett case, $U \in G$ iff all pairs (V_h, W_h) satisfy $V_h \cap U \neq \emptyset \Rightarrow W_h \cap U \neq \emptyset$.*

A strategy for a player*, say for Anke, maps, for every situation where Anke has to move, the current node and history of previous moves to a suggested move for Anke. A* winning strategy *for Anke is a strategy for Anke which guarantees that Anke wins a play whenever she follows the suggested moves. A strategy is called* memoryless *iff it only depends on the current node and not on any other aspects of the history of the play.*

The winner of a game *is the player who has a winning strategy for this game.*

Remark 2. All games considered in this paper (including parity games and colored Muller games) always have a winner; this winner wins every play in the case that the winner follows a winning strategy.

The additional structures of parity games, colored Muller games, and other games enforce that the winning set G is of a certain form; in particular in the case that the parameter m (number of colors of a colored Muller game or number of values of a parity game) is small compared to n, the algorithms to solve these games have a better time bound than in the general case.

As choosing for each node a unique color not shared with any other node does not impose any restriction on G, one can without loss of generality require that $m \leq n$.

For parity games, if a value $k > 1$ does not occur in a game, but $k+1$ does, then one can for all nodes v with $\mathrm{val}(v) > k$ replace $\mathrm{val}(v)$ by $\mathrm{val}(v) - 2$ without changing the winner of the game. Furthermore, if the value 1 does not occur in the game, then one can replace $\mathrm{val}(v)$ by $\mathrm{val}(v) - 1$ throughout the game and invert the parity of the players. For that reason, the maximum value m of a parity game can always be assumed to satisfy $m \leq n$.

In colored Muller games, representations of G as tables might have the size 2^m, and one has several choices of how to handle this situation: (a) one only considers such colored Muller games where G can be decided by a Boolean circuit not larger than $p(n)$ size for some polynomial p; (b) the same as (a) with a polynomial time algorithm instead with program size $p(n)$; (c) one uses the space needed for representing G as a Boolean circuit as an additional parameter for the game. The approach taken in the present paper is (a) or (b).

Remark 3. One can also consider games where the player moving depends only on the current node of the play and players do not necessarily take turns. Both versions of parity or Muller games can be translated into each other with a potential increase in the number of nodes by a factor 2.

In the case that one goes from turn-based to position-based Muller games, one doubles up each node: Instead of the node v, one uses a node (Anke, v) when it is Anke's turn to move, and a node (Boris, v) when it is Boris's turn to move; the nodes (Anke, v) and (Boris, v) in the new game have the same values or colors as v in the old game. For every edge from v to w in the old game, one puts the edges from (Anke, v) to (Boris, w) and from (Boris, v) to (Anke, w) into the game.

For the other direction, each node w receives a prenode w' with exactly one outgoing edge from w' to w. Now, for each edge (v, w) from the original game, if the same player moves at v and at w in the original game, then one puts the edge (v, w')

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

into the new game, else one puts the edge (v, w) into the new game. The rationale behind this is that the longer path $v - w' - w$ has even length in the case that the players moving at v and w should be the same for alternating moves. Furthermore, if Anke moves at the original starting node s, then s is also the starting node of the new game, else s' is the starting node of the new game. Again, the nodes w and w' in the new game have the same value or color as the node w in the old game.

Parameterized complexity studies the complexity of solving a problem in dependence of not only the main parameter n (size of input), but also other related parameters $m, k, \ldots$ which are expected to arise naturally from the problem description. In the following, let n denote the main parameter and m a natural further parameter.

DEFINITION 4. *A problem is called fixed parameter tractable (***FPT***) iff there is a polynomial p and a further function f such that all instances of the problem can be solved in time $f(m) + p(n)$.*

The class of all problems in **FPT** can also be characterized as those problems which can be solved in $g(m) \cdot p(n)$ for some polynomial p and an arbitrary function g.

For the current work, the main parameter n is the number of nodes, and the parameter m is the number of values in the parity game or the number of colors in the colored Muller game. The so chosen second parameter m is a very natural parameter to the games considered and occurs widely in prior work studying the complexity of the games [5, 9, 61, 70, 72, 73]. However, in the literature, other parameters and parameter combinations have also been studied.

The number m of colors used in the game is an important parameter of colored Muller games; for complexity-theoretic considerations, the exact complexity class of solving colored Muller games with n nodes and m colors may also depend on how G is represented, in particular in cases when m is large. The size of this representation can thus be a further parameter for determining the complexity class of solving colored Muller games. However, this parameter is not studied in the present work.

DEFINITION 5. *A problem is in the class* **XP** *if it can be solved in time $O(n^{f(m)})$ for some function f.*

Between **FPT** at the bottom and **XP** at the top, there are the levels of the **W**-hierarchy **W[1]**, **W[2]**, **W[3]**, ...; it is known that **FPT** is a proper subclass of **XP**, and it is widely believed that the levels of the **W**-hierarchy are all different. The books of Downey and Fellows [27, 28] and Flum and Grohe [37] give further information on parameterized complexity.

Given as input a conjunctive normal form Boolean formula, **SAT** is the problem of determining whether the formula is satisfiable. **3SAT** and **4SAT**, respectively, denote the restriction of **SAT** to conjunctive normal form formulas where each clause has at most three (respectively, four) literals.

DEFINITION 6. *The Exponential Time Hypothesis says that for the usual satisfiability problems like* **3SAT***,* **4SAT***, and* **SAT***, for n being the number of variables in the formula, any algorithm determining whether the formula is satisfiable needs worst case time at least c^n for some rational number $c > 1$ and almost all n.*

The Exponential Time Hypothesis implies that **W[1]** differs from **FPT**, but the converse is not known. Note that the **NP**-complete problems are spread out over all levels of this hierarchy and that even the bottom level **FPT** also contains sets outside **NP**. The level of a problem can depend on the choice of the parameters to describe

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

the problem, and therefore one has to justify the choice of the parameters.

Chandra, Kozen and Stockmeyer [13] investigated alternating Turing machines. Such machines can be defined in an asymmetric and a symmetric way; the latter is in particular needed for lower complexity bounds in certain settings. Furthermore, Cook [23] and Levin [60] initiated the systematic study of **NP** and formalized the question of whether $\mathbf{NP} = \mathbf{P}$.

DEFINITION 7. *Alternating Turing machines can be viewed as a game: Besides the usual Turing machine steps, there are also branching Turing machine steps. In the case of an existential branching, one player, say Anke, decides which of the possible steps the Turing machine is taking; in the case of a universal branching, the other player, here Boris, decides which of the possible steps the Turing machine is taking. Anke wins iff Anke can always force the game into an accepting state. Boris wins iff the game never goes into an accepting state. Now for every x as input, one of the players has a winning strategy; the alternating Turing machine decides L iff the following holds: For all $x \in L$, Anke has a winning strategy; for all $x \notin L$, Boris has a winning strategy.*

A language L is in alternating time/space $f(n)$ iff for every $x \in L$ with $|x| = n$, Anke can play such that x is accepted and the play does not violate the resource bound $f(n)$; for $x \notin L$, Boris can play such that x is never accepted and, in the case of a space resource bound, the play does not violate the resource bound.

A language L is in nondeterministic time/space $f(n)$ iff it is in an alternating time/space $f(n)$ via a Turing machine where Boris always has only one choice. A language is in $\mathbf{NP} \cap \mathbf{coNP}$ *iff there are a nondeterministic Turing machine and a polynomial p such that if $L(x) = a$, then Anke can play such that the input (x, a) is accepted within time $p(|x|)$, and if $L(x) \neq a$, then Anke cannot achieve that (x, a) gets accepted. A language is in* $\mathbf{UP} \cap \mathbf{coUP}$ *iff it is in* $\mathbf{NP} \cap \mathbf{coNP}$ *via a machine which has, for every pair $(x, L(x))$, exactly one computation path which Anke can choose such that $(x, L(x))$ gets accepted.*

A language L satisfies $L \in \mathbf{\Sigma_2^P}$ *iff there is an alternating Turing machine recognizing L in polynomial time such that on every computation path, all the points where Anke can branch the computation come before those points where Boris can branch the computation.*

In the case of alternating computation, for small complexity classes where one cannot check the complexity within the mechanism given, one employs for alternating computations a symmetric setting where the alternating Turing machine has explicit accepting and explicit rejecting states and it halts in both. Now L is in the given time class iff the following holds: For all $x \in L$, Anke has a winning strategy which guarantees that, while obeying the given resource bound, the game ends up in an accepting state; for all $x \notin L$, Boris has a winning strategy which guarantees that, while obeying the given resource bound, the game ends up in a rejecting state.

If the space bound or the time bound are constructible within the given complexity class, then the alternating computation for the standard model can also be equipped with a counter; then the machine can go to the rejecting state when the runtime is exhausted; here one uses that if an alternating machine using space $f(n)$ does not accept within $c^{f(n)}$ steps for a suitable constant c, then one can safely reject the computation. The first approach to solving the parity games in polylogarithmic space below also has this symmetric approach implicitly, even without using explicit counters for the used up time.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

3. The complexity of the parity game. The main result in this section is an alternating polylogarithmic space algorithm to decide the winner in parity games; later, more concrete bounds will be shown. The idea is to collect, in polylogarithmic space, for both players in the game, Anke and Boris, the statistics of their performance in the play. In particular, these statistics store information about whether the play has surely gone through a loop where the largest valued node has the parity of the corresponding player. Though these statistics do not capture all such loops, in the case that one player plays a memoryless winning strategy, the player's own statistics will eventually find evidence for such a loop, while the opponent statistics will not provide false evidence which would lead in the opposite direction.

The following notation will be used throughout the paper. In order to avoid problems with fractional numbers and $\log(0)$, let $\lceil \log(k) \rceil = \min\{h \in \mathbb{N} : 2^h \geq k\}$. Furthermore, a function (or sequence) f is called *increasing* whenever for all i, j the implication $i \leq j \Rightarrow f(i) \leq f(j)$ holds.

THEOREM 8. *There exists an alternating polylogarithmic space algorithm deciding which player has a winning strategy in a given parity game. When the game has n nodes with values in the set $\{1, 2, 3, \ldots, m\}$, then the algorithm runs in $O(\log(n) \cdot \log(m))$ alternating space.*

Proof. The idea of the proof is that, in each play of the parity game, one maintains winning statistics for both players Anke and Boris. These statistics are updated after every move for both players. In case a player plays according to a memoryless winning strategy for the parity game, the winning statistics of this player will eventually indicate the win (in this case one says that the "winning statistics of the player mature"), while the opponent's winning statistics will never mature. This will be explained in more detail below.

The winning statistics of Anke (Boris) has the following goal: to track whether the play goes through a loop where the largest value of a node in the loop is of Anke's (Boris's) parity. Note that if Anke follows a memoryless winning strategy, then the play will eventually go through a loop and the node with the largest value occurring in any loop the play goes through is always a node of Anke's parity. Otherwise, Boris can repeat a loop with the largest value being of Boris's parity infinitely often and thus win, contradicting that Anke is using a memoryless winning strategy.

The naïve method to do the tracking is to archive the last $2n + 1$ nodes visited, to find two identical moves out of the same node by the same player, and to check whose parity has the largest value between these two moves. This would determine the winner in case the winner uses a memoryless winning strategy. This tracking needs $O(n \cdot \log(n))$ space—too much space for the intended result. To save space one constructs a winning statistics which still leads to an Anke win in case Anke plays a memoryless winning strategy, but memorizes only partial information.

The winning statistics of the players are used to track whether certain sequences of nodes have been visited in the play so far, and the largest value of a node visited at the end or after the sequence is recorded. The definitions are similar for both players. For simplicity the definition is given here just for player Anke.

DEFINITION 9. *In Anke's winning statistics, an i-sequence is a sequence of nodes $a_1, a_2, , a_3, , \ldots, a_{2^i}$ which have been visited (not necessarily consecutively, but in order) during the play so far such that, for each $k \in \{1, 2, 3, \ldots, 2^i - 1\}$,*

$$\max\{\mathrm{val}(a) : a = a_k \vee a = a_{k+1} \vee a \text{ was visited between } a_k \text{ and } a_{k+1}\}$$

is of Anke's parity.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

The aim of Anke is to find a sequence of length at least $2n+1$, as such a sequence must contain a loop. So she aims for a $(\lceil \log(n) \rceil + 2)$-sequence to occur in her winning statistics. Such a sequence is built by combining smaller sequences over time in the winning statistics.

Here a winning statistics $(b_0, b_1, \ldots, b_{\lceil \log(n) \rceil + 2})$ of a player consists of $\lceil \log(n) \rceil + 3$ numbers between 0 and m, both inclusive, where $b_i = 0$ indicates that currently no i-sequence is being tracked and $b_i > 0$ indicates the following:

> **Property-b_i**: an i-sequence is being tracked, and the largest value of a node visited at the end of or after this i-sequence is b_i.

Note that for each i, at most one i-sequence is tracked. The value b_i is the only information of an i-sequence which is kept in the winning statistics.

The following invariants are kept throughout the play and are formulated for Anke's winning statistics; those for Boris's winning statistics are defined with the names of Anke and Boris interchanged. In the description below, "i-sequence" always refers to the i-sequence being tracked in the winning statistics.

(I1) Only b_i with $0 \leq i \leq \lceil \log(n) \rceil + 2$ are considered, and each such b_i is either zero or a value of a node which has occurred in the play so far.

(I2) An entry b_i refers to an i-sequence which has occurred in the play so far iff $b_i > 0$.

(I3) If b_i, b_j are both nonzero and $i < j$, then $b_i \leq b_j$.

(I4) If b_i, b_j are both nonzero and $i < j$, then in the play of the game so far, the i-sequence starts only after a node with value b_j was visited at or after the end of the j-sequence.

When a play starts, the winning statistics for both players are initialized with $b_i = 0$ for all i. During the play when a player moves to a node with value b, the winning statistics of Anke is updated as follows (the same algorithm is used for Boris, with the names of the players interchanged everywhere):

1. If b is of Anke's parity or $b > b_i > 0$ for some i, then one selects the largest i such that either
   (a) b_i is not of Anke's parity—that is, it is either 0 or of Boris' parity—but all b_j with $j < i$ and also b are of Anke's parity, or
   (b) $0 < b_i < b$,

   and then one updates $b_i = b$ and $b_j = 0$ for all $j < i$.
2. If this update produces a nonzero b_i for any i with $2^i > 2n$, then the play terminates with Anke being declared winner.

Note that it is possible that both 1(a) and 1(b) apply to the same largest i. In that case, it does not matter which case is chosen, as the updated winning statistics is the same for both cases. However, the tracked i-sequences referred to may be different; this does not affect the rest of the proof.

Example 10. Here is an example of i-sequences for player Anke. This example is only for illustrating how the i-sequences and b_i's work; in particular this example does not use memoryless strategy for either of the players. Consider a game where there is an edge from every node to every node (including itself) and the nodes are $\{1, 2, 3, \ldots, 7\}$ and have the same values as names; Anke has odd parity. Consider the following initial part of a play:

```
1 6 7 5 1 4 5 3 2 1 3 2 3 1 3 3 1 2 1
```

The i-sequences and the b_i's change over the course of the above play as given in the following table. In the table, the nodes prefixed by "i :" are those of the corresponding i-sequence.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Move	b_4, b_3, b_2, b_1, b_0	i-sequences in play so far	Rule
1	0,0,0,0,1	0:1	1(a)
6	0,0,0,0,6	0:1 6	1(b)
7	0,0,0,0,7	1 6 0:7	1(a)
5	0,0,0,5,0	1 6 1:7 1:5	1(a)
1	0,0,0,5,1	1 6 1:7 1:5 0:1	1(a)
4	0,0,0,5,4	1 6 1:7 1:5 0:1 4	1(b)
5	0,0,0,5,5	1 6 1:7 1:5 1 4 0:5	1(a)
3	0,0,3,0,0	1 6 2:7 2:5 1 4 2:5 2:3	1(a)
2	0,0,3,0,0	1 6 2:7 2:5 1 4 2:5 2:3 2	
1	0,0,3,0,1	1 6 2:7 2:5 1 4 2:5 2:3 2 0:1	1(a)
3	0,0,3,3,0	1 6 2:7 2:5 1 4 2:5 2:3 2 1:1 1:3	1(a)
2	0,0,3,3,0	1 6 2:7 2:5 1 4 2:5 2:3 2 1:1 1:3 2	
3	0,0,3,3,3	1 6 2:7 2:5 1 4 2:5 2:3 2 1:1 1:3 2 0:3	1(a)
1	0,1,0,0,0	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1	1(a)
3	0,3,0,0,0	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1 3	1(b)
3	0,3,0,0,3	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1 3 0:3	1(a)
1	0,3,0,1,0	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1 3 1:3 1:1	1(a)
2	0,3,0,2,0	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1 3 1:3 1:1 2	1(b)
1	0,3,0,2,1	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1 3 1:3 1:1 2 0:1	1(a)

If at an update of an i-sequence both possible updates 1(a) and 1(b) apply to the same level i, then it does not matter for the statistics which is chosen. However, for the i-sequences, one has to commit to one choice, and for simplicity (for the above table) one assumes that 1(a) has priority. So the formal algorithm for updating the sequences is as follows:

1. If b is of Anke's parity or $b > b_i > 0$ for some i, then one selects the largest i such that either
   (a) b_i is not of Anke's parity—that is, it is either 0 or of Boris's parity—but all b_j with $j < i$ and also b are of Anke's parity, or
   (b) $0 < b_i < b$;
   else there is no update and one goes to step 3.
2. For the selected i, one does the following update according to the first of the two above cases that applies:
   (a) Let $b_i = b$.
   Let the new i-sequence contain all the nodes of the old j-sequences, with $j < i$, plus the new node with value b.
   Let $b_j = 0$ for all $j < i$ as the corresponding j-sequences are merged into the new i-sequence.
   (b) Let $b_i = b$, and let the i-sequence be unchanged except for the update of the associated value b_i, and all j-sequences with $j < i$ are made void by setting $b_j = 0$ for all $j < i$.
   Furthermore, all j-sequences with $j > i$ are maintained as they are.
3. If this update produces a nonzero b_i for any i with $2^i > 2n$, then the play terminates with Anke being declared winner and no further tracking of i-sequences is needed.

The 3-sequence in the above table already has a loop, as there are three occurrences of "$3:3$" and the second and third of these have that the same player moves. However, as the sequences are not stored but only the b_i, Anke's winning statistics only surely indicates a win for player Anke when there is an $i \geq \log(2n+1)$ with $b_i > 0$; this i is 4 as $2^4 > 2 \cdot 7 + 1$.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Before proceeding to the verification of the algorithm correctness, an outline of the strategy is given.

Remark 11. The winning statistics of both players are maintained via a deterministic algorithm which updates each statistic based on the prior value and the current node visited—more precisely, the value of the node visited. These statistics use only $O(\log(m)\cdot\log(n))$ bits of memory. If a player, during a play, follows a memoryless winning strategy, then the player's winning statistics will eventually indicate a win, while the opponent's winning statistics never will. However, if neither of the players follows a memoryless winning strategy, then no guarantees on the outcome of the evolution of the statistics are made. Furthermore, if one identifies "Anke's winning strategy indicates a win" with "accept" and "Boris's winning strategy indicates a win" with "reject," then one can view the game as a run of an alternating $O(\log(n)\cdot\log(m))$ space Turing machine which keeps in its memory only the statistics, the current node, and the player to move and which explicitly accepts a computation in the case that Anke can win the game and explicitly rejects a computation in the case that Boris can win the game. For the case of checking whether Anke can win, the existential branchings are the choice of the next move by Anke, and the universal branchings are the choice of the next move by Boris. The obtained characterization is heavily based on the fact that in every parity game one of the players has a memoryless winning strategy; see Corollary 21 below. One can approximately halve the space usage by maintaining only Anke's winning statistics. If the winning player plays a memoryless winning strategy, then the alternating Turing machine would explicitly accept if Anke can win and will reject by "running forever" without ever visiting an accepting state in the case that Boris can win.

An anonymous referee suggested that such an algorithm—which maintains the winning statistics—might be called a "space-efficient one-pass streaming algorithm inspecting the play."

Verification that the algorithm is correct. Note that, in the updating algorithm for Anke's winning statistics, if b is of Anke's parity, then there is an i that satisfies 1(a), as otherwise the algorithm would have terminated earlier. Initially, the invariants clearly hold as all b_i's are 0. Now it is shown that the invariants are preserved at updates of the b_i's according to case 1(a) or 1(b).

It is easy to verify that the invariants are maintained if the update is due to 1(b), and it also ensures that Property-b_i is maintained for the i-sequences being tracked. In case the update is done due to 1(a), then Property-$b_{i'}$ is maintained for all i'-sequences being tracked for $i' > i$ (with $b_{i'} \geq b$ in these cases). For $i' < i$, $b_{i'}$ is made 0 by the update algorithm. The next paragraph argues about an appropriate i-sequence being formed. Thus, it is easy to verify that (I1) to (I4) are maintained by the update algorithm. Note that (I1) implies that the space bound needed is at most $O(\log n \log m)$, (I2) is used implicitly to indicate which i-sequences are being tracked, and (I3), (I4) give the order of the i-sequences tracked: a $(j+1)$-sequence appears earlier in the play than a j-sequence. This is used implicitly when one combines the smaller j-sequences into a larger one as mentioned below.

When updating Anke's winning statistics by case 1(a), one forms a new i-sequence of length 2^i by putting the older j-sequences for $j = i-1, i-2, \ldots, 1, 0$ together and appending the newly visited one-node sequence with value b; when $i = 0$, one forms a new 0-sequence of length 2^0 consisting of just the newly visited node with value b. Note that in case $i > 0$ both b and b_0 are of Anke's parity, and therefore the highest valued node between the last member a of the older 0-sequence and the last node in

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

the new i-sequence (both inclusive) has the value $\max\{b_0, b\}$ (by (I4) and Property-b_0 for the older 0-sequence). Furthermore, for every $j < i-1$, for the last node a of the older $(j+1)$-sequence and the first node a' of the older j-sequence, in the new i-sequence a highest valued node in the play between these two nodes a, a' (both inclusive) has value b_{j+1} (by (I4) and Property-b_{j+1} of the older $(j+1)$-sequence) which, by choice, has Anke's parity. Thus the overall combined new sequence indeed satisfies the properties needed for an i-sequence, and b is the value of the last node of this sequence and thus, currently, also the largest value of a node visited at or after the end of the sequence. All older j-sequences with $j < i$ are discarded, and thus their entries are set back to $b_j = 0$.

The same rules apply to the updates of Boris's winning statistics with the roles of Anke and Boris interchanged everywhere.

CLAIM 12. *If a player is declared a winner by the algorithm, then the play contains a loop, with its maximum valued node being a node of the player.*

To prove the claim, it is assumed without loss of generality that Anke is declared the winner by the algorithm. The play is won by an i-sequence being observed in Anke's winning statistics with $2^i > 2n$; thus some node occurs at least three times in the i-sequence and there are $h, \ell \in \{1, 2, 3, \ldots, 2^i\}$ with $h < \ell$ such that the same player moves at a_h and a_ℓ and furthermore $a_h = a_\ell$ with respect to the nodes $a_1, a_2, a_3, \ldots, a_{2^i}$ of the observed i-sequence. The maximum value b' of a node between a_h and a_ℓ in the play is occurring between some a_k and a_{k+1} (both inclusive) for a k with $h \leq k < \ell$. Now, by the definition of an i-sequence, b' has Anke's parity. Thus a loop has been observed for which the maximum value of a node in the loop has Anke's parity.

CLAIM 13. *If a player follows a memoryless winning strategy, then the opponent is never declared a winner.*

To prove the claim, suppose that a player follows a memoryless winning strategy but the opponent is declared a winner. Then the opponent, by Claim 12, goes into a loop with the maximum node of the opponent's parity. Hence, the opponent can cycle in that loop forever and win the play, a contradiction.

CLAIM 14. *If a player follows a memoryless winning strategy, then the player is eventually declared a winner.*

To prove the claim, it is assumed that the player is Anke, as the case of Boris is symmetric. The values b_i analyzed below refer to Anke's winning statistics. Assume that the sequence of values of the nodes in an infinite play of the game has the limit superior c which, by assumption, is a value of Anke's parity. To prove the claim one needs to argue that eventually b_i becomes nonzero for an i with $2^i > 2n$. For this purpose it will be argued that a counter to be defined, associated with the values of b_i's, eventually keeps increasing (except for some initial part of the play, where it may oscillate). This is argued by using $\mathrm{count}(c, t)$ below, which gives the value of the counter after t steps of the play.

Consider a step as making a move and updating of the statistics. For each step t let $b_k(t)$ refer to the value of b_k at the end of step t (that is, after the updates in the statistics following the tth move in the play). Let $B_c(t)$ be the set of all k such that $b_k(t)$ has Anke's parity and $b_k(t) \geq c$. Let

$$\mathrm{count}(c, t) = \textstyle\sum_{k \in B_c(t)} 2^k.$$

Now it is shown that whenever at steps t, t' with $t < t'$ a move to a node with value c

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

was made and no move, strictly between steps t, t', was made to any node with value $c' \geq c$, then $\text{count}(c,t) < \text{count}(c,t')$. To see this, let i be the largest index for which there is a step t'' with $t < t'' \leq t'$ such that b_i is updated at step t''.

Note that this implies [$b_i(t) < c$ or $b_i(t)$ is of Boris's parity], and [$0 < b_i(t'') \leq c$]. Now, in the case that $b_i(t'') < c$, it holds that $t'' < t'$ and at time t', condition 1(b) of the update algorithm will ensure that an update (either 1(a) or 1(b)) is done to enforce $b_i(t') = c$. Thus

$$\text{count}(c,t') - \text{count}(c,t) \geq 2^i - \textstyle\sum_{j \in B_c(t): j<i} 2^j \geq 1.$$

Accordingly, once all moves involving nodes larger than c in value have been done in the play, there will still be infinitely many moves to nodes of value c, and for each two subsequent such moves at t, t' the inequality $\text{count}(c,t) + 1 \leq \text{count}(c,t')$ will hold. Consequently, the number $\text{count}(c,t)$, for sufficiently large t where a move to a node with value c is made at step t, needs to have, for some i, $b_i(t) \geq c$ and $2^i > 2n$; thus the termination condition of Anke will terminate the play with a win.

The above arguments show that an alternating Turing machine can simulate both players and, taking the winning statistics into account, will accept the computation whenever Anke has a winning strategy for the game.

Recall that an alternating Turing machine can be viewed as a game between two players, Anke (existential) and Boris (universal), who perform in turns part of the computations and can branch in the part they do; when the game terminates, it says which player has won; if Anke wins, it means "accept," and if Boris wins, it means "reject"; if it never terminates, it means "undecided."

An alternating Turing machine can decide a set iff for every input x, if $x \in L$, then Anke has a winning strategy for the alternating Turing machine and can force an "accept," else Boris has a winning strategy for the alternating Turing machine and can avoid that it comes to an "accept"; in the case of the above game, Boris can even enforce an explicit "reject." For the alternating Turing machine, in order to simulate the game, one has to keep track of the following pieces of information: the winning statistics of the players; the current node in the play and the player who is to move next. Thus, the alternating Turing machine uses only $O(\log(n) \cdot \log(m))$ space to decide whether the parity game, from some given starting point, will be won by Anke (or Boris), provided the winner plays a memoryless winning strategy (which always exists when the player can win the parity game). ∎

Chandra, Kozen, and Stockmeyer [13] showed how to simulate an alternating Turing machine working in polylogarithmic space by a deterministic Turing machine working in quasi-polynomial time. Their simulation bounds for the alternating Turing machine described in Theorem 8 give a deterministic Turing machine working in time $O(n^{c \log(m)})$ for some constant c. As mentioned above, one can always assume that in a parity game with n nodes and values from $\{1, 2, 3, \ldots, m\}$, one can choose $m \leq n$, so using this result one gets the following parameterized version of the main results that parity games can be solved in quasi-polynomial time.

THEOREM 15. *There is an algorithm which finds the winner of a parity game with n nodes and values from $\{1, 2, 3, \ldots, m\}$ in time $O(n^{c \log(m)})$.*

For some special choices of m with respect to n, one can obtain even a polynomial time bound. McNaughton [61] showed that for every constant m, one can solve a parity game with n nodes having values from $\{1, 2, 3, \ldots, m\}$ in time polynomial in n; however, in all prior works the degree of this polynomial depends on m [40]; subsequent improvements were made to bring the dependence from approximately $n^{m+O(1)}$ first

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

down to $n^{m/2+O(1)}$ [9, 73] and then to approximately $n^{m/3+O(1)}$ [53, 72]. The following theorem shows that one can bound the computation time by a fixed-degree polynomial in n for all pairs (m, n) with $m < \log(n)$.

THEOREM 16. *If $m \leq \log(n)$, then one can solve the parity game with n nodes and values from $\{1, 2, 3, \ldots, m\}$ in time $O(n^5)$.*

Proof. Note that Theorem 8 actually shows that the following conditions are equivalent:

- Anke can win the parity game.
- Anke can play the parity game such that her winning statistics matures while Boris's winning statistics does not mature.

Thus one can simplify the second condition and show that it is equivalent to the following two games [57, 74]:

- One only maintains Anke's winning statistics and a play terminates with a win for Anke iff she is eventually declared a winner, and the play is a win for Boris iff it runs forever.
- One only maintains Boris's winning statistics and a play is a win for Anke iff it never happens that the winning statistics of Boris declare him a winner.

The first game is called a reachability game [57] and the second game a survival game [74, Chapter 9]. Both games are isomorphic, as they are obtained from each other only by switching the player who is supposed to win. Such types of reductions, though not with good complexity bounds, were also considered by Bernet, Janin, and Walukiewicz [3]. The reachability game to which one reduces the parity game can now be described as follows:

- The set Q of nodes of the reachability game consists of nodes of the form $(a, p, \tilde{b})$, where a is a node of the parity game, the player $p \in \{\text{Anke}, \text{Boris}\}$ moves next, and $\tilde{b}$ represents the winning statistics of Anke.
- The starting node is $(s, p, \tilde{0})$, where $\tilde{0}$ is the vector of all b_i with value 0, s is the starting node of the parity game, and p is the player who moves first.
- Anke can move from $(a, \text{Anke}, \tilde{b})$ to $(a', \text{Boris}, \tilde{b}')$ iff she can move from a to a' in the parity game and this move causes Anke's winning statistics to be updated from $\tilde{b}$ to $\tilde{b}'$ and $\tilde{b}$ does not yet indicate a win for Anke.
- Boris can move from $(a, \text{Boris}, \tilde{b})$ to $(a', \text{Anke}, \tilde{b}')$ iff he can move from a to a' in the parity game and this move causes Anke's winning statistics to be updated from $\tilde{b}$ to $\tilde{b}'$ and $\tilde{b}$ does not yet indicate a win for Anke.

The number of elements of Q can be bounded by $O(n^4)$. First note that the number of increasing functions from $\{0, 1, 2, \ldots, \lceil\log(n)\rceil + 2\}$ to $\{1, 2, 3, \ldots, \lceil\log(n)\rceil\}$ can be bounded by $O(n^2)$, as any such sequence $(b'_0, b'_1, b'_2 \ldots, b'_{\lceil\log(n)\rceil+2})$ can be represented by the subset $\{b'_k + k : 0 \leq k \leq \lceil\log(n)\rceil + 2\}$ of $\{1, 2, 3, \ldots, 2\lceil\log(n)\rceil + 2\}$ and that there are at most $O(n^2)$ such sets. Further, note that $b'_k \leq b'_{k+1}$ implies $b'_k + k < b'_{k+1} + k + 1$, and thus all b'_k can be reconstructed from the set. Given a winning statistics $\tilde{b} = (b_0, b_1, b_2, \ldots, b_{\lceil\log(n)\rceil+2})$, one defines $b'_0 = \max\{1, b_0\}$ and $b'_{k+1} = \max\{b'_k, b_{k+1}\}$ and notes that only those b_k with $b_k = 0$ differ from b'_k. Thus one needs at most $\lceil\log(n)\rceil + 3$ additional bits to indicate which b_k is 0. The overall winning statistics can then be represented by $3\lceil\log(n)\rceil + 5$ bits. Furthermore, one needs 1 bit to represent the player and $\lceil\log(n)\rceil$ bits to represent the current node in the play. Accordingly, each node in Q can be represented with $4\lceil\log(n)\rceil + 6$ bits, resulting in $O(n^4)$ nodes in Q. The set Q itself can be represented by using a set of such representations of nodes.

Note that one can compute the set Q of vertices and determine a list of nodes $Q' \subseteq Q$ where Anke's winning statistics indicate a win in time $O(|Q| \cdot n)$; the set Q'

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

is the set of target nodes in the reachability game.

Proposition 17 shows that the such constructed reachability game can be decided in time $O(|Q| \cdot n)$ by a well-known algorithm. For the general case of a reachability game, the time complexity is linear in the number of vertices plus number of edges of the game graph; note that the reachability game constructed has $|Q|$ nodes and at most $|Q| \cdot n$ edges. This completes the proof. ∎

The algorithm below is listed explicitly by Khaliq and Imran [56] and appeared much earlier in the literature, though sometimes in different or only related contexts [1, 22, 43, 45, 50]. The algorithm is now included for the reader's convenience.

PROPOSITION 17 (Beeri [1], Cook [22], Gurevich and Harrington [45], Immerman [50]). *In a reachability game with a set Q of nodes, a subset $Q' \subseteq Q$ of target nodes Q', out degree up to n per node, and start node s, one can decide in time $O(|Q| \cdot n)$ which player can win the game.*

Proof. One computes for each node $q \in Q$ a linked list of q's successors (which are at most n in number) and a linked list of q's predecessors. Note that the collection of all the successor and predecessor lists for different nodes in Q taken together has length at most $|Q| \cdot n$. These lists can also be generated in time $O(|Q| \cdot n)$.

Note that a node q is a winning node for Anke if $q \in Q'$ or either Anke moves from q and one successor node of q is a winning node for Anke or Boris moves from q and all successor nodes of q are winning nodes for Anke. This idea leads to the algorithm below.

Next, for each node q, a tracking number k_q is introduced and maintained such that the winning nodes for Anke will eventually all have $k_q = 0$, where k_q indicates how many further times one has to visit the node until it can be declared a winning node for Anke. The numbers k_q are initialized by the following rules:

- On nodes $q \in Q'$ the number k_q is 1.
- On nodes $q = (a, \text{Anke}, \tilde{b}) \notin Q'$, the number k_q is initialized as 1.
- On nodes $q = (a, \text{Boris}, \tilde{b}) \notin Q'$, the number k_q is initialized as the number of nodes q' such that Boris can move from q to q'.

These numbers can be computed from the length of the list of successors of q for each $q \in Q$. Now one calls the following recursive procedure, initially for all $q \in Q'$ such that each call updates the number k_q. The recursive call does the following:

- If $k_q = 0$, then return without any further action, else update $k_q = k_q - 1$.
- If after this update it still holds that $k_q > 0$, then return without further action.
- Otherwise, that is, when k_q originally was 1 when entering the call, recursively call all predecessors q' of q with the same recursive call.

After the termination of all these recursive calls, one looks at k_q for the start node q of the reachability game. If $k_q = 0$, then Anke wins, else Boris wins.

In the above algorithm, the predecessors of each node $q \in Q$ are called at most once from a call in q, namely, when k_q goes down from 1 to 0; furthermore, this is the time where it is determined that the node is a winning node for Anke. Thus there are at most $O(|Q| \cdot n)$ recursive calls and the overall complexity is $O(|Q| \cdot n)$.

For the verification, the main invariant is that, for nodes $q \in Q - Q'$, k_q indicates how many more successors of q one still has to find which are winning nodes for Anke until q can be declared a winning node for Anke. In the case that Anke's winning statistics has matured in the node q, the value k_q is taken to be 1 so that the node is processed once in all the recursive calls in the algorithm. For nodes where it is Anke's turn to move, only one outgoing move which produces a win for Anke is needed.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Consequently, one initializes k_q to 1, and as soon as this outgoing node is found, k_q goes to 0, which means that the node is declared a winning node for Anke. In case the node q is a node where Boris moves, then one has to enforce that Boris has no choice but to go to a winning node for Anke. Thus k_q is initialized to the number of moves which Boris can make in this node; each time one of these successor nodes is declared a winning node for Anke, k_q goes down by one. Observe that once the algorithm is completed, the nodes with $k_q = 0$ are exactly the winning nodes for Anke in the reachability game. ∎

The next result carries over the methods of Theorem 16 to the general case, that is, it uses everything except those parts which make use of $m \leq \log(n)$. So the size of the code representing a winning statistic for Anke is given by $\lceil \log(n) \rceil + 3 \leq \log(n) + 4$ numbers of $\lceil \log(m+1) \rceil \leq \log(m) + 1$ bits. As $\log(m) \leq \log(n)$, the overall size of representation of a node in the set Q of nodes of the reachability game can be bounded by $\log(n) \cdot (\log(m) + 5) + c$. Hence, the size of $|Q|$ is $O(n^{\log(m)+5})$ and the number of edges in the reachability game is $O(n^{\log(m)+6})$.

For many decision problems in **NP**, in particular for the **NP**-complete ones, one can find solutions witnessing the given answer (like the winning strategy for the winner of the parity game) by solving several variants of the decision problem where more and more parameters of the problem are fixed by constants [2]. This is now outlined for finding the memoryless winning strategy of the winner of a parity game using an algorithm which decides who is the winner. For ease of notation, assume that Anke can win the game on a graph (V, E). Now one does the following steps to retrieve the winning strategy:

1. Maintain, for each node $a \in V$, a list of possible successors V_a which is initialized as $\{b : (a, b) \in E\}$ at the beginning.
2. If there is no node $a \in V$ with, currently, $|V_a| > 1$, then one terminates, with a winning strategy for Anke in the parity game being to move from every node a to the unique node in V_a, else one selects a node $a \in V$ with $|V_a| > 1$.
3. Now one splits V_a into two nearly equal sized subsets V'_a and V''_a with $|V'_a| \leq |V''_a| \leq |V'_a| + 1$.
4. One replaces V_a by V'_a and permits, in the derived reachability game, moves from $(\tilde{a}, \text{Anke}, \tilde{b})$ to $(\tilde{a}', \text{Boris}, \tilde{b}')$ only when $\tilde{a}' \in V_{\tilde{a}}$ for all nodes $\tilde{a}$.
5. If Anke does not win this game, then one replaces $V_a = V''_a$, else one keeps $V_a = V'_a$.
6. Go to step 2.

The above algorithm works since whenever Anke has a winning strategy for the parity game, then there is a memoryless one, and therefore when splitting the options at node a, some memoryless winning strategy either always takes a node from V'_a or always takes a node from V''_a. It is straightforward to verify that the above loop runs $n \log(n)$ rounds and each round involves $O(|Q| \cdot n)$ time plus one solving of the reachability game, which can also be solved in time $O(|Q| \cdot n)$. Thus one can derive the following result.

THEOREM 18. *There is an algorithm which finds the winner of a parity game with n nodes and values from $\{1, 2, 3, \ldots, m\}$ in time $O(n^{\log(m)+6})$. Furthermore, the algorithm can compute a memoryless winning strategy for the winner in time $O(n^{\log(m)+7} \cdot \log(n))$.*

Thus, as shown, when $m \leq \log(n)$ the runtime is $O(n^5)$; if $m > \log(n)$, then $2^m > n$ and one can bound $n^{\log(m)+6}$ from above by $2^{m \cdot (\log(m)+6)}$. Thus one has the bound $O(n^5 + 2^{m \cdot (\log(m)+6)})$ for the runtime of solving a parity game with n nodes

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

and values from $\{1, 2, 3, \ldots, m\}$. In other words, parity games are fixed-parameter tractable for their main parameter m.

COROLLARY 19. *Parity games are in the class* **FPT** *and can be solved in time* $O(n^5 + 2^{m(\log(m)+6)})$.

Followup work obtained better bounds on the runtime by using that the translation into the reachability game provides a game with the number of edges bounded by

$$\binom{m + 2 \cdot (\lceil \log(n) \rceil + 3)}{\lceil \log(n) \rceil + 3} \cdot n^2.$$

The above formula led to the bound $O(2^m \cdot n^4)$ [14], which is based on the fact that $\binom{i}{j} \leq 2^i$ for all i, j. A further estimate can be obtained by slightly increasing the binomial upper bound to

$$\binom{(\lceil m/\log(n) \rceil + 2) \cdot (\lceil \log(n) \rceil + 3)}{\lceil \log(n) \rceil + 3} \cdot n^2$$

and then using common estimates on binomials, where the upper number is a multiple of the lower number. The calculations provide a runtime bound of

$$O(\lceil m/\log(n) \rceil^4 \cdot n^{3.45+\log(\lceil m/\log(n) \rceil+2)});$$

this and similar bounds of this type were obtained by several researchers [34, 42, 54, 74]. Subsequent improvements included replacing the term n^2 in the above formulas by the number of edges in the parity game [34, 42, 54].

The main improvement over the current algorithm by followup work is, however, the usage of space. The current algorithm uses quasi-polynomial time and quasi-polynomial space. Subsequent work has brought down this complexity from quasi-polynomial to quasi-linear [34, 54]; more precisely Jurdziński and Lazić have the space bound $O(n \cdot \log(n) \cdot \log(m))$ and Fearnley et al. [34] have the space bound $O(n \cdot \log(n) \cdot \log(m) + \ell \cdot \log\log(n))$, where ℓ is the number of edges in the parity game and thus $\ell \leq n^2$; the time bounds of both algorithms are approximately the same as those of the algorithm presented here, but due to the better space bound an additional overhead from managing large space can be avoided in an implementation.

Lehtinen [59] introduced the notion of the register index complexity of a parity game and showed that every parity game has register index complexity of at most $\log(n) + 1$. She then gave an algorithm to translate the given parity game of register index k into a usual parity game of size $O(m^k \cdot n)$ with $2k+1$ values on the edges. This game can then be solved in polynomial time (with respect to $m^k \cdot n$), as the number $2k + 1$ of values is bounded logarithmically in the number of nodes; furthermore, results prior to the current work would also have already shown that the translated game can be solved in quasi-polynomial time, and thus Lehtinen [59] has supplied a quasi-polynomial time algorithm for solving parity games which can be verified without making reference to the present work.

4. Parity games versus Muller games. Muller games are a well-studied topic [7, 8, 61, 76, 80] and had already been investigated as a general case before researchers aimed for the more specific parity games. A Muller game (V, E, s, G) consists of a directed graph (V, E), a starting node s, and a set $G \subseteq \{0, 1\}^V$. For every infinite play starting in s, one determines the set U of nodes visited infinitely often during the play: if $U \in G$, then Anke wins the play, else Boris wins the play. In a Muller

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

game the complement of G is closed under union iff for all $U, U' \notin G$ the set $(U \cup U')$ is not in G.

For complexity assumptions, it is natural to consider the case where G is not given as an explicit list, but as an algorithm, which is polynomial in size, which runs in polynomial time, and which computes the membership of a set U (given by its explicit list) in the set G or some similar equivalent effective representation. The reason for considering such a representation for G is that Horn [47] showed that if G is given as an explicit list of all possible sets of nodes infinitely visited when Anke wins, then the resulting game is solvable in polynomial time in the sum of the number of nodes and the number of explicitly listed sets. Hence, only more flexible ways of formulating winning conditions lead to interesting cases of Muller games.

For Muller games, Björklund, Sandberg, and Vorobyov [5] considered a parameter which is given by the number of colors. For this, they assign to every node a color from $\{1, 2, 3, \ldots, m\}$ and take G to be some set of subsets of $\{1, 2, 3, \ldots, m\}$. Then U is not the set of infinitely often visited nodes, but instead the set of colors of the infinitely often visited nodes. Again, if $U \in G$, then Anke wins the play, else Boris wins the play. Colored Muller games permit more compact representations of the winning conditions. In the worst case there is a 2^m-bit vector, where m is the number of colors; however, one also considers the case where this compressed winning condition is given in a more compact form, say, by a polynomial-sized algorithm or formula.

In the following, the interactions between Muller games, memoryless winning strategies, and parity games are presented. The first result is due to Emerson [30] and Zielonka [80, Corollary 11], and the second one is in Hunter's thesis [48].

THEOREM 20 (Emerson [30], Zielonka [80]). *Consider a Muller game (V, E, s, G) in which the complement of the set G of winning conditions is closed under union. If Anke has a winning strategy, then Anke also has a memoryless winning strategy.*

Proof. The possible choices for Anke at any node will be progressively constrained. The proof is by induction on the number of possible moves of Anke in the constrained game. The result holds when, for each node, Anke has only one movement choice. For the induction step, suppose some node v for Anke's move has more than one choice. It is now shown that for some fixed Anke move at node v, Anke has a winning strategy; thus one can constrain the move of Anke at node v, and by induction this case is done. Suppose, by way of contradiction, that for every Anke move w at v, Boris has a winning strategy S_w. This allows Boris to have a winning strategy for the whole game as follows.

Assume without loss of generality that the play starts with Anke's move at v. Intuitively, think of Boris playing several parallel plays against Anke (each play in which Anke moves w at node v for different w) which are interleaved. For ease of notation, consider the individual play with Anke using move w at node v as play H_w, and the interleaved full play as H.

Initially H and all the plays H_w are at the starting point. At any time in the play H, if it is Anke's move at v and Anke makes the move w', then Boris continues as if it is playing the play $H_{w'}$ (and suspends the previous play H_w if $w \neq w'$). Thus the nodes visited in H can be seen as the merger of the nodes visited in the plays H_w for each choice w of Anke at node v. This implies that the set of nodes visited infinitely often in H is equal to the union of the sets of nodes visited infinitely often in the various H_w. As Boris wins each play H_w which is played for infinitely many moves, by closure of the complement of G under union, Boris wins the play H. ∎

As a parity game is also a Muller game in which G is closed under union for both

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Anke and Boris, the following corollary holds.

COROLLARY 21 (Emerson and Jutla [32], Mostowski [62]). *The winners in parity games have memoryless winning strategies.*

Hunter [48, p. 23] showed the following characterization for Muller games. Note that McNaughton [61] also investigated Muller games with memoryless strategies and characterized them through the concept of splitting [61], which is just another way of stating that both G and its complement are union-closed. However, his paper does not connect these Muller games with parity games explicitly.

THEOREM 22 (Hunter [48]). *Every Muller game (V, E, s, G) in which both G and its complement are closed under the union operation is a parity game, and the translation can be done in polynomial time whenever the winning set G can be decided in polynomial time.*

Proof. In this proof a parity game isomorphic to the given Muller game will be constructed. In this parity game player Anke owns the nodes with even value and Boris owns the nodes with odd value. Given V, let

$$V_1 = \{a \in V : \{a\} \in G\} \text{ and } V_2 = \{b \in V : \{b\} \notin G\}.$$

Obviously V is the disjoint union of V_1 and V_2. By the closure under union, any subset $V' \subseteq V_1$ is in G and no subset $V' \subseteq V_2$ is in G.

To prove the theorem, values will be inductively assigned to the nodes one by one.

Suppose values have already been assigned to all nodes in $V - V'$, where V' is initially V. Then assign the value to one node in V' as follows. Let $V_1' = V' \cap V_1$ and $V_2' = V' \cap V_2$.

Case 1: *Suppose* $V' \in G$. Now, there is a node $a \in V_1'$ such that $\{a\} \cup V_2' \in G$, as otherwise $V' \notin G$ since the complement of G is closed under the union operation. Now let $V_1'' \subseteq V_1'$ and $V_2'' \subseteq V_2'$. The set $\{a\} \cup V_2''$ is in G, as otherwise $(\{a\} \cup V_2'') \cup V_2'$ is not in G, in contradiction to the choice of a. Furthermore, as $V_1'' \cup \{a\} \in G$, $(V_1'' \cup \{a\}) \cup (\{a\} \cup V_2'') = \{a\} \cup V_1'' \cup V_2''$ is in G. Thus whenever $V'' \subseteq V'$ and $a \in V''$, $V'' \in G$. Hence, the value $2|V'|$ is assigned to a accordingly.

Case 2: *Suppose* $V' \notin G$. Then there exists a node $b \in V_2'$ such that $\{b\} \cup V_1' \notin G$, by reasons similar to those given in Case 1. Note that this implies that whenever $V'' \subseteq V'$ and $b \in V''$, then $V'' \notin G$. Hence, the value $2|V'| + 1$ is assigned to b.

The above process of assigning values to nodes is clearly consistent, since for $V'' \subseteq V'$ being the set of infinitely visited nodes, in Case 1 if a is in V'', then Anke wins, and in Case 2 if b is in V'', then Boris wins. It follows that this Muller game is a parity game. ∎

Besides the standard colored Muller game of Björklund, Sandberg, and Vorobyov [5], one can also consider the memoryless colored Muller game. These are considered in order to see whether the game is easier to solve if one permits Anke only to win when she follows a memoryless strategy; otherwise she loses by the rules of the game. The main finding comparing memoryless colored Muller games with standard colored Muller games is as follows: On one hand, memoryless colored Muller games are easier in terms of the best known complexity class to which they belong: memoryless colored Muller games are in $\mathbf{\Sigma}_2^P$, while the decision complexity of standard colored Muller games is in **PSPACE**. On the other hand, the time complexity of memoryless colored Muller games is worse, as one cannot exploit a small number of colors to bring the problem into P; already four colors make it **NP**-hard to find the winner in memoryless

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

colored Muller games; see Theorem 27.

Björklund, Sandberg, and Vorobyov [5] proved that the colored Muller game is fixed parameter tractable iff the parity game is fixed parameter tractable (with respect to the number of values m of the parity game). It follows from Theorem 16 that also the colored Muller game is fixed parameter tractable. More precisely, McNaughton [61] and Björklund, Sandberg, and Vorobyov [5] showed the following result.

THEOREM 23 (Björklund, Sandberg, and Vorobyov [5], McNaughton [61]). *One can translate a colored Muller game with m colors and n nodes in time polynomial in $m! \cdot n$ into an equivalent parity game with $2m$ colors and $m! \cdot n$ nodes.*

Proof. In this proof, one considers Muller games with nodes possibly having multiple colors. The idea is based on the last appearance record of the colors.

Each node v from the original game will be replaced by all nodes of the form (v, r) in the new game, where r denotes an ordered list of colors as to how recently they were observed in the nodes visited before the current node.

One lets Anke have the odd and Boris the even numbers. The value of the node (v, r) is computed in two steps. First one computes the set U of colors in r which are at least as recent as one of the colors of v in the Muller game; that is, U is the set of colors whose position might be affected by an update of r when leaving the current node for the next node. For example, if the game has four colors which were observed in the order (c_1, c_2, c_3, c_4) (c_1 is the most recent color) and if the node v in the Muller game carries the colors c_2 and c_3, then $U = \{c_1, c_2, c_3\}$, and when passing to the next node r will be updated to $r' = (c_2, c_3, c_1, c_4)$. Second, one lets the value of the node (v, r) be $2 \cdot |U| + 1$ in the case that U is a winning set for Anke in the Muller game and $2 \cdot |U| + 2$ in the case that U is a winning set for Boris in the Muller game.

If a player can move from v to w in the original Muller game, then the player can now move from (v, r) to (w, r') in the constructed parity game where r' is obtained from r by moving all the colors belonging to v to the front, as they are most recent when arriving in w, and by keeping the other colors in their order behind the new recent colors; other moves than those derived ones are not possible. Furthermore, when s is the starting node in the original colored Muller game, then the new starting node in the parity game is of the form (s, r) for some arbitrary but fixed record r.

Given now a play $(v_0, r_0), (v_1, r_1), (v_2, r_2), \ldots$ in the parity game, it defines a play $v_0, v_1, v_2, \ldots$ in the original Muller game and a set U which consists of the colors of the infinitely often visited nodes. For almost all k, these colors in U are in the front of the last appearance record r_k. As each of them is occurring infinitely often, there are infinitely many nodes (v_k, r_k) in the run where one of the colors of v_k is the last member of U in the current record r_k. It follows that U is the set of selected colors for (v_k, r_k) and the node (v_k, r_k) has Anke's parity iff U is a winning set for Anke. Furthermore, only the nodes where all colors of U are taken into account have the maximal value of the run. For that reason, Anke wins the run in the parity game iff she wins the corresponding run in the original Muller game.

Assume now that Anke has a winning strategy for the parity game. Then, when playing the original Muller game, in her memory Anke can keep track of the appearance record r_k for the current node v_k and then, in the case that it is her turn, move to that v_{k+1} such that in the parity game she would have made a move to a node of the form (v_{k+1}, r_{k+1}). As it is a winning strategy, the derived play in the parity game would be winning for Anke and thus also winning in the original play in the Muller game. The situation when Boris has a winning strategy for the parity game is similar, as he can then translate by the same method his winning strategy into one for the

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

colored Muller game. Thus the winner of the original Muller game is the same as the winner of the translated parity game; that is, the original game is equivalent to the translated game.

The bound on the number of nodes is $n \cdot m!$; the number of values in the game is $2m+2$ in the case that one allows nodes without colors so that the set U of the colors of the infinitely often visited nodes can be empty. It is $2m$ if every node needs to have at least one color, as then one can cut out the case of no color and would assign to the set U computed for a node (v, r) a value of either $2|U|-1$ or $2|U|$, depending on the parity of the player who wins when U is the set of colors of the infinitely often visited nodes. ∎

Now one uses this result in order to prove the bounds on the algorithm to solve the colored Muller games. Note that $\log(m! \cdot n) \geq 2m$ for all $m \geq 24$ and $n \geq m$: $\log(m!) \geq \log(8^{m-8}) \geq 3 \cdot (m-8) = 3m - 24$. For $m \geq 24$, $3m - 24 \geq 2m$. Thus, the remaining cases can be reduced to finite ones by observing that for all m and $n \geq \max\{m, 2^{48}\}$, $\log(m! \cdot n) \geq 2m$. So, for almost all pairs of (m, n), $\log(m! \cdot n) \geq 2m$, and therefore one can use the polynomial time algorithm of Theorem 16 to get the following explicit bounds.

THEOREM 24. *One can decide in time $O(m^{5m} \cdot n^5)$ which player has a winning strategy in a colored Muller game with m colors and n nodes.*

For the special case of $m = \log(n)$, the corresponding number of nodes in the translated parity game is approximately $n^{\log(\log(n))+2}$ and the polynomial time algorithm of Theorem 16 becomes an $O(n^{5 \log \log(n)+10})$ algorithm. The algorithm is good for this special case, but the problem is in general hard and the algorithm is slow.

One might ask whether this bound can be improved. Björklund, Sandberg, and Vorobyov [5] showed that under the Exponential Time Hypothesis it is impossible to improve the above algorithm to $2^{o(m)} \cdot \mathrm{Poly}(n)$; see Definition 6 above for the Exponential Time Hypothesis. The following result enables us to get a slightly better lower bound.

THEOREM 25. *A Muller game with m colors and n nodes and $1 \leq m \leq n$ cannot be solved in time $2^{o(m \cdot \log(m))} \cdot \mathrm{Poly}(n)$, provided that the Exponential Time Hypothesis is true.*

Proof. Note that for this result, multiple colors per node are allowed. However, one can translate a colored Muller game with multiple colors per node into one with one color per node and $m' = m+1$ colors and $n' = n \cdot m$ nodes. As it is required that $m \leq n$, the expressions $2^{o(m \cdot \log(m))} \cdot \mathrm{Poly}(n)$ and $2^{o(m' \cdot \log(m'))} \cdot \mathrm{Poly}(n')$ contain the same runtimes of algorithms.

Theorem 30 provides as a special case a translation of k-dimensional parity games with n nodes and 3 values per dimension into colored Muller games with n nodes and $m = 2k$ colors without changing the winner; the underlying game is not changed, but the way the plays are evaluated by the auxiliary structure of multidimensional parities is replaced by colors for the nodes. Furthermore, Theorem 31 shows that if a k-dimensional parity game with 3 values per dimension can be solved in time $2^{o(k \cdot \log(k))} \cdot \mathrm{Poly}(n)$, then the Exponential Time Hypothesis would fail. The proof of the current theorem then follows from the fact that if $m = 2k$, then $2^{o(k \cdot \log(k))} = 2^{o(m \cdot \log(m))}$, which is based on the equations $o(m \cdot \log(m)) = o(2k \cdot \log(2k)) = o(k \cdot \log(2k)) = o(k \cdot \log(k) + k \cdot 2) = o(k \cdot \log(k))$. This completes the proof. ∎

Memoryless games are games where Anke wins iff she (a) plays a memoryless strategy, and (b) wins the game according to the specification of the game. If she does

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

not do (a), this is counted as a loss for her. This was already defined by Björklund, Sandberg, and Vorobyov [5, section 5] for Streett games, and it can also be defined for Muller games.

The complexity of the memoryless games differs from those of normal games. Björklund, Sandberg, and Vorobyov [5, section 5] considered memoryless Streett games (called Quasi-Streett games in their paper) and showed that these are $\mathbf{W[1]}$-hard in a suitable parameterization.

The next theorem establishes the complexity of finding memoryless strategies for player Anke for Muller games. For this one needs some effective way of representing the winning conditions on the colors, and here it is assumed that they are given by a Boolean formula or circuit of size polynomial in the game (one has to fix such a polynomial, and any polynomial which is at least cubic in the number of colors would be sufficient for the hardness). The hardness part in (b) slightly extends what is known in the literature.

Dawar, Horn, and Hunter [24] extended a conference publication of Horn [46] in which it is shown that Muller games, where the winning condition is given as an explicit list of all sets of infinitely often visited nodes which are winning, is decidable in polynomial time; here the polynomial time algorithm, for input size, also takes into account the length of the explicit list. Dziembowski, Jurdziński, and Walukiewicz [29] investigated mainly the space complexity needed to implement strategies and provided some applications towards the complexity of solving the problem. Zielonka [80] used similar methods to show **NP**-hardness of the Muller games, even in the special case of games where player Anke, in case she wins, also has a memoryless winning strategy.

THEOREM 26 (see also Dawar, Horn, and Hunter [24], Dziembowski, Jurdziński, and Walukiewicz [29], Horn [46], Zielonka [80]).

(a) *The problem of whether Anke can win a memoryless colored Muller game is* $\mathbf{\Sigma_2^P}$*-complete.*

(b) *Suppose A is a polynomial time computable set of instances of formulas $F(x_1, \dots, x_i, y_1, \dots, y_j)$ in conjunctive normal form with two types of variables which satisfy that for each choice of $(x_1, \dots, x_i)$ there is at most one choice of $(y_1, \dots, y_j)$ which makes $F(x_1, \dots, x_i, y_1, \dots, y_j)$ true. Let B be the set of all such formulas F for which the statement $(*)$ given as*

$$\exists x_1 \dots \exists x_i \, \forall y_1 \dots \forall y_j \, [F(x_1, \dots, x_i, y_1, \dots, y_j) \text{ is not satisfied}]$$

is true. Then there is a polynomial time many-one reduction from $A \cap B$ to the set of all colored Muller games in which the winning conditions of Boris are closed under union such that $F \in A \cap B$ iff Anke is the winner of the game constructed for F. Furthermore, the problem of whether Anke can win such a game is in $\mathbf{\Sigma_2^P}$.

Proof. First, to see the membership in $\mathbf{\Sigma_2^P}$, consider the following well-known method: One guesses the memoryless winning strategy of Anke and then fixes Anke's moves to be always based on this strategy. This basically results in a one-player game where Boris always moves and successors of a node are not the original ones, but those which can be reached if in the original graph one first follows one step of Anke's strategy to a neighbor and then considers all moves of Boris from that neighbor. In this new graph, only Boris is moving, so it is effectively a one-player-game. Now Boris can only win this new game iff there is the corresponding periodic path which leads to Boris's win. That is, one guesses a path of up to length n from the starting node to this period as well as the periodic part of the path and verifies that the periodic part produces a set of colors on which Boris wins. Here, a period is not longer than the number n of nodes times the number of colors. Thus, if such a path does not exist,

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

then Anke has a winning strategy, and this verification is in **coNP**; hence the overall complexity is in $\mathbf{\Sigma_2^P}$.

The set of formulas F which satisfy $(*)$ is in general $\mathbf{\Sigma_2^P}$-complete. However, in the case of (b) one will enforce a promise, that is, take only those formulas which are members of a certain polynomial time computable set A satisfying the promise from the statement of the theorem; this makes the set $A \cap B$ incomplete for $\mathbf{\Sigma_2^P}$.

To show hardness, one reduces in both cases (a) and (b), the given formulas of the form $F(x_1, \ldots, x_i, y_1, \ldots, y_j)$ to Muller games. First one adds additional variables $\tilde{x}_1, \ldots, \tilde{x}_i$ and modifies the formula $(*)$ to the following formula (@):

$$\exists x_1 \ldots \exists x_i \forall \tilde{x}_1 \ldots \forall \tilde{x}_i \forall y_1 \ldots \forall y_j \, [x_1 \neq \tilde{x}_1 \vee \cdots \vee x_i \neq \tilde{x}_i \vee F(\tilde{x}_1, \ldots, \tilde{x}_i, y_1, \ldots, y_j) \text{ is not satisfied}\,].$$

The intuition behind the reduction is that Anke chooses the truth values $x_1, \ldots, x_i$ and copies them to $\tilde{x}_1, \ldots, \tilde{x}_i$. Boris is then responsible for finding a satisfying assignment, and this assignment is valid iff it does not produce any inconsistencies in the variables $\tilde{x}_1, \ldots, \tilde{x}_i, y_1, \ldots, y_j$. This will make it easier to detect which player is responsible for an inconsistent situation in the game, and the evaluation of a winner of a play takes this into account.

Formally, for the reduction from a formula $F(x_1, \ldots, x_i, y_1, \ldots, y_j)$, having m clauses, where the rth clause has n_r literals, the Muller game constructed is the following. The colors used by the game are of the form $\mathrm{pos}(x_h)$, $\mathrm{pos}(\tilde{x}_h)$, $\mathrm{neg}(x_h)$, $\mathrm{neg}(\tilde{x}_h)$, $\mathrm{pos}(y_h), \mathrm{neg}(y_h)$.

(a) Vertices: $\{E_h, P_h, N_h : 1 \leq h \leq i\}$.
Colors on P_h are $\mathrm{pos}(x_h)$ and $\mathrm{pos}(\tilde{x}_h)$. Colors on N_h are $\mathrm{neg}(x_h)$ and $\mathrm{neg}(\tilde{x}_h)$.
There is no color on E_h.
E_1 is the starting node, where Anke starts the play.

(b) Vertices: $\{C_h, X_h^r : 1 \leq h \leq m, 1 \leq r \leq n_h\}$, where m is the number of clauses in F and n_h is the number of literals in the hth clause of F.
No color on C_h.
If the rth literal in the hth clause of F is x_k (respectively, $\neg x_k$), then the color on X_h^r is $\mathrm{pos}(\tilde{x}_k)$ (respectively, $\mathrm{neg}(\tilde{x}_k)$).
If the rth literal in the hth clause is y_k (respectively, $\neg y_k$), then the color on X_h^r is $\mathrm{pos}(y_k)$ (respectively, $\mathrm{neg}(y_k)$).

(c) There are two dummy nodes Z_1, Z_2 with no colors.

(d) There is an edge from E_h to P_h and N_h if $1 \leq h \leq i$.
There is an edge from each of P_h, N_h to E_{h+1} if $1 \leq h < i$.
There is an edge from each of P_i and N_i to Z_1.
There is an edge from Z_1 to C_1.
There is an edge from C_h to X_h^r if $1 \leq h \leq m$ and $1 \leq r \leq n_h$.
There is an edge from X_h^r to C_{h+1} if $1 \leq h < m$ and $1 \leq r \leq n_h$.
There is an edge from X_m^r to Z_2 if $1 \leq r \leq n_m$.
There is an edge from Z_2 to E_1.

(e) Winning condition for Boris: For a set U of colors of the infinitely often visited nodes of a play, Boris wins if either there is a $z \in \{x_1, \ldots, x_i\}$ where both $\mathrm{pos}(z), \mathrm{neg}(z)$ are in U or there is no $z \in \{\tilde{x}_1, \ldots, \tilde{x}_i, y_1, \ldots, y_j\}$ where both $\mathrm{pos}(z), \mathrm{neg}(z)$ are in U. In other words, Anke wins iff $\{z : \mathrm{pos}(z) \in U \wedge \mathrm{neg}(z) \in U\}$ is a nonempty subset of $\{\tilde{x}_1, \ldots, \tilde{x}_i, y_1, \ldots, y_j\}$.

Intuitively, the Muller game graph consists of a list of subunits (E_h, P_h, N_h), where each subunit consists of Anke choosing an option to assign the truth value to x_h and $\tilde{x}_h$ ($\mathrm{pos}(x_h)$ denotes that x_h, and thus $\tilde{x}_h$, is true; $\mathrm{neg}(x_h)$ denotes that x_h, and thus

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

$\tilde{x}_h$, is false). After each subunit, the corresponding nodes P_h, N_h lead to the entry node E_{h+1} of the next subunit, except for the last subunit P_i, N_i, where (through a dummy node Z_1) it leads to the clauses. There are subunits (C_h, X_h^r) for each clause in F, and Boris has to choose between nodes representing the literals with the corresponding colors. So if the clause is $\tilde{x}_3 \vee y_1 \vee \mathrm{neg}(y_5)$, then Boris can move from C_h into one of three nodes with colors $\{\mathrm{pos}(\tilde{x}_3)\}$, $\{\mathrm{pos}(y_1)\}$, and $\{\mathrm{neg}(y_5)\}$ based on which literal Boris takes to be true. Each clause leads to the subunit of the next clause, except for the last mth clause, which, via a dummy node Z_2, leads back to the start node E_1. Note that every time in E_h it is Anke's turn to move, and in C_h it is Boris's turn to move.

Now given a set U of colors of the infinitely often visited nodes of a play, the winning condition for Boris is that either there is a $z \in \{x_1, \ldots, x_i\}$ where both $\mathrm{pos}(z)$, $\mathrm{neg}(z)$ are in U or there is no $z \in \{\tilde{x}_1, \ldots, \tilde{x}_i, y_1, \ldots, y_j\}$ where both $\mathrm{pos}(z)$, $\mathrm{neg}(z)$ are in U. In other words, Anke wins iff $\{z : \mathrm{pos}(z) \in U \wedge \mathrm{neg}(z) \in U\}$ is a nonempty subset of $\{\tilde{x}_1, \ldots, \tilde{x}_i, y_1, \ldots, y_j\}$.

For the set of colors U on the infinitely often visited nodes in a play, if the condition on U is winning for Boris, then either Anke has played inconsistently (that is, she has made two different choices of $x_1, x_2, \ldots, x_i$), as witnessed by the colors $\{\mathrm{pos}(z), \mathrm{neg}(z)\}$ for some $z \in \{x_1, \ldots, x_i\}$, or Boris has played in a way that all variables are always instantiated the same way in the literals selected by Boris to witness the veracity of the clauses; furthermore, those z which are in $\{\tilde{x}_1, \ldots, \tilde{x}_i\}$ coincide with Anke's choice. Thus U witnesses that the formula F can be satisfied with Anke's choice of $x_1, \ldots, x_i$. Therefore, if Boris has a winning strategy, then all choices of $(x_1, \ldots, x_i)$ can be extended to a satisfying assignment for F.

Note that Anke can win playing consistently whenever there exists $(x_1, \ldots, x_i)$ witnessing that $F \in B$; indeed she can only win when she plays memorylessly. On the other hand, if each choice of $(x_1, \ldots, x_i)$ can be extended to a satisfying assignment for F, then whatever Anke does, Boris can win the game: If Anke plays inconsistently, she loses; if Anke commits to some choice for $(x_1, \ldots, x_i)$ and always moves accordingly, then Boris can also always choose the literal witnessing of the truth of clauses and the resulting colors do not give an inconsistent choice for any variable; those variables with neither $\mathrm{pos}(z)$ nor $\mathrm{neg}(z)$ appearing in the colors of X_h^r are not relevant for making the formula F true and can be ignored.

The argument above directly proves the result (a), and therefore the problem whether Anke can win a memoryless colored Muller game is $\mathbf{\Sigma_2^P}$-complete.

For (b), assume that A and B are as in the theorem. As the nonmembers of A can be detected in polynomial time, without loss of generality, for the following analysis it is always assumed that the formulas F are from A. Furthermore, as above, Anke wins the constructed parity game iff the modified F satisfies (@) iff F satisfies $(*)$. Thus one only has to prove that the winning condition for Boris is closed under union when the promise is satisfied.

Thus consider two sets of colors V, W where Boris wins and let $U = V \cup W$. If there is a $z \in \{x_1, \ldots, x_i\}$ such that both $\mathrm{pos}(z), \mathrm{neg}(z) \in U$, then Boris wins. If such a z does not exist, then U and thus V, W encode a fixed choice of the truth values of $\{x_1, \ldots, x_i\}$. By the winning condition on the game, the variables $\{\tilde{x}_1, \ldots, \tilde{x}_i, y_1, \ldots, y_j\}$ all have at most one truth assignment in the colors for both V and W, as otherwise Boris would lose. Due to the promise of F, this truth assignment depends uniquely on the choice of the truth values of $\{x_1, \ldots, x_i\}$ and is thus the same for both V and W; furthermore, both V and W have, for every $z \in \{y_1, \ldots, y_j\}$, at least one of the colors $\mathrm{pos}(z), \mathrm{neg}(z)$, as otherwise there would be at least two sat-

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

isfying assignments (as no value of z is enforced). Thus the union U equals both V and W; it follows that U is a set of colors which is winning for Boris. So the winning conditions of Boris are closed under union. ∎

Note that one can reduce sets in **NP** ∪ **coUP** to sets A (with corresponding B) satisfying the promise condition in part (b). To see this, consider sets in **NP** of the form $X = \{z : (\exists x_1, x_2, \ldots, x_i)[G(z, x_1, x_2, \ldots, x_i)]\}$, where $G(z, x_1, \ldots, x_i)$ can be solved in deterministic polynomial time. Then, for each z, one can construct a formula in conjunctive normal form $F_z(x_1, x_2, \ldots, x_i, y_1, \ldots, y_j)$ such that $F_z(x_1, x_2, \ldots, x_i, y_1, \ldots, y_j)$ is true iff $y_1, \ldots, y_j$ codes the deterministic computation of $G(z, x_1, \ldots, x_i)$ and $G(z, x_1, \ldots, x_i)$ is false. Here A would be the set of all formulas F_z. As there is only one deterministic computation of $G(z, x_1, \ldots, x_i)$, for each $x_1, \ldots, x_i$, there is at most one satisfying assignment for $F_z(x_1, x_2, \ldots, x_i)$. Furthermore, if $z \in X$, then for some appropriate choice of $x_1, x_2, \ldots, x_i$, $G(z, x_1, x_2, \ldots, x_i)$ is true, and thus $F_z(x_1, x_2, \ldots, x_i, y_1, \ldots, y_j)$ is not satisfied for at least one possible value of $y_1, \ldots, y_j$ (the one which codes the deterministic computation of $G(z, x_1, x_2, \ldots, x_i)$). In the case $z \notin X$, for all $x_1, x_2, \ldots, x_i$, $G(z, x_1, x_2, \ldots, x_i)$ is false, and thus, for all $x_1, x_2, \ldots, x_i$, for $y_1, \ldots, y_j$ coding the deterministic computation of $G(z, x_1, \ldots, x_i)$, $F_z(x_1, x_2, \ldots, x_i, y_1, \ldots, y_j)$ is satisfiable. Thus, the requirements as in part (b) are satisfied.

Similar reductions can be done for problems X in **coUP** by using $i = 0$ (and thus no x_i's are used) and using $y_1, \ldots, y_j$ to code the computations of the **UP** machine. This would give that F_z satisfies $(*)$ iff $z \in X$.

The result that memoryless colored Muller games can be solved in $\mathbf{\Sigma_2^P}$ stands in contrast to the fact that Dawar, Horn, and Hunter [24] showed that deciding the winner of a Muller game is a **PSPACE**-complete problem.

The next result shows that unless **NP** can be solved in quasi-polynomial time there is no analogue of the translation of Björklund, Sandberg, and Vorobyov [5] from memoryless colored Muller games into parity games. In contrast, solving memoryless colored Muller games with four colors is already **NP**-complete and thus solving memoryless colored Muller games is not in **XP**, unless **P** = **NP**.

THEOREM 27. *Solving memoryless colored Muller games with four colors is* **NP***-complete.*

Proof. For seeing that the game is in **NP**, one guesses the strategy and translates the original game into a new colored Muller game with $2n$ nodes: (i) each original node v is represented by two nodes (Anke, v) and (Boris, v) in the new game, (ii) the unique edge from (Anke, v) to (Boris, w) is picked as given by the memoryless winning strategy, and (iii) the move from (Boris, v) to (Anke, w) is there iff there is an edge from v to w in the original Muller game. By Theorem 23 mentioned above, one can first translate this intermediate colored Muller game into a parity game with 8 values and $24 \cdot n$ nodes [5, 61] and then solve the parity game in polynomial time $O(n^5)$, as $\log(24 \cdot n) \geq 8$ whenever $n \geq 11$.

For **NP**-hardness, **SAT** is reduced to memoryless colored Muller game as follows. For ease of writing the proof, Muller games where nodes determine the player moving are considered. This could be easily converted to a game where the moves of Anke and Boris alternate by inserting intermediate nodes if needed.

Suppose $x_1, x_2, x_3, \ldots, x_k$ are the variables and $y_1, y_2, y_3, \ldots, y_h$ are the clauses in a **SAT** instance. Without loss of generality assume that no variable appears both as a positive and a negative literal in the same clause. Then the above **SAT** instance is reduced to the following Muller game (where the graph is an undirected graph and

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

we interpret any edge as having both directions):

1. $V = \{s\} \cup \{u_1, u_2, u_3, \ldots, u_k\} \cup \{v_1, v_2, v_3, \ldots, v_h\} \cup \{w_{i,j} : [1 \leq i \leq h]$ and $[1 \leq j \leq k]$ and $[x_j$ or $\neg x_j$ appears in the clause $y_i]\}$.
   Boris moves at nodes s and u_j with $1 \leq j \leq k$. Anke moves at all other nodes.
2. $E = \{(v_i, w_{i,j}), (w_{i,j}, u_j), (w_{i,j}, v_i), (u_j, w_{i,j}) : x_j$ or $\neg x_j$ appears in $y_i\} \cup \{(s, u_j), (u_j, s) : 1 \leq j \leq k\}$.
3. The colors are $\{x, y, +, -\}$; s has the color y; all nodes u_j have the color x; all nodes v_i have the color y; for every node $w_{i,j}$ in the graph, if x_j appears in the clause y_i positively, then the color is $+$, else $\neg x_j$ appears in y_i and the color is $-$.
4. The winning sets for Boris are $\{x, +, -\}$ and all subsets of $\{y, +, -\}$; the winning sets for Anke are $\{x, +\}$, $\{x, -\}$, $\{x\}$, and all supersets of $\{x, y\}$.

Now it is shown that the **SAT** instance is satisfiable iff the Muller game is a win for Anke playing in a memoryless way.

Suppose the instance is satisfiable. Then fix a satisfying assignment $f(x_j)$ for the variables, and let $g(y_i) = j$ such that x_j (or $\neg x_j$) makes the clause y_i true. Now Anke has the following winning strategy: At node v_i, move to $w_{i,g(y_i)}$. At node $w_{i,j}$, if $g(y_i) = j$, then move to u_j, else move to v_i. Intuitively, at nodes v_i, Anke directs the play to the node $u_{g(y_i)}$ (via $w_{i,g(y_i)}$). Similarly, for the nodes $w_{i,j}$, Anke directs the play to $u_{g(y_i)}$ either directly or via nodes v_i and $w_{i,g(y_i)}$.

Thus, clearly, if an infinite play goes through color y infinitely often, then it also goes through color x infinitely often; thus Anke wins. On the other hand, if an infinite play does not go through color y infinitely often, then the set of nodes the play goes through infinitely often is, for some fixed j, u_j and some of the nodes of the form $w_{i,j}$. But then, by the definition of Anke's strategy, the play can only go through nodes of color $-$ finitely often (if $f(x_j)$ is true) and through nodes of color $+$ finitely often (if $f(x_j)$ is false). Thus, Anke wins the play.

Now suppose Anke has a winning strategy. If there is an i such that Anke moves from $w_{i,j}$ to u_j, then do the following: If x_j appears positively in the clause, then let $f(x_j)$ be true, else let $f(x_j)$ be false. If there is no i such that Anke moves from $w_{i,j}$ to u_j, then the truth value of $f(x_j)$ does not matter (and can be assigned either true of false).

To see that the above defines a satisfying assignment, first note that for each clause y_i, there exists a $w_{i,j}$ such that Anke moves from $w_{i,j}$ to u_j. Otherwise, Boris can first move from the start node to u_j and then to $w_{i,j}$ such that x_j appears in clause y_i; afterwards the play will go infinitely often only through a subset of the nodes of the form $v_i, w_{i,j}$, and thus the colors which appear infinitely often in the above play are a subset of $\{y, +, -\}$.

Furthermore, for no j and two nodes $w_{i,j}$ and $w_{i',j}$ such that x_j appears in y_i and $\neg x_j$ appears in $y_{i'}$, does Anke move from $w_{i,j}$ and $w_{i',j}$ to node u_j. Otherwise, Boris could win by first moving from s to u_j and then alternately going to nodes $w_{i,j}$ and $w_{i',j}$. It follows that f gives a satisfying assignment for the given **SAT** instance. ∎

5. Multidimensional parity games. Point [67] considered a generalization of parity games where each node has a vector of k values and each value is a number from 1 to m. To evaluate a play, one determines for each coordinate of the vector the largest infinitely often occurring value in the play and calls the so obtained vector of k values the limit superior of the sequence of the play. The same idea has recently also been applied to mean payoff games, Rabin and Streett games, as well as combinations of these games with parity games [10, 16, 17, 19, 20, 77]. The winner of a play is determined as follows: If all values of the limit superior vector are odd, then Anke

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

wins the play, else Boris wins the play. The approach in which the first player Anke has a conjunction and the second player Boris a disjunction of the player's winning conditions in each dimension is quite common in the field [16, 19, 20, 77]. In this section, it is assumed that $n \geq 2$, $m \geq 2$, and $k \geq 2$.

Rabin games and Streett games are games where the winner of a play is determined by a list of pairs of sets of nodes $(V_1, W_1), (V_2, W_2), (V_3, W_3), \ldots, (V_m, W_m)$. Now, in the Rabin case, Anke wins a play iff there is an i such that the set of infinitely often visited nodes U intersects V_i and is disjoint to W_i; in the Streett case, Anke wins a play iff all i satisfy that U intersects W_i or U is disjoint to V_i.

PROPOSITION 28 (Chatterjee, Henzinger, and Piterman [17]). *One can translate k-dimensional parity games with values from $\{1, 2, 3, \ldots, m\}$ in each dimension into Streett games with $k \cdot \lceil (m-1)/2 \rceil$ pairs and Streett games with k pairs into k-dimensional parity games with values from $\{1, 2, 3\}$.*

Proof. Both directions do not change the graph of the game; they only replace the value vectors by conditions in the Streett pair, and vice versa. Recall that each Streett pair is a pair (V, W) of two subsets of the set of nodes, and a winning play for Anke satisfies the pair if whenever a node in V is infinitely often visited, then also some node in W is infinitely often visited.

For the direction from k-dimensional parity games to Streett games, one generates for every even value $i \in \{1, 2, 3, \ldots, m\}$ and every dimension $j \in \{1, 2, 3, \ldots, k\}$ a pair (V, W), where V consists of all nodes where the jth component of the value vector is i, and W consists of all nodes where the jth component of the value vector is strictly larger than i. Now the limit superior of the values in each dimension of the given play is odd iff the play of the game satisfies all these Streett pairs.

For the direction from a game with k Streett pairs to the k-dimensional parity game, one assigns to the hth Streett pair (V, W) the hth dimension where every node outside $V \cup W$ has the hth value 1, every node in $V - W$ has the hth value 2, and every node in W has the hth value 3. ∎

The following corollary is due to previously known results on Streett games like the **coNP**-completeness by Emerson and Jutla [31]; note that Chatterjee, Henzinger, and Piterman [17] showed that the **coNP**-hardness part can even be achieved when only considering two-dimensional parity games.

COROLLARY 29. *If Boris has a winning strategy for a multidimensional parity game, then he has a memoryless winning strategy. Furthermore, the problem of whether Anke can win a multidimensional parity game is* **coNP***-complete.*

The following result provides an algorithm with runtime $O((2^{k \cdot \log(k) \cdot m} \cdot n)^{5.45})$ for multidimensional parity games which translates into a bound of $(2^{3 \cdot k \cdot \log(k)} \cdot n)^5$ for solving Streett games and Rabin games with n nodes and k conditions, where $k \geq 4$. For a comparison, a direct solution without translating into other games by Piterman and Pnueli [66] has a runtime $O(n^{k+1} \cdot k!)$.

THEOREM 30. *The winner of a multidimensional parity game with k values from $\{1, 2, 3, \ldots, m\}$ per node and n nodes can be determined in time $O((2^{k \cdot \log(k) \cdot m} \cdot n)^{5.45})$. If $k \geq 4$, then the formula can be improved to $O((2^{k \cdot \log(k) \cdot m} \cdot n)^5)$.*

Proof. The algorithm is based on ideas of Point [67] and also later by Chatterjee, Henzinger, and Piterman [17], who observed that the algorithm of Björklund, Sandberg, and Vorobyov [5] for translating Muller games into parity games can be adjusted to translate multidimensional parity games into normal parity games. The idea is to

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

use colors $c_{m',k'}$ with $m' \in \{2,3,4,\ldots,m\}$ and $k' \in \{1,2,3,\ldots,k\}$. Now, a node has a color $c_{m',k'}$ iff its value vector $(\tilde{m}_1, \tilde{m}_2, \tilde{m}_3, \ldots, \tilde{m}_k)$ satisfies that $m' \leq \tilde{m}_{k'}$ (note that a node may have multiple colors). Note that it is not needed to use $c_{1,k'}$ as always $1 \leq \tilde{m}_{k'}$, and therefore the color $c_{1,k'}$ would not carry any information. Now one tweaks the translation of the last appearance records in Theorem 23. Recall from the proof of Theorem 23 that the translation was realized by mapping each node v to a collection of nodes (v,r), where r is the record of colors in the order of their last appearance in prior visited nodes; those never visited can be in any order at the end of r. As every node which contains a color $c_{m',k'}$ also contains all colors $c_{m'',k'}$ with $m'' < m'$, one can assume the tiebreaker rule that whenever $m'' < m'$, then the color $c_{m'',k'}$ comes in the record r before the color $c_{m',k'}$. This permits one to consider and update only vectors where, for each fixed coordinate k', the colors are in their natural order. Thus one can describe the last appearance records by giving a $k \cdot m$-vector which gives, for each entry of a color $c_{m',k'}$, only the value k', as m' is just equal to the number of k' in this record up to the position of the current entry. As a result, the overall number of last appearance records per node can be bounded by $k^{k\cdot(m-1)}$, and thus a k-dimensional parity game with each coordinate having a range from 1 to m and with n nodes can be translated into a parity game with $2^{\log(k)\cdot k\cdot(m-1)} \cdot n$ nodes and $2 \cdot k \cdot (m-1)$ values.

One computes as before from v and r the set U of current colors and then assigns to the node (v,r) in the parity game the value as follows: If U is winning for Anke, then the value is $2|U|+1$, else it is $2|U|+2$, where one defines that Anke has the odd and Boris the even numbers. Note that $|U| \leq 2 \cdot k \cdot (m-1)$ and the number of values is bounded by $2 \cdot k \cdot (m-1) + 2 \leq 2 \cdot k \cdot m$. In the resulting parity game the number of values divided by the logarithm of the number of nodes is at most 2.

Thus the parity game can be solved in $O((2^{\log(k)\cdot k\cdot m} \cdot n)^{5.45})$ time, and the time for computing the translation is also bounded by this term; see the formulas after Corollary 19. So the same bound applies for the overall running time, as summarized in the theorem, which makes use of the observation of Point [67]. Furthermore, if $k \geq 4$, then

$$\log(2^{\log(k)\cdot k\cdot m} \cdot n) \geq 2 \cdot k \cdot m,$$

as $\log(k) \geq 2$, and one can therefore apply the better bound $O((2^{\log(k)\cdot k\cdot m} \cdot n)^5)$ on the runtime. ∎

Now it is shown that the result is optimal in the following sense: There is no sublinear function f such that the runtime of an algorithm solving the multidimensional parity game with the parameters m, k, n as above is $2^{f(k\cdot\log(k)\cdot m)} \cdot \mathrm{Poly}(n)$, unless the Exponential Time Hypothesis fails. To see this, one either fixes m to a constant which is at least 3 or fixes k to a constant which is at least 2, but one does not fix both variables. Then one obtains either a runtime bound $2^{o(k\cdot\log(k))}$ or $2^{o(m)}$, respectively. It will be shown in the following that both cannot happen unless the Exponential Time Hypothesis fails.

Both results are based on reducing the dominating set problem into the respective decision problem. Here a dominating set of a graph is a set of nodes such that from every node in the graph there is an edge to one of the nodes in the dominating set; for this property one deviates from the usual convention of the nonexistence of self-edges and assumes that every node has an edge to itself.

THEOREM 31. *Assume that one can solve k-dimensional parity games with values from $\{1,2,3\}$ and n' nodes in time $2^{o(k\cdot\log(k))} \cdot \mathrm{Poly}(n')$. Then there is an algorithm which solves the dominating set problem for graphs with n nodes and a target size of*

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

m for the dominating set in time $n^{o(m)}$, and thus the Exponential Time Hypothesis fails.

Proof. Assume that one can solve the k-dimensional parity game problem as in the hypothesis. Suppose a graph H with n nodes $\{1, 2, 3, \ldots, n\}$ and a target size m of the dominating set are given. Now one chooses k to be the least even integer satisfying $k \geq 2$ and

$$m \cdot \lceil \log(n) \rceil \leq k/2 \cdot \lfloor \log(k/2) \rfloor.$$

Note that the dominating set can be described by listing the m nodes using $\lceil \log(n) \rceil$ bits each. Now one reinterprets these bits as $k/2$ numbers of $\log(k/2)$ bits each for the above chosen k. The idea is to represent the $m \cdot \lceil \log(n) \rceil$ bits to describe the dominating set by a sequence of $k/2$ numbers $a_1, a_2, a_3, \ldots, a_{k/2}$ from $\{1, 2, 3, \ldots, k\}$ with the additional requirement that a_i is among the first $k/2$ members of $\{1, 2, 3, \ldots, k\} - \{a_j : j < i\}$ for all i. This requirement is assumed on a_i's throughout the proof, without explicitly stating so.

Boris has in mind a dominating set, and Anke tries to check Boris's answers in order to make sure that the set in mind is correct. For this, one needs to check whether the $m \cdot \lceil \log(n) \rceil$ bits representing the dominating set are consistent with $k/2 \lfloor \log(k/2) \rfloor$ bits of a_i's. To check this, the statement "choice (j, r) is consistent with $(w, \tilde{m})$" means the following condition: the binary representations $d_1 d_2 d_3 \ldots d_{\lfloor \log(k/2) \rfloor}$ of $(r-1)$ and $w_1 w_2 w_3 \ldots w_{\lceil \log(n) \rceil}$ of w satisfy that for all i, h with $1 \leq i \leq \log(k/2)$ and $1 \leq h \leq \lceil \log(n) \rceil$, if $(j-1) \cdot \lfloor \log(k/2) \rfloor + i = (\tilde{m} - 1) \cdot \lceil \log(n) \rceil + h$, then $d_i = w_h$.

The game graph will be given below. The game goes infinitely often through the following rounds where in each round the game goes through steps 1 and 2 and then a finite number of repetitions of steps 3 and 4, where the number of repetitions is bounded by $k/2$, followed by step 5, which takes the game back to step 1.

The following descriptions of a round also give the nodes which are in the game, along with edges, values of the nodes, and the players to move. All the nodes, except the nodes of the form $(0, b, B)$ described in step 5, have value vector $(1, 1, 1, \ldots, 1)$. Below B is always a subset of $\{1, 2, 3, \ldots, k\}$, $a_1, a_2, a_3, \ldots, a_{k/2} \in \{1, 2, 3, \ldots, k\}$, and v, w are vertices of H. Intuitively, B gives the choices $a_1, a_2, \ldots$, used by Boris, to describe the dominating set as mentioned above; here the ordering of members of B is based on the order they entered set B in the play.

1. In each round, the game starts in a node called (0). There are edges from node (0) to nodes (v) for each vertex v in H.
   Thus, at node (0) Anke chooses a node v of the graph, for which it is asking Boris to give a neighbor from the dominating set, and moves to node (v).
2. The nodes (v), for vertices v in H, have edges to nodes of the form $(\tilde{m}, w, a_i, B)$, where $i = 1$, $B = \emptyset$, w is a neighbor of v in H, $1 \leq \tilde{m} \leq m$, and the choice $(1, a_1)$ is consistent with $(w, \tilde{m})$ (note that a_1 is the a_1th member of $\{1, 2, 3, \ldots, k\}$). Boris moves in the nodes (v) for v being a vertex in H.
   Intuitively, the intention of Boris moving from (v) to $(\tilde{m}, w, a_i, B)$, with $i = 1$, $B = \emptyset$, and w being a neighbor of v, is that w is the $\tilde{m}$th vertex in the dominating set chosen by Boris.
3. For $\tilde{m} \in \{1, 2, 3, \ldots, m\}$, w a vertex of H, $a_i \in \{1, 2, 3, \ldots, k\} - B$, and the cardinality of B being less than $k/2$, there exists a node $(\tilde{m}, w, a_i, B)$. The node $(\tilde{m}, w, a_i, B)$ with $a_i \notin B$ has edges to $(\tilde{m}, w, a_i, B \cup \{a_i\})$ and to $(0, b, B \cup \{a_i\})$, where $b \in \{1, 2, 3, \ldots, k\} - (B \cup \{a_i\})$.
   Anke moves in nodes of the form $(\tilde{m}, w, a_i, B)$, with $a_i \notin B$.
   Intuitively, Anke can move from $(\tilde{m}, w, a_i, B)$, where $a_i \notin B$, either to $(\tilde{m}, w,$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

$a_i, B \cup \{a_i\}$) and indicate that Boris should reveal more information (only possible when $|B \cup \{a_i\}| < k/2$) or move to a node $(0, b, B \cup \{a_i\})$ where $b \in \{1, 2, 3, \ldots, k\} - \{a_i\} - B$, which indicates visiting a node with certain value; see item 5 below.

4. For $\tilde{m} \in \{1, 2, 3, \ldots, m\}$, w a vertex of H, $a_{i-1} \in B$, and the cardinality of B being less than $k/2$, there is a node $(\tilde{m}, w, a_{i-1}, B)$, and this has edges to nodes of the form $(\tilde{m}, w, a_i, B)$, where $a_i \notin B$ and the choice (i, r) is consistent with $(w, \tilde{m})$, where a_i is the rth member of $\{1, 2, 3, \ldots, k\} - B$. Boris moves in nodes of the form $(\tilde{m}, w, a_{i-1}, B)$ with $a_{i-1} \in B$.
   Intuitively, Boris has to select a_i and move to $(\tilde{m}, w, a_i, B)$ where $a_i \notin B$; at that node it is then Anke's turn to move as described in step 3.
5. There are nodes of the form $(0, b, B)$ with $B \subset \{1, 2, 3, \ldots, k\}$ and $b \in \{1, 2, 3, \ldots, k\} - B$. There is exactly one edge from such a node, and it goes to (0). Boris moves in the nodes of the form $(0, b, B)$.
   The nodes $(0, b, B)$ are the only nodes with a value vector different from $(1, 1, 1, \ldots, 1)$. Here the value vector $(m_1, m_2, m_3, \ldots, m_k)$ of a node $(0, b, B)$ is defined by the equation

$$m_h = \begin{cases} 1 & \text{if } h \notin B \cup \{b\}, \\ 2 & \text{if } h = b, \\ 3 & \text{if } h \in B. \end{cases}$$

   Intuitively, Boris moves from this node to (0), and the next round of the game starts in step 1.

In the case that there is a dominating set of size m, Boris can always choose in the game nodes $(\ldots, B)$ such that the sets B of the form $\{a_j : j < i\}$ occurring there are ordered under inclusion, and these sets can be computed from a fixed sequence $a_1, a_2, a_3, \ldots, a_{k/2}$ derived from a binary representation describing the dominating set. In a play, whenever it is Boris's turn to move, the sets B in the last component of the names of the nodes would be derived using $a_1, a_2, \ldots, a_{k/2}$ as above. Thus, in any particular play there is a largest set B such that nodes of the form $(\cdot, \cdot, \cdot, B)$ are visited infinitely often in the play, and all other sets B', with node $(\cdot, \cdot, \cdot, B')$ occurring in the play, satisfy $B' \subseteq B$. Thus for this largest set B, player Anke has to choose b, when going to node $(0, b, B)$, to be nonmember of B, and so the vectors $(m_1, m_2, m_3, \ldots, m_k)$ when moving to $(0, b, B)$ will have that $m_b = 2$ and $m_h = 3$ for all $h \in B$; furthermore, m_b will never be 3. It follows that Anke cannot satisfy the condition that the limit superior of each m_h over the play is odd, and thus Boris is winning the game.

In the case that there is no dominating set of size m, Boris cannot achieve that all the sets B occurring in nodes of the form $(\ldots, B)$ are comparable. To see this, one can assume without loss of generality that the strategy of Boris is fixed, that Anke knows the strategy, and that she exploits its weakness. Now, as there is no dominating set of size m, Boris has selected two different nodes $w, \tilde{w}$ at the same position $\tilde{m}$ when Anke asks for the node in the dominating set that are neighbors of suitable nodes v and $\tilde{v}$. As w, $\tilde{w}$ get coded into different witnesses $(a_1, a_2, a_3, \ldots, a_{k/2})$ and $(\tilde{a}_1, \tilde{a}_2, \tilde{a}_3, \ldots, \tilde{a}_{k/2})$, there is a first i where $a_i \neq \tilde{a}_i$. Thus Anke can go alternately from (0) to (v) and $(\tilde{v})$ and then run through the cycles of building up the witnesses until she reaches the nodes $(\tilde{m}, w, a_i, B)$ and $(\tilde{m}, \tilde{w}, \tilde{a}_i, B)$, respectively, where $B = \{a_j : j < i\} = \{\tilde{a}_j : j < i\}$. From these nodes, Anke goes to $(0, \tilde{a}_i, B \cup \{a_i\})$ and $(0, a_i, B \cup \{\tilde{a}_i\})$, respectively, and the game returns from them to (0). Thus the limit

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

superior $(m_1, m_2, m_3, \ldots, m_k)$ of the value vectors of the play will satisfy that $m_h = 3$ for all $h \in B \cup \{a_i, \tilde{a}_i\}$ and $m_h = 1$ for all $h \notin B \cup \{a_i, \tilde{a}_i\}$. So the m_h are odd for all $h \in \{1, 2, 3, \ldots, k\}$, and Anke wins the game.

In summary, Boris can win the so constructed multidimensional parity game iff the given graph has a dominating set of size m.

One can bound the number n' of nodes in this game by the formula $1 + n + m \cdot n \cdot k \cdot 2^k + k \cdot 2^k \leq 4n^2 k 2^k$, as $m \leq n$. Thus, $2^{o(k \log(k))} poly(n')$ is in $2^{o(k \log(k))} poly(n)$, which in turn is in $2^{o(m \log(n))} poly(n)$ and thus in $n^{o(m)}$. Thus, if there is an algorithm which solves k-dimensional parity games with n' nodes in time $2^{o(k \cdot \log(k))} \cdot \mathrm{Poly}(n')$, then one can solve the dominating set problem in time $n^{o(m)}$.

Now one can use the following result of Chen et al. [21, Theorem 5.8]: If one can solve the problem of determining whether a graph of n nodes has a dominating set of size m in time $n^{o(m)}$, then the Exponential Time Hypothesis fails. This connection then translates into the following bound: If k-dimensional parity games with n' nodes and values from $\{1, 2, 3\}$ can, uniformly in n', k, be decided in time $2^{o(k \cdot \log(k))} \cdot \mathrm{Poly}(n')$, then the Exponential Time Hypothesis fails. ∎

The next result is again a translation of the dominating set problem. One needs dimension two, and the main technique is to compare the bits in the witnesses for a dominating set. Note that dimension one is equivalent to the normal parity games; thus requiring dimension two is unavoidable.

THEOREM 32. *Given a graph H with n nodes and a number m with the constraint that $2 \leq m \leq n$, one can compute in time polynomial in n a two-dimensional parity game with n' nodes and m' colors such that the following conditions hold:*

- $m' = 2m \cdot \lceil \log(n) \rceil + 1$*;*
- $n' = 1 + (m+1) \cdot n + 2m \cdot \lceil \log(n) \rceil$*; and*
- *the given graph H has a dominating set of size up to m iff player Boris has a winning strategy in the resulting two-dimensional parity game.*

Furthermore, the so obtained two-dimensional parity games cannot be solved in time $2^{o(m')} \cdot \mathrm{Poly}(n')$, provided that the Exponential Time Hypothesis holds.

Proof. Consider the nodes of graph H, and let them have as names the first n strings from $\{0,1\}^{\lceil \log n \rceil}$. Without loss of generality assume $n \geq 4$. The proof is similar to the proof of Theorem 31 except that the graph construction and the checking of consistency of the dominating set are modified to have a constant bound on the dimension rather than on the number of values. The basic idea of the game is to go through following rounds:

1. Anke selects a vertex v in the graph H.
2. Boris selects a neighboring vertex w of v in the graph H and a number $\tilde{m}$ to indicate that w is the $\tilde{m}$th member of the dominating set.
3. Anke selects a bit position $o \in \{1, 2, 3, \ldots, \lceil \log n \rceil\}$; if the oth bit of the name of w is 1, then Anke moves in the game to a node with value $(2(\tilde{m}-1) \cdot \lceil \log(n) \rceil + 2o + 1, 2(\tilde{m}-1) \cdot \lceil \log(n) \rceil + 2o)$, else Anke moves to a node with value $(2(\tilde{m}-1) \cdot \lceil \log(n) \rceil + 2o, 2(\tilde{m}-1) \cdot \lceil \log(n) \rceil + 2o + 1)$.
4. Boris moves back to the start of the game, where Anke selects a node in graph H.

The values of all nodes except those at step 3 above in the game will be "small." In the case that there is a dominating set of size m, Boris can play a memoryless winning strategy for the game by always selecting the right node in the second step; this will ensure that the limit superior of the values in the two dimensions are of different parity. In the case in which there is no dominating set, when playing memoryless, Boris has to be inconsistent and choose, for two different vertices v, v' chosen by Anke

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

in step 1 above, two different vertices w, w' at the same position $\tilde{m}$ of the candidate for the dominating set. These w, w' will differ in some bit position o; thus Anke can then force the game to go through the nodes with values $(2(\tilde{m}-1)\cdot\lceil\log(n)\rceil+2o$, $2(\tilde{m}-1)\cdot\lceil\log(n)\rceil+2o+1)$ and $(2(\tilde{m}-1)\cdot\lceil\log(n)\rceil+2o+1, 2(\tilde{m}-1)\cdot\lceil\log(n)\rceil+2o)$ infinitely often to win the game.

Based on the above motivation, the nodes and edges of the game are described as follows. Note that 0 is not a name of any vertex in H.

1. The node $(0,0)$ has the value $(1,1)$. The node $(0,0)$ is the starting node, and Anke moves in this node. There is an edge from $(0,0)$ to $(v,0)$ for all vertices v in H.
2. There are nodes $(v,0)$ for v being a vertex in H. The values of these nodes are $(1,1)$. Boris moves in these nodes. For any w such that (v,w) is an edge in H, there is an edge from $(v,0)$ to $(w,\tilde{m})$ for $\tilde{m}$ with $1\le\tilde{m}\le m$.
   Intuitively, a move from $(v,0)$ to $(w,\tilde{m})$ denotes that Boris is specifying the neighbor w of v as being the $\tilde{m}$th element of the dominating set chosen by it.
3. There are nodes $(w,\tilde{m})$ for $w,\tilde{m}$, with w being a vertex in H and $1\le\tilde{m}\le m$; the values of these nodes are $(1,1)$, and Anke moves in these nodes.
   For each $o\in\{1,2,3,\ldots,\lceil\log n\rceil\}$, there is an edge from $(w,\tilde{m})$ to node $(0, 2(\tilde{m}-1)\cdot\lceil\log(n)\rceil+2o-b)$, where b is the oth bit of w, that is, $b=b_o$ where $w=b_1b_2b_3\ldots b_{\lceil\log n\rceil}$.
   Intuitively, Anke chooses o to ask Boris to prove that the oth bit of the $\tilde{m}$th vertex in the dominating set is always consistent.
4. There are nodes $(0,h)$ for all $h\in\{1,2,3,\ldots,2m\lceil\log n\rceil\}$. The value of the node $(0,h)$ is $(h,h+1)$ when h is even, and its value is $(h+2,h+1)$ when h is odd. Boris moves in these nodes. There is an edge from $(0,h)$ to $(0,0)$.

In the case in which there is a dominating set $\{w_1,w_2,w_3,\ldots,w_m\}$, Boris moves in step 2 above always from a node $(v,0)$ to a node $(w_{\tilde{m}},\tilde{m})$ such that there is an edge in H from v to $w_{\tilde{m}}$. This is a winning strategy, as then for all positions o in a $w_{\tilde{m}}$, as chosen by Anke in step 3 above, the bit b is always the same, and thus the limit superior of the values attained in a play is of the form $(2(\tilde{m}-1)\cdot\lceil\log(n)\rceil+2o+b, 2(\tilde{m}-1)\cdot\lceil\log(n)\rceil+2o+1-b)$ for some $\tilde{m}$ and o, with b being the oth bit of $w_{\tilde{m}}$.

If there is no dominating set of size m and Boris plays a memoryless winning strategy, then he will on two nodes $(0,v)$ and $(0,v')$ move to two different nodes $(w,\tilde{m})$ and $(w',\tilde{m})$, as otherwise Boris would have a consistent dominating set contradicting the assumption. Now there is a position o such that the bits b and b' of w and w' at this position differ. Without loss of generality assume $b=0$ and $b'=1$. Therefore Anke can move to nodes with value $(2(\tilde{m}-1)\cdot\lceil\log(n)\rceil+2o+b, 2(\tilde{m}-1)\cdot\lceil\log(n)\rceil+2o+1-b)$ and $(2(\tilde{m}-1)\cdot\lceil\log(n)\rceil+2o+b', 2(\tilde{m}-1)\cdot\lceil\log(n)\rceil+2o+1-b')$ which are of the form $(h,h+1)$ and $(h+1,h)$ for some even h. That is, by alternately moving to the nodes $(0,v)$ and $(0,v')$ when in node $(0,0)$, and moving to the node $(0,2(\tilde{m}-1)\cdot\lceil\log n\rceil+2o-b)$ when in node $(w,\tilde{m})$, where b is the oth bit of w, Anke will achieve that the limit superior of a play is $(h+1,h+1)$ for some even h, and therefore the game is won by Anke. It follows that Boris's memoryless strategy is not a winning strategy, and therefore he does not have a winning strategy at all. In summary, Boris wins the two-dimensional parity game iff there is a dominating set of size m in H.

The number n' of nodes is the sum of 1 (for node $(0,0)$) and n (for nodes $(v,0)$, with v being a vertex in H) and $n\cdot m$ (for nodes $(w,\tilde{m})$, with w being a vertex in H and $\tilde{m}\in\{1,2,3,\ldots,m\}$) and $2m\cdot\lceil\log(n)\rceil$ (for nodes $(0,h)$). The number m' is just $2m\cdot\lceil\log(n)\rceil+1$, as h is bounded by $2m\lceil\log n\rceil$.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Now assume that there would be an algorithm for this problem which runs in time $2^{o(m')} \cdot \mathrm{Poly}(n')$. Let $f(m')$ be a function in $o(m')$ such that the runtime is in $2^{f(m')} \cdot \mathrm{Poly}(n')$. Now, one can replace $f(m')$ by $g(m') \cdot m'$, where $g(m') = \sup\{f(m'')/m'' : m'' \geq m'\}$, noting that g is monotonically nonincreasing. As g is monotonically nonincreasing, one can also obtain that

$$2^{f(2m \cdot \lceil \log(n) \rceil + 1)} \leq 2^{g(2m) \cdot (2m \cdot \lceil \log(n) \rceil + 1)} = n^{o(m)}.$$

As $n' \leq n^2 \cdot \lceil \log(n) \rceil$, one can conclude that the runtime for finding a solution to the existence of a dominating set is $n^{o(m)} \cdot \mathrm{Poly}(n)$, which is $n^{o(m)}$. However, Chen et al. [21, Theorem 5.8] showed that under these hypotheses, the Exponential Time Hypothesis fails. This completes the proof. ∎

Recall that the question of whether a problem is in **FPT** depends on which parameters are considered constants and which are running parameters. The dependence of the algorithm runtime on the constant parameters can be arbitrary, but that on the running parameters has to be a polynomial of fixed degree which is independent of the constant parameters. Theorem 30 shows that if one fixes both parameters m and k as constants, then multidimensional parity games are in **FPT**. Theorems 31 and 32 show that, unless the Exponential Time Hypothesis is wrong, multidimensional parity games are not fixed parameter tractable in the case that only one of the parameters m and k is fixed as a constant. Bruyère, Hautem, and Raskin [10] investigate the fixed parameter tractability of generalizations of multidimensional parity games and related games in detail.

There is some connection between parity games and mean payoff games; for the latter, Velner et al. [77] studied the computational complexity of the multidimensional analogue of mean payoff games and discovered that one has to distinguish the cases of evaluation by limit superior and evaluation by limit inferior in the multidimensional game. For the case of evaluation by limit superior, they are in **NP** ∩ **coNP**; for the case of evaluation by limit inferior, they are **coNP**-complete. In light of the above result, multidimensional parity games are more related to the evaluation of limit inferior.

6. Conclusion. The progress reported in this paper shows that solving parity games is not as difficult as it was widely believed. Indeed, parity games can be solved in quasi-polynomial time—the previous bounds were roughly $n^{O(\sqrt{n})}$—and they are fixed parameter tractable with respect to the number m of values (a.k.a. colors or priorities)—the previously known algorithms were roughly $O(n^{m/3})$. These results are in agreement with earlier results stating that parity games can be solved in **UP** ∩ **coUP** [52] and that there are subexponential algorithms to solve the problem [55].

In spite of current progress, the original question, asked by Emerson and Jutla [32] in 1991 and others, of whether parity games can be decided in polynomial time still remains an important open question.

The above results on parity games are then used to give an algorithm of runtime $O((m^m \cdot n)^5)$ for colored Muller games with n nodes and m colors; this upper bound is almost optimal, since an algorithm with runtime $O((2^m \cdot n)^c)$, for some constant c, only exists in the case that the Exponential Time Hypothesis fails.

One might ask whether the results obtained for parity games permit further transfers to Muller games, for example, in special cases where (a) player Anke can employ a memoryless winning strategy due to the special type of game, or (b) one does not permit player Anke to use strategies other than memoryless ones. Note that case (b) differs from case (a), as in case (b) the condition on using memoryless strategies can

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

be restrictive, while case (a) applies to Muller games where one knows that "if Anke has a winning strategy, then she has a memoryless winning strategy." Case (a) was analyzed by Emerson [30], McNaughton [61], and Zielonka [80]; it applies to Muller games where the winning condition of player Boris is closed under union [30, 80].

The above-mentioned lower bound directly also applies to case (a). For case (b), the complexity class of the general problem is also in the polynomial hierarchy but not **PSPACE**-complete (unless $\mathbf{PSPACE} = \mathbf{\Sigma}_2^P$) as the decision problem for colored Muller games; however, the algorithmic bounds are much worse, as one can code **NP**-hard problems into instances with four colors.

Another variant of parity games is to consider vectors of values where in the default case player Anke wins if the limit superior of all of each of these values is odd and player Boris wins if the limit superior of at least one of the values is even. For this type of game, the k-dimensional parity game with values from 1 to m and n nodes can be decided in time $O((2^{k\cdot\log(k)\cdot m}\cdot n)^{5.45})$ and slight improvements of the exponent 5.45 might be possible. However, much better algorithms, even for the special case where either k or m is constant, would imply that the Exponential Time Hypothesis fails, which seems unlikely. More precisely, under the assumption that the Exponential Time Hypothesis is true, there are no algorithms which solve k-dimensional parity games with m values and n nodes in time $2^{f(k\cdot\log(k)\cdot m)}\cdot \mathrm{Poly}(n)$ for any sublinear function f; this even holds when either m is fixed to be a constant at least 3 or k is fixed to be a constant which is at least 2, but not both are fixed. This shows that the multidimensional parity games are very similar to colored Muller games with respect to the runtime behavior of algorithms to solve them.

Acknowledgments. The authors would like to thank Krishnendu Chatterjee, Sasha Rubin, Sven Schewe, and Moshe Vardi for correspondence and comments. Further thanks go to the referees of the STOC 2017 paper and of this journal for numerous suggestions. A referee of this journal as well as Rod Downey pointed out that the lower bound for Muller games is more suitably done with reference to the Exponential Time Hypothesis. There was an error caused by a mis-citation in the corresponding lower bound result of the conference version of this paper.

REFERENCES

[1] C. Beeri, *On the membership problem for functional and multivalued dependencies in relational databases*, ACM Trans. Database Syst., 5 (1980), pp. 241–259.

[2] M. Bellare and S. Goldwasser, *The complexity of decision versus search*, SIAM J. Comput., 23 (1994), pp. 97–119, https://doi.org/10.1137/S0097539792228289.

[3] J. Bernet, D. Janin, and I. Walukiewicz, *Permissive strategies: From parity games to safety games*, Theor. Inform. Appl., 36 (2002), pp. 251–275.

[4] D. Berwanger and E. Grädel, *Fixed-point logics and solitaire games*, Theory Comput. Syst., 37 (2004), pp. 675–694.

[5] H. Björklund, S. Sandberg, and S. Vorobyov, *On Fixed-Parameter Complexity of Infinite Games*, Technical report 2003-038, Uppsala University, Uppsala, Sweden, 2003.

[6] H. Björklund, S. Sandberg, and S. Vorobyov, *Memoryless determinacy of parity and mean payoff games: A simple proof*, Theoret. Comput. Sci., 310 (2004), pp. 365–378.

[7] H. L. Bodlaender, M. J. Dinneen, and B. Khoussainov, *On game-theoretic models of networks*, in Proceedings of the Twelfth International Symposium on Algorithms and Computation, ISAAC 2001, Christchurch, New Zealand, 2001, Lecture Notes in Comput. Sci. 2223, Springer, 2001, pp. 550–561.

[8] H. L. Bodlaender, M. J. Dinneen, and B. Khoussainov, *Relaxed update and partition network games*, Fund. Inform., 49 (2002), pp. 301–312.

[9] A. Browne, E. M. Clarke, S. Jha, D. E. Long, and W. R. Marrero, *An improved algorithm for the evaluation of fixpoint expressions*, Theoret. Comput. Sci., 178 (1997), pp. 237–255.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

[10] V. Bruyère, Q. Hautem, and J.-F. Raskin, *Games with Lexicographically Ordered ω-Regular Objectives*, preprint, https://arxiv.org/abs/1707.05968, 2017.
[11] C. S. Calude, S. Jain, B. Khoussainov, W. Li, and F. Stephan, *Deciding parity games in quasipolynomial time*, in STOC 2017 Theory Fest: Forty-Ninth Annual ACM Symposium on Theory of Computing, Montreal, Canada, ACM, 2017, pp. 252–263.
[12] F. Canavoi, E. Grädel, and R. Rabinovich, *The discrete strategy improvement algorithm for parity games and complexity measures for directed graphs*, Theoret. Comput. Sci., 560 (2014), pp. 235–250.
[13] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer, *Alternation*, J. ACM, 28 (1981), pp. 114–133.
[14] K. Chatterjee, *Comments on the Quasipolynomial Time Algorithm*, private communication, 2017.
[15] K. Chatterjee and T. A. Henzinger, *Strategy Improvement and Randomized Subexponential Algorithms for Stochastic Parity Games*, in Proceedings of the Twenty-Third Annual Symposium on Theoretical Aspects of Computer Science, STACS 2006, Marseille, France, 2006, Lecture Notes in Comput. Sci. 3885, Springer, 2006, pp. 512–523.
[16] K. Chatterjee, T. A. Henzinger, and M. Jurdziński, *Mean-payoff parity games*, in Proceedings of the Twentieth Annual IEEE Symposium on Logic in Computer Science, LICS 2005, IEEE, 2005, pp. 178–187.
[17] K. Chatterjee, T. A. Henzinger, and N. Piterman, *Generalized parity games*, in Foundations of Software Science and Computational Structures, Tenth International Conference, FOSSACS 2007, Lecture Notes in Comput. Sci. 4423, Springer, 2007, pp. 153–167.
[18] K. Chatterjee, M. Jurdziński, and T. A. Henzinger, *Quantitative stochastic parity games*, in Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, ACM, SIAM, 2004, pp. 121–130.
[19] K. Chatterjee, M. Randour, and J.-F. Raskin, *Strategy synthesis for multi-dimensional quantitative objectives*, Acta Inform., 51 (2014), pp. 129–163.
[20] K. Chatterjee and Y. Velner, *Hyperplane separation technique for multidimensional mean-payoff games*, in Concurrency Theory – Twenty-Fourth International Conference, CONCUR 2013, Buenos Aires, Argentina, 2013, Lecture Notes in Comput. Sci. 8052, Springer, 2013, pp. 500–515.
[21] J. Chen, X. Huang, I. A. Kanj, and G. Xia, *Strong computational lower bounds via parameterized complexity*, J. Comput. System Sci., 72 (2006), pp. 1346–1367.
[22] S. A. Cook, *Path systems and language recognition*, in Proceedings of the Second Annual ACM Symposium on Theory of Computing, STOC 1970, Northampton, MA, ACM, 1970, pp. 70–72.
[23] S. A. Cook, *The complexity of theorem proving procedures*, in Proceedings of the Third Annual ACM Symposium on the Theory of Computing, STOC 1971, Shaker Heights, OH, ACM, 1971, pp. 151–158.
[24] A. Dawar, F. Horn, and P. Hunter, *Complexity Bounds for Muller Games*, manuscript, 2011.
[25] A. Di Stasio, A. Murano, G. Perelli, and M. Y. Vardi, *Solving parity games using an automata-based algorithm*, in Twenty-First International Conference on Implementation and Application of Automata, CIAA 2016, Seoul, South Korea, Lecture Notes in Comput. Sci. 9705, Springer, 2016, pp. 64–76.
[26] C. Dittmann, S. Kreutzer, and A. I. Tomescu, *Graph operations on parity games and polynomial-time algorithms*, Theoret. Comput. Sci., 614 (2016), pp. 97–108.
[27] R. G. Downey and M. R. Fellows, *Parameterised Complexity*, Springer, Heidelberg, 1999.
[28] R. G. Downey and M. R. Fellows, *Fundamentals of Parameterized Complexity Theory*, Springer, Heidelberg, 2013.
[29] S. Dziembowski, M. Jurdziński, and I. Walukiewicz, *How much memory is needed to win infinite games?*, in Proceedings of the Twelfth Annual IEEE Symposium on Logic in Computer Science, LICS 1997, IEEE, 1997, pp. 99–110.
[30] E. A. Emerson, *Automata, tableaux, and temporal logics*, in Proceedings of the Workshop on Logic of Programs, Lecture Notes in Comput. Sci. 193, Springer, 1985, pp. 79–88.
[31] E. A. Emerson and C. S. Jutla, *The complexity of tree automata and logics of programs*, in Proceedings of the 29th Annual Symposium on Foundations of Computer Science, 1988, IEEE, pp. 328–337.
[32] E. A. Emerson and C. S. Jutla, *Tree automata, μ-calculus and determinacy*, in Proceedings of the 32nd Annual Symposium on Foundations of Computer Science, 1991, IEEE, pp. 368–377.
[33] E. A. Emerson, C. S. Jutla, and A. P. Sistla, *On model checking for the μ-calculus and its*

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

fragments, Theoret. Comput. Sci., 258 (2001), pp. 491–522.

[34] J. Fearnley, S. Jain, B. de Keijzer, S. Schewe, F. Stephan, and D. Wojtczak, *An ordered approach to solving parity games in quasi- polynomial time and quasi-linear space*, Internat. J. Software Tools Technol. Transfer, 21 (2019), pp. 325–349.

J. Fearnley, S. Jain, B. de Keijzer, S. Schewe, F. Stephan and D. Wojtczak. An Ordered Approach to Solving Parity Games in Quasi-Polynomial Time and Quasi-Linear Space. In *International Journal on Software Tools for Technology Transfer*, 21 (2019), pp. 325–349. Special issue on SPIN 2017.

[35] O. Finkel and S. Todorčević, *The isomorphism relation between tree-automatic structures*, Cent. Eur. J. Math., 8 (2010), pp. 299–313.

[36] O. Finkel and S. Todorčević, *A hierarchy of tree-automatic structures*, J. Symbolic Logic, 77 (2012), pp. 350–368.

[37] J. Flum and M. Grohe, *Parameterized Complexity Theory*, Springer, 2006.

[38] O. Friedmann, *An exponential lower bound for the parity game strategy improvement algorithm as we know it*, in Logic in Computer Science, LICS 2009, 2009, pp. 145–156.

[39] O. Friedmann and M. Lange, *Solving parity games in practice*, in Automated Technology for Verification and Analysis, Seventh International Symposium, ATVA 2009, Macao, China, 2009, Lecture Notes in Comput. Sci. 5799, Springer, 2009, pp. 182–196.

[40] J. Gajarský, M. Lampis, K. Makino, V. Mitsou, and S. Ordyniak, *Parameterized algorithms for parity games*, in Mathematical Foundations of Computer Science, MFCS 2015, Lecture Notes in Comput. Sci. 9235, Springer, 2015, pp. 336–347.

[41] A. Gandhi, B. Khoussainov, and J. Liu, *Efficient algorithms for games played on trees with back-edges*, Fund. Inform., 111 (2011), pp. 391–412.

[42] H. Gimbert and R. Ibsen-Jensen, *A Short Proof of Correctness of the Quasi-Polynomial Time Algorithm for Parity Games*, preprint, https://arxiv.org/abs/1702.01953, 2017.

[43] E. Grädel, P. G. Kolaitis, L. Libkin, M. Marx, J. Spencer, M. Y. Vardi, Y. Venema, and S. Weinstein, *Finite Model Theory and Its Applications*, Springer, 2007.

[44] A. Grinshpun, P. Phalitnonkiat, S. Rubin, and A. Tarfulea, *Alternating traps in Muller and parity games*, Theoret. Comput. Sci., 521 (2014), pp. 73–91.

[45] Y. Gurevich and L. Harrington, *Trees, automata and games*, in Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, STOC 1982, San Francisco, CA, 1982, pp. 60–65.

[46] F. Horn, *Dicing on the Streett*, Inform. Process. Lett., 104 (2007), pp. 1–9.

[47] F. Horn, *Explicit Muller games are* **PTIME**, in IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2008, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2008, pp. 235–245.

[48] P. W. Hunter, *Complexity and Infinite Games on Finite Graphs*, Ph.D. thesis, University of Cambridge, 2007.

[49] P. W. Hunter and A. Dawar, *Complexity bounds for regular games*, in Proceedings of the Thirtieth International Symposium on Mathematical Foundations of Computer Science, MFCS 2005, Lecture Notes in Comput. Sci. 3618, Springer, 2005, pp. 495–506.

[50] N. Immerman, *Number of quantifiers is better than number of tape cells*, J. Comput. System Sci., 22 (1981), pp. 384–406.

[51] H. Ishihara and B. Khoussainov, *Complexity of some infinite games played on finite graphs*, in Graph-Theoretic Concepts in Computer Science, Proceedings of the Twenty-Eighth International Workshop, WG 2002, Cesky Krumlov, Czech Republic, 2002, Lecture Notes in Comput. Sci. 2573, Springer, 2002, pp. 270–281.

[52] M. Jurdziński, *Deciding the winner in parity games is in* $\mathbf{UP} \cap \mathbf{coUP}$, Inform. Process. Lett., 68 (1998), pp. 119–124.

[53] M. Jurdziński, *Small progress measures for solving parity games*, in STACS 2000, Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Comput. Sci. 1770, Springer, 2000, pp. 290–301.

[54] M. Jurdziński and R. Lazić, *Succinct progress measures for solving parity games*, in Logic in Computer Science, LICS 2017; preprint, https://arxiv.org/abs/1702.05051, 2017.

[55] M. Jurdziński, M. Paterson, and U. Zwick, *A deterministic subexponential algorithm for solving parity games*, SIAM J. Comput., 38 (2008), pp. 1519–1532, https://doi.org/10.1137/070686652.

[56] I. Khaliq and G. Imran, *Reachability games revisited*, in Proceedings of the Second International Conference on Advances and Trends in Software Engineering, SOFTENG 2016, Lisbon, Portugal, 2016, International Academy, Research and Industry Association (IARIA), Wellington, DE, 2016, pp. 129–133.

[57] B. Khoussainov and A. Nerode, *Automata Theory and Its Applications*, Birkhäuser, 2001.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

[58] D. Kuske, J. Liu, and M. Lohrey, *The isomorphism problem for omega-automatic trees*, in Proceedings of Computer Science Logic, CSL 2010, Lecture Notes in Comput. Sci. 6247, Springer, 2010, pp. 396–410.
[59] K. Lehtinen, *A modal μ perspective on solving parity games in quasi-polynomial time*, in Proceedings of the Thirty-Third Conference on Logic in Computer Science, LICS 2018, Oxford, UK, 2018, pp. 639–648.
[60] L. A. Levin, *Universal sequential search problems*, Problemy Peredachi Informatsii, 3 (1973), pp. 115–116.
[61] R. McNaughton, *Infinite games played on finite graphs*, Ann. Pure Appl. Logic, 65 (1993), pp. 149–184.
[62] A. W. Mostowski, *Games with Forbidden Positions*, Technical report 78, Uniwersytet Gdanski, Instytut Matematyki, 1991.
[63] J. Obdrzalek, *Algorithmic Analysis of Parity Games*, Ph.D. thesis, University of Edinburgh, 2006.
[64] V. Petersson and S. G. Vorobyov, *A randomized subexponential algorithm for parity games*, Nordic J. Comput., 8 (2001), pp. 324–345.
[65] N. Piterman, *From nondeterministic Büchi and Streett automata to deterministic parity automata*, Logical Methods Comput. Sci., 3 (2007), pp. 1–21.
[66] N. Piterman and A. Pnueli, *Faster solutions of Rabin and Streett games*, in Proceedings of the Twenty-First IEEE Symposium on Logic in Computer Science, LICS 2006, Seattle, WA, 2006, IEEE Computer Society, 2006, pp. 528–539.
[67] G. Point, *The Synthesis Toolbox: From Modal Automata to Controller Synthesis*, Research report 1342–05, LaBRI – UMR CNRS 5800, 2005.
[68] S. Safra, *On the complexity of ω-automata*, in Proceedings of the Twenty-Ninth IEEE Symposium on Foundations of Computer Science, IEEE, 1988, pp. 319–327.
[69] S. Safra, *Exponential determinization for ω-automata with a strong fairness acceptance condition*, SIAM J. Comput., 36 (2006), pp. 803–814, https://doi.org/10.1137/S0097539798332518.
[70] S. Schewe, *Solving parity games in big steps*, in FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science, Lecture Notes in Comput. Sci. 4855, Springer, 2007, pp. 449–460; journal version published in 2017 [72].
[71] S. Schewe, *From parity and payoff games to linear programming*, in Mathematical Foundations of Computer Science 2009, Proceedings of the Thirty-Fourth International Symposium, MFCS 2009, Novy Smokovec, High Tatras, Slovakia, 2009, Lecture Notes in Comput. Sci. 5734, Springer, 2009, pp. 675–686.
[72] S. Schewe, *Solving parity games in big steps*, J. Comput. System Sci., 84 (2017), pp. 243–262.
[73] H. Seidl, *Fast and simple nested fixpoints*, Inform. Process. Lett., 59 (1996), pp. 303–308.
[74] F. Stephan, *Methods and Theory of Automata and Languages*, Lecture Notes, School of Computing, National University of Singapore, 2016, http://www.comp.nus.edu.sg/~fstephan/fullautomatatheory-nov2016.ps.
[75] C. Stirling, *Bisimulation, modal logic and model checking games*, Log. J. IGPL, 7 (1999), pp. 103–124.
[76] W. Thomas, *On the synthesis of strategies in infinite games*, in Proceedings of the Twelfth International Symposium on Theoretical Aspects of Computer Science, STACS 1995, Lecture Notes in Comput. Sci. 900, Springer, 1995, pp. 1–13.
[77] Y. Velner, K. Chatterjee, L. Doyen, T. A. Henzinger, A. Rabinovich, and J.-F. Raskin, *The complexity of multi-mean-payoff and multi-energy games*, Inform. and Comput., 241 (2015), pp. 177–196.
[78] I. Walukiewicz, *Pushdown processes: Games and model-checking*, Inform. and Comput., 36 (2001), pp. 261–275.
[79] T. Wilkie, *Alternating tree automata, parity games and modal μ-calculus*, Bull. Belg. Math. Soc. Simon Stevin, 8 (2001), pp. 359–391.
[80] W. Zielonka, *Infinite games on finitely coloured graphs with applications to automata on infinite trees*, Theoret. Comput. Sci., 200 (1998), pp. 135–183.

Downloaded 09/15/25 to 139.19.240.132 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.