Conditionally Optimal Algorithms for Generalized Büchi Games*

Krishnendu Chatterjee[1], Wolfgang Dvořák[2], Monika Henzinger[3], and Veronika Loitzenbauer[4]

1 **IST Austria**
2 **University of Vienna, Faculty of Computer Science, Vienna, Austria**
3 **University of Vienna, Faculty of Computer Science, Vienna, Austria**
4 **University of Vienna, Faculty of Computer Science, Vienna, Austria**

Abstract

Games on graphs provide the appropriate framework to study several central problems in computer science, such as verification and synthesis of reactive systems. One of the most basic objectives for games on graphs is the liveness (or Büchi) objective that given a target set of vertices requires that some vertex in the target set is visited infinitely often. We study generalized Büchi objectives (i.e., conjunction of liveness objectives), and implications between two generalized Büchi objectives (known as GR(1) objectives), that arise in numerous applications in computer-aided verification. We present improved algorithms and conditional super-linear lower bounds based on widely believed assumptions about the complexity of (A1) combinatorial Boolean matrix multiplication and (A2) CNF-SAT. We consider graph games with n vertices, m edges, and generalized Büchi objectives with k conjunctions. First, we present an algorithm with running time $O(k \cdot n^2)$, improving the previously known $O(k \cdot n \cdot m)$ and $O(k^2 \cdot n^2)$ worst-case bounds. Our algorithm is optimal for dense graphs under (A1). Second, we show that the basic algorithm for the problem is optimal for sparse graphs when the target sets have constant size under (A2). Finally, we consider GR(1) objectives, with k_1 conjunctions in the antecedent and k_2 conjunctions in the consequent, and present an $O(k_1 \cdot k_2 \cdot n^{2.5})$-time algorithm, improving the previously known $O(k_1 \cdot k_2 \cdot n \cdot m)$-time algorithm for $m > n^{1.5}$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, F.3.1 Specifying and Verifying and Reasoning about Programs

Keywords and phrases Generalized Büchi objective, GR(1) objective, Conditional lower bounds, Graph games, Graph algorithms, Computer-aided verification

Digital Object Identifier 10.4230/LIPIcs.MFCS.2016.25

1 Introduction

Games on graphs. Two-player games on graphs, between player 1 and the adversary player 2, are central in many problems in computer science, specially in formal analysis of reactive systems, where vertices of the graph represent states of the system, edges represent transitions, infinite paths of the graph represent behaviors (or non-terminating executions)

* K. C., M. H., and W. D. are partially supported by the Vienna Science and Technology Fund (WWTF) through project ICT15-003. K. C. is partially supported by the Austrian Science Fund (FWF) NFN Grant No S11407-N23 (RiSE/SHiNE) and an ERC Start grant (279307: Graph Games). For W. D., M. H., and V. L. the research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement no. 340506.

of the system, and the two players represent the system and the environment, respectively. Games on graphs have been used in many applications related to verification and synthesis of systems, such as, synthesis of systems from specifications and controller-synthesis [30, 54, 55], verification of open systems [8], checking interface compatibility [31], well-formedness of specifications [32], and many others. We will distinguish between results most relevant for *sparse graphs*, where the number of edges m is roughly proportional to the number of vertices n, and *dense graphs* with $m = \Theta(n^2)$. Sparse graphs arise naturally in program verification, as control-flow graphs are sparse [57, 28]. Graphs obtained as synchronous product of several components (where each component makes transitions at each step) [45, 23] can lead to dense graphs.

Objectives. Objectives specify the desired set of behaviors of the system. The most basic objective for reactive systems is the *reachability* objective, and the next basic objective is the *Büchi* (*also called liveness or repeated reachability*) objective that was introduced in the seminal work of Büchi [17, 18, 19] for automata over infinite words. Büchi objectives are specified with a target set T and the objective specifies the set of infinite paths in the graph that visit some vertex in the target set infinitely often. Since for reactive systems there are multiple requirements, a very central objective to study for games on graphs is the conjunction of Büchi objectives, which is known as generalized Büchi objective. Finally, currently a very popular class of objectives to specify behaviors for reactive systems is called the GR(1) (generalized reactivity (1)) objectives [53]. A GR(1) objective is an implication between two generalized Büchi objectives.

We present a brief discussion about the significance of the objectives we consider, for a detailed discussion see [26]. The conjunction of liveness objectives is required to specify progress conditions of mutual exclusion protocols, and deterministic Büchi automata can express many important properties of linear-time temporal logic (LTL) (the de-facto logic to specify properties of reactive systems) [47, 46, 9, 44]. The analysis of reactive systems with such objectives naturally gives rise to graph games with generalized Büchi objectives. Finally, graph games with GR(1) objectives have been used in many applications, such as the industrial example of synthesis of AMBA AHB protocol [14, 36] as well as in robotics applications [35, 21].

Basic problem and conditional lower bounds. In this work we consider games on graphs with generalized Büchi and GR(1) objectives, and the basic algorithmic problem is to compute the *winning set*, i.e., the set of starting vertices where player 1 can ensure the objective irrespective of the way player 2 plays; the way player 1 achieves that is called her *winning strategy*. These are core algorithmic problems in verification and synthesis. For the problems we consider, while polynomial-time algorithms are known, there are no super-linear lower bounds. Since for polynomial-time algorithms unconditional super-linear lower bounds are extremely rare in the whole of computer science, we consider *conditional lower bounds*, which assume that for some well-studied problem the known algorithms are optimal up to some lower-order factors. In this work we consider two such well-studied assumptions: (A1) there is no combinatorial[1] algorithm with running time of $O(n^{3-\varepsilon})$ for any $\varepsilon > 0$ to multiply two $n \times n$ Boolean matrices; or (A2) for all $\varepsilon > 0$ there exists a k such that there is no algorithm for the k-CNF-SAT problem that runs in $O(2^{(1-\varepsilon)\cdot n} \cdot \text{poly}(m))$ time, where n is the number of variables and m the number of clauses. These two assumptions have been used to establish

[1] Combinatorial here means avoiding fast matrix multiplication [48], see also the discussion in [38].

lower bounds for several well-studied problems, such as dynamic graph algorithms [3, 5], measuring the similarity of strings [4, 15, 16, 10, 2], context-free grammar parsing [49, 1], and verifying first-order graph properties [52, 61].

Our results. We consider games on graphs with n vertices, m edges, generalized Büchi objectives with k conjunctions, and target sets of size $b_1, b_2, \ldots, b_k$, and GR(1) objectives with k_1 conjunctions in the assumptions and k_2 conjunctions in the guarantee. Our results are as follows.

- *Generalized Büchi objectives.* The classical algorithm for generalized Büchi objectives requires $O(k \cdot \min_{1 \leq i \leq k} b_i \cdot m)$ time. Further there exists an $O(k^2 \cdot n^2)$-time algorithm via a reduction to Büchi games [13, 26].
  1. *Dense graphs.* Since $\min_{1 \leq i \leq k} b_i = O(n)$ and $m = O(n^2)$, the classical algorithm has a worst-case running time of $O(k \cdot n^3)$. First, we present an algorithm with worst-case running time $O(k \cdot n^2)$, which is an improvement for instances with $\min_{1 \leq i \leq k} b_i \cdot m = \omega(n^2)$. Second, for dense graphs with $m = \Theta(n^2)$ and $k = \Theta(n^c)$ for any $0 < c \leq 1$ our algorithm is optimal under (A1); i.e., improving our algorithm for dense graphs would imply a faster (sub-cubic) combinatorial Boolean matrix multiplication algorithm.
  2. *Sparse graphs.* We show that for $k = \Theta(n^c)$ for any $0 < c \leq 1$, for target sets of constant size, and sparse graphs with $m = \Theta(n^{1+o(1)})$ the basic algorithm is optimal under (A2). In fact, our conditional lower bound under (A2) holds even when each target set is a singleton. Quite strikingly, our result implies that improving the basic algorithm for sparse graphs even with singleton sets would require a major breakthrough in overcoming the exponential barrier for SAT.

  In summary, for games on graphs, we present an improved algorithm for generalized Büchi objectives for dense graphs that is optimal under (A1); and show that under (A2) the basic algorithm is optimal for sparse graphs and constant size target sets.

  The conditional lower bound for dense graphs means in particular that for unrestricted inputs the dependence of the runtime on n cannot be improved, whereas the bound for sparse graphs makes the same statement for the dependence on m. Moreover, as the graphs in the reductions for our lower bounds can be made acyclic by deleting a single vertex, our lower bounds also apply to a broad range of digraph parameters. For instance let w be the DAG-width [12] of a graph, then there is no $O(f(w) \cdot n^{3-\epsilon})$-time algorithm under (A1) and no $O(f(w) \cdot m^{2-\epsilon})$-time algorithm under (A2).
- *GR(1) objectives.* We present an algorithm for games on graphs with GR(1) objectives that has $O(k_1 \cdot k_2 \cdot n^{2.5})$ running time and improves the previously known $O(k_1 \cdot k_2 \cdot n \cdot m)$-time algorithm [43], for $m > n^{1.5}$. Note that since generalized Büchi objectives are special cases of GR(1) objectives, our conditional lower bounds for generalized Büchi objectives apply to GR(1) objectives as well but are not tight.

All our algorithms can easily be modified to also return the corresponding winning strategies for both players within the same time bounds.

Implications. We discuss the implications of our results.

1. *Comparison with related models.* We compare our results for game graphs to the special case of standard graphs (i.e., games on graphs with only player 1) and the related model of Markov decision processes (MDPs) (with only player 1 and stochastic transitions). First note that for reachability objectives, linear-time algorithms exist for game graphs [11, 39],

whereas for MDPs[2] the best-known algorithm has running time $O(\min(n^2, m^{1.5}))$ [29, 26]. For MDPs with reachability objectives, a linear or even $O(m \log n)$ time algorithm is a major open problem, i.e., there exist problems that seem harder for MDPs than for game graphs. Our conditional lower bound results show that under assumptions (A1) and (A2) the algorithmic problem for generalized Büchi objectives is strictly harder for games on graphs as compared to standard graphs and MDPs. More concretely, for $k = \Theta(n)$, (a) for dense graphs ($m = \Theta(n^2)$) and $\min_{1 \le i \le k} b_i = \Omega(\log n)$, our lower bound for games on graphs under (A2) is $\Omega(n^{3-o(1)})$, whereas both the graph and the MDP problems can be solved in $O(n^2)$ time [25, 26]; and (b) for sparse graphs ($m = \Theta(n^{1+o(1)})$) with $\min_{1 \le i \le k} b_i = O(1)$, our lower bound for games on graphs under (A1) is $\Omega(m^{2-o(1)})$, whereas the graph problem can be solved in $O(m)$ time and the MDP problem in $O(m^{1.5})$ time [7, 24]; respectively.

2. *Relation to SAT.* We present an algorithm for game graphs with generalized Büchi objectives and show that improving the algorithm would imply a better algorithm for SAT, and thereby establish an interesting algorithmic connection for classical objectives in game graphs and the SAT problem.

Due to the lack of space, some technical details are omitted. A full version is available at `http://eprints.cs.univie.ac.at/4708/`.

2 Preliminaries

2.1 Basic definitions for Games on Graphs

Game graphs. A *game graph* $\mathcal{G} = ((V, E), (V_1, V_2))$ is a directed graph $G = (V, E)$ with n vertices V and m edges E and a partition of V into *player 1 vertices* V_1 and *player 2 vertices* V_2. Given such a game graph $\mathcal{G}$, we denote with $\overline{\mathcal{G}}$ the game graph where the player 1 and player 2 vertices of $\mathcal{G}$ are interchanged, i.e, $\overline{\mathcal{G}} = ((V, E), (V_2, V_1))$. We use p to denote a player and $\bar{p}$ to denote its opponent. For a vertex $u \in V$, we write $Out(u) = \{v \in V \mid (u, v) \in E\}$ for the set of successor vertices of u and $In(u) = \{v \in V \mid (v, u) \in E\}$ for the set of predecessor vertices of u. If necessary, we refer to the successor vertices in a specific graph by using, e.g., $Out(G, u)$. We denote by $Outdeg(u) = |Out(u)|$ the number of outgoing edges from u, and by $Indeg(u) = |In(u)|$ the number of incoming edges. We assume for technical convenience $Outdeg(u) \ge 1$ for all $u \in V$.

Plays and strategies. A *play* on a game graph is an infinite sequence $\omega = \langle v_0, v_1, v_2, \ldots \rangle$ of vertices such that $(v_\ell, v_{\ell+1}) \in E$ for all $\ell \ge 0$. The set of all plays is denoted by Ω. Given a finite prefix $\omega \in V^* \cdot V_p$ of a play that ends at a player p vertex v, a *strategy* $\sigma : V^* \cdot V_p \to V$ of player p is a function that chooses a successor vertex $\sigma(\omega)$ among the vertices of $Out(v)$. We denote by Σ and Π the set of all strategies for player 1 and player 2 respectively. The play $\omega(v, \sigma, \pi)$ is uniquely defined by a start vertex v, a player 1 strategy $\sigma \in \Sigma$, and a player 2 strategy $\pi \in \Pi$ as follows: $v_0 = v$ and for all $j \ge 0$, if $v_j \in V_1$, then $v_{j+1} = \sigma(\langle v_1, \ldots, v_j \rangle)$, and if $v_j \in V_2$, then $v_{j+1} = \pi(\langle v_1, \ldots, v_j \rangle)$.

Objectives. An objective ψ is a set of plays that is winning for a player. We consider zero-sum games where for a player-1 objective ψ the complementary objective $\Omega \setminus \psi$ is winning

[2] For MDPs the winning set refers to the almost-sure winning set that requires that the objective is satisfied with probability 1.

for player 2. In this work we consider only *prefix independent objectives*, for which the set of desired plays is determined by the set of vertices $\mathrm{Inf}(\omega)$ that occur *infinitely often* in a play ω. Given a target set $T \subseteq V$, a play ω belongs to the *Büchi objective* $\mathrm{Büchi}(T)$ iff $\mathrm{Inf}(\omega) \cap T \neq \emptyset$. For the complementary *co-Büchi objective* we have $\omega \in \mathrm{coBüchi}(T)$ iff $\mathrm{Inf}(\omega) \cap T = \emptyset$. A *generalized* (*or conjunctive*) *Büchi objective* is specified by a set of k target sets T_ℓ for $1 \leq \ell \leq k$ and is satisfied for a play ω iff $\mathrm{Inf}(\omega) \cap T_\ell \neq \emptyset$ for *all* $1 \leq \ell \leq k$. Its complementary objective is the *disjunctive co-Büchi objective* that is satisfied iff $\mathrm{Inf}(\omega) \cap T_\ell = \emptyset$ for *one of* the k target sets. A *generalized reactivity-1* (GR(1)) objective is specified by two generalized Büchi objectives, $\bigwedge_{t=1}^{k_1} \mathrm{Büchi}(L_t)$ and $\bigwedge_{\ell=1}^{k_2} \mathrm{Büchi}(U_\ell)$, and is satisfied if whenever the first generalized Büchi objective holds, then also the second generalized Büchi objective holds; in other words, either $\bigvee_{t=1}^{k_1} \mathrm{coBüchi}(L_t)$ holds, or $\bigwedge_{\ell=1}^{k_2} \mathrm{Büchi}(U_\ell)$ holds.

All the games in this paper will be given by a game graph $\mathcal{G}$ and an objective ψ for player 1 (player 2 has the complementary objective $\Omega \setminus \psi$).

Winning strategies and sets. A strategy σ is winning for player p at a start vertex v if the resulting play is winning for player p irrespective of the strategy of his opponent, player $\bar{p}$, i.e., $\omega(v, \sigma, \pi) \in \psi$ for all π. A vertex v belongs to the *winning set* W_p of player p if player p has a winning strategy from v. Every vertex is winning for exactly one of the two players [50]. When required for explicit reference of a specific game graph $\mathcal{G}$ and objective ψ, we use $W_p(\mathcal{G}, \psi)$ to refer to the winning sets.

Closed sets and attractors. A set $U \subseteq V$ is *p-closed* (in $\mathcal{G}$) if for all p-vertices u in U we have $Out(u) \subseteq U$ and for all $\bar{p}$-vertices v in U there exists a vertex $w \in Out(v) \cap U$. Note that player $\bar{p}$ can ensure that a play that currently ends in a p-closed set never leaves the p-closed set against any strategy of player p by choosing an edge (v, w) with $w \in Out(v) \cap U$ whenever the current vertex v is in $U \cap V_{\bar{p}}$ [62]. Given a game graph $\mathcal{G}$ and a p-closed set U, we denote by $\mathcal{G}[U]$ the game graph induced by the set of vertices U. Note that given that in $\mathcal{G}$ each vertex has at least one outgoing edge, the same property holds for $\mathcal{G}[U]$. We further use the shortcut $\mathcal{G} \setminus X$ to denote $\mathcal{G}[V \setminus X]$.

In a game graph $\mathcal{G}$, a *p-attractor* $Attr_p(\mathcal{G}, U)$ of a set $U \subseteq V$ is the set of vertices from which player p has a strategy to reach U against all strategies of player $\bar{p}$ [62]. We have that $U \subseteq Attr_p(\mathcal{G}, U)$. A p-attractor can be constructed inductively as follows: Let $R_0 = U$; and for all $j \geq 0$ let $R_{j+1} = R_j \cup \{v \in V_p \mid Out(v) \cap R_j \neq \emptyset\} \cup \{v \in V_{\bar{p}} \mid Out(v) \subseteq R_j\}$. Then $Attr_p(\mathcal{G}, U) = \bigcup_{j \geq 0} R_j$. The computation of attractors can be done in linear time [11, 39].

Dominions. A set of vertices $D \subseteq V$ is a player-p *dominion* if $D \neq \emptyset$ and player p has a winning strategy from every vertex in D that also ensures only vertices in D are visited. The notion of dominions was introduced by [42]. Note that a player-p dominion is also a $\bar{p}$-closed set and the p-attractor of a player-p dominion is again a player-p dominion.

▶ **Lemma 1.** *The following assertions hold for game graphs $\mathcal{G}$ where each vertex has at least one outgoing edge. The assertions referring to winning sets hold for graph games with prefix independent objectives. Let $X \subseteq V$.*

1. *The set $V \setminus Attr_p(\mathcal{G}, X)$ is p-closed on $\mathcal{G}$ [62, Lemma 4].*
2. *Let X be p-closed on $\mathcal{G}$. Then $W_{\bar{p}}(\mathcal{G}[X]) \subseteq W_{\bar{p}}(\mathcal{G})$ [42, Lemma 4.4].*
3. *Let X be a subset of the winning set $W_p(\mathcal{G})$ of player p and let A be its p-attractor $Attr_p(\mathcal{G}, X)$. Then the winning set $W_p(\mathcal{G})$ of the player p is the union of A and the winning set $W_p(\mathcal{G}[V \setminus A])$, and the winning set $W_{\bar{p}}(\mathcal{G})$ of the opponent $\bar{p}$ is equal to $W_{\bar{p}}(\mathcal{G}[V \setminus A])$ [42, Lemma 4.5].*

2.2 Conjectured Lower Bounds

While classical complexity results are based on complexity-theoretical assumptions about relationships between complexity classes, e.g., $\mathsf{P} \neq \mathsf{NP}$, polynomial lower bounds are often based on widely believed, conjectured lower bounds about well studied algorithmic problems. We next discuss the popular conjectures that will be the basis for our lower bounds.

First, we consider conjectures on Boolean matrix multiplication [58, 3] and triangle detection [3] in graphs, which build the basis for our lower bounds on dense graphs. A triangle in a graph is a triple x, y, z of vertices such that $(x, y), (y, z), (z, x) \in E$.

▶ **Conjecture 2** (Combinatorial Boolean Matrix Multiplication Conjecture (BMM)). *There is no $O(n^{3-\varepsilon})$ time combinatorial algorithm for computing the Boolean product of two $n \times n$ matrices for any $\varepsilon > 0$.*

▶ **Conjecture 3** (Strong Triangle Conjecture (STC)). *There is no $O(n^{3-\varepsilon})$ time combinatorial algorithm that can detect whether a graph contains a triangle for any $\varepsilon > 0$.*

BMM is equivalent to STC [58]. A weaker assumption, without the restriction to combinatorial algorithms, is that detecting a triangle in a graph takes super-linear time.

Second, we consider the Strong Exponential Time Hypothesis [40, 20] and the Orthogonal Vectors Conjecture [6], the former dealing with satisfiability in propositional logic and the latter with the *Orthogonal Vectors Problem*.

The Orthogonal Vectors Problem (OV). Given two sets S_1, S_2 of d-bit vectors with $|S_i| \leq N$ and $d \in \Theta(\log N)$, are there $u \in S_1$ and $v \in S_2$ such that $\sum_{i=1}^{d} u_i \cdot v_i = 0$?

▶ **Conjecture 4** (Strong Exponential Time Hypothesis (SETH)). *For each $\varepsilon > 0$ there is a k such that k-CNF-SAT on n variables and m clauses cannot be solved in time $O(2^{(1-\varepsilon)n} \operatorname{poly}(m))$.*

▶ **Conjecture 5** (Orthogonal Vectors Conjecture (OVC)). *There is no $O(N^{2-\varepsilon})$ time algorithm for the Orthogonal Vectors Problem for any $\varepsilon > 0$.*

SETH implies OVC [59], i.e., whenever a problem is hard assuming OVC, it is also hard when assuming SETH. Hence, it is preferable to use OVC for proving lower bounds. Finally, to the best of our knowledge, no such relations between the former two conjectures and the latter two conjectures are known.

▶ Remark. The conjectures that no *polynomial* improvements over the best known running times are possible do not exclude improvements by sub-polynomial factors such as poly-logarithmic factors or factors of, e.g., $2^{\sqrt{\log n}}$ as in [60].

3 Algorithms for Generalized Büchi Games

For generalized Büchi games we first present the basic algorithm that follows from the results of [33, 51, 62]. The basic algorithm runs in time $O(knm)$. We then improve it to an $O(k \cdot n^2)$-time algorithm by exploiting ideas from the $O(n^2)$-time algorithm for Büchi games in [25]. The basic algorithm is fast for instances where one Büchi set, say T_1, is small, i.e., the algorithm runs in time $O(k \cdot b_1 \cdot m)$ time, where $b_1 = |T_1|$. Generalized Büchi games can also be solved via a reduction to Büchi games [13], which yields an $O(k^2 n^2)$ time algorithm when combined with the $O(n^2)$-time Büchi algorithm [25].

Our algorithms iteratively identify sets of vertices that are winning for player 2, i.e., player-2 dominions, and remove them from the graph. We denote the sets in the jth-iteration with superscript j, in particular $\mathcal{G}^1 = \mathcal{G}$, where $\mathcal{G}$ is the input game graph, G^j is the graph of $\mathcal{G}^j$, V^j is the vertex set of G^j, and $T_\ell^j = V^j \cap T_\ell$. We also use $\{T_\ell^j\}$ to denote the list of Büchi sets $(T_1^j, T_2^j, \ldots, T_k^j)$, in particular when updating all the sets in a uniform way.

Algorithm GenBuchiGame: Algorithm for Generalized Büchi Games

Input : Game graph $\mathcal{G} = ((V, E), (V_1, V_2))$ and objective $\bigwedge_{1 \le \ell \le k} \text{Büchi}(T_\ell)$
Output : Winning set of player 1

1 $\mathcal{G}^1 \leftarrow \mathcal{G}$; $\{T_\ell^1\} \leftarrow \{T_\ell\}$; $j \leftarrow 0$
2 **repeat**
3 $j \leftarrow j + 1$
4 **for** $i \leftarrow 1$ **to** $\lceil \log_2 n \rceil$ **do**
5 construct G_i^j
6 $Z_i^j \leftarrow \{v \in V_2 \mid \mathit{Outdeg}(G_i^j, v) = 0\} \cup \{v \in V_1 \mid \mathit{Outdeg}(G_i^j, v) > 2^i\}$
7 **for** $1 \le \ell \le k$ **do**
8 $Y_{\ell,i}^j \leftarrow \mathit{Attr}_1(G_i^j, T_\ell^j \cup Z_i^j)$
9 $S^j \leftarrow V^j \setminus Y_{\ell,i}^j$
10 **if** $S^j \neq \emptyset$ **then** player 2 dominion found, continue with line 11
11 $D^j \leftarrow \mathit{Attr}_2(G^j, S^j)$
12 $\mathcal{G}^{j+1} \leftarrow \mathcal{G}^j \setminus D^j$; $\{T_\ell^{j+1}\} \leftarrow \{T_\ell^j \setminus D^j\}$
13 **until** $D^j = \emptyset$
14 **return** V^j

Basic Algorithm. For each set U that is closed for player 1 we have that from each vertex $u \in U$ player 2 has a strategy to ensure that the play never leaves U [62]. Thus, if there is a Büchi set T_ℓ with $T_\ell \cap U = \emptyset$, then the set U is a player-2 dominion. Moreover, if U is a player-2 dominion, also the attractor $\mathit{Attr}_2(\mathcal{G}, U)$ of U is a player-2 dominion. The basic algorithm proceeds as follows. It iteratively computes vertex sets S^j closed for player 1 that do not intersect with one of the Büchi sets. If such a player-2 dominion S^j is found, then all vertices of $\mathit{Attr}_2(\mathcal{G}^j, S^j)$ are marked as winning for player 2 and removed from the game graph; the remaining game graph is denoted by $\mathcal{G}^{j+1}$. To find a player-2 dominion S^j, for each $1 \le \ell \le k$ the attractor $Y_\ell^j = \mathit{Attr}_1(\mathcal{G}^j, T_\ell^j)$ of the Büchi set T_ℓ^j is determined. If for some ℓ the complement of Y_ℓ^j is not empty, then we assign $S^j = V^j \setminus Y_\ell^j$ for the smallest such ℓ. The algorithm terminates if in some iteration j for each $1 \le \ell \le k$ the attractor Y_ℓ^j contains all vertices of V^j. In this case the set V^j is returned as the winning set of player 1. The winning strategy of player 1 from these vertices is then a combination of the attractor strategies to the sets T_ℓ^j.

▶ **Theorem 6.** *The basic algorithm for generalized Büchi games computes the winning set for player 1 in $O(k \cdot \min_{1 \le \ell \le k} b_\ell \cdot m)$ time, where $b_\ell = |T_\ell|$, and thus also in $O(knm)$ time.*

Our Improved Algorithm. The $O(k \cdot n^2)$-time Algorithm GenBuchiGame for generalized Büchi games combines the basic algorithm described above with the method used for the $O(n^2)$-time Büchi game algorithm [26], called *hierarchical graph decomposition* [37]. The hierarchical graph decomposition defines for a directed graph $G = (V, E)$ and integers $1 \le i \le \lceil \log_2 n \rceil$ the graphs $G_i = (V, E_i)$. Assume the incoming edges of each vertex in G are given in some fixed order in which first the edges from vertices of V_2 and then the edges from vertices of V_1 are listed. The set of edges E_i contains all the outgoing edges of each $v \in V$ with $\mathit{Outdeg}(G, v) \le 2^i$ and the first 2^i incoming edges of each vertex. Note that $G = G_{\lceil \log_2 n \rceil}$ and $|E_i| \in O(n \cdot 2^i)$. The runtime analysis uses that we can identify small player-2 dominions (i.e., player-1 closed sets that do not intersect one of the target sets) that contain $O(2^i)$ vertices by only looking at G_i. The algorithm first searches for such a set S^j in G_i for $i = 1$ and each target set and then increases i until the search is successful. In this way the time spent for the search is proportional to $k \cdot n$ times the number of vertices in the found

dominion, which yields a total runtime bound of $O(k \cdot n^2)$. To obtain the $O(k \cdot n^2)$ running time bound, it is crucial to put the loop over the different Büchi sets as the innermost part of the algorithm. Given a game graph $\mathcal{G} = (G, (V_1, V_2))$, we denote by $\mathcal{G}_i$ the game graph where G was replaced by G_i from the hierarchical graph decomposition, i.e., $\mathcal{G}_i = (G_i, (V_1, V_2))$.

Properties of hierarchical graph decomposition. The following lemma identifies two essential properties of the hierarchical graph decomposition. The first is crucial for correctness: When searching in $\mathcal{G}_i$ for a player-1 closed set that does not contain one of the target sets, we can ensure that such a set is also closed for player 1 in $\mathcal{G}$ by excluding certain vertices that are missing outgoing edges in $\mathcal{G}_i$ from the search. The second is crucial for the runtime: Whenever the basic algorithm would remove (i.e., identify as winning for player 2) a set with at most 2^i vertices, then we can identify this set also by searching in $\mathcal{G}_i$ instead of $\mathcal{G}$.

▶ **Lemma 7.** *Let $\mathcal{G} = (G = (V, E), (V_1, V_2))$ be a game graph and $\{G_i\}$ its hierarchical graph decomposition. For $1 \leq i \leq \lceil \log_2 n \rceil$ let Z_i be the set consisting of the player 2 vertices that have no outgoing edge in $\mathcal{G}_i$ and the player 1 vertices with $> 2^i$ outgoing edges in $\mathcal{G}$.*

1. *If a set $S \subseteq V \setminus Z_i$ is closed for player 1 in $\mathcal{G}_i$, then S is closed for player 1 in $\mathcal{G}$.*
2. *If a set $S \subseteq V$ is closed for player 1 in $\mathcal{G}$ and $|Attr_2(\mathcal{G}, S)| \leq 2^i$, then (i) $\mathcal{G}_i[S] = \mathcal{G}[S]$, (ii) the set S is in $V \setminus Z_i$, and (iii) S is closed for player 1 in $\mathcal{G}_i$.*

With the above lemma we can show that whenever a player-2 dominion is found in $\mathcal{G}_i$ but not in $\mathcal{G}_{i-1}$, then at least $\Omega(2^i)$ vertices are removed from the maintained game graph. Together with a runtime bound of $O(k \cdot 2^i \cdot n)$ for the search, this yields a total runtime of $O(k \cdot n)$ per vertex, i.e., time $O(k \cdot n^2)$ in total.

▶ **Theorem 8.** *Algorithm* GenBuchiGame *computes the winning set of player 1 in a generalized Büchi game in $O(k \cdot n^2)$ time.*

4 Conditional Lower bounds for Generalized Büchi Games

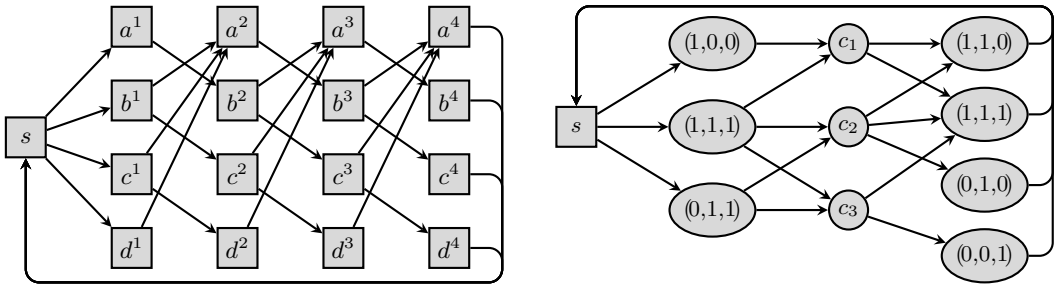
In this section we present two conditional lower bounds, one for dense graphs ($m = \Theta(n^2)$) based on STC & BMM, and one for sparse graphs ($m = \Theta(n^{1+o(1)})$) based on OVC & SETH.

▶ **Theorem 9.** *There is no combinatorial $O(n^{3-\epsilon})$ or $O((k \cdot n^2)^{1-\epsilon})$-time algorithm (for any $\epsilon > 0$) for generalized Büchi games under Conjecture 3 (i.e., unless STC & BMM fail).*

The result can be obtained from a reduction from triangle detection to disjunctive co-Büchi objectives on graphs in [22], and we present the reduction in terms of game graphs below and illustrate it on an example in Figure 1a.

▶ **Reduction 10.** *Given a graph $G = (V, E)$ (for triangle detection), we build a game graph $\mathcal{G}' = (G = (V', E'), (V_1, V_2))$ (for generalized Büchi objectives) as follows. As vertices V' we have four copies V^1, V^2, V^3, V^4 of V and a vertex s. A vertex $v^i \in V^i, i \in \{1, 2, 3\}$ has an edge to a vertex $u^{i+1} \in V^{i+1}$ iff $(v, u) \in E$. Moreover, s has an edge to all vertices of V^1 and all vertices of V^4 have an edge to s. All the vertices are owned by player 2, i.e., $V_1 = \emptyset$ and $V_2 = V$. Finally, we consider the generalized Büchi objective $\bigwedge_{v \in V} Büchi(T_v)$, with $T_v = (V^1 \setminus \{v^1\}) \cup (V^4 \setminus \{v^4\})$.*

We have that there is a triangle in the graph G if and only if the vertex s is winning for player 2 in the generalized Büchi game on $\mathcal{G}'$. Notice that the sets T_v in the above reduction are of linear size but can be reduced to logarithmic size using a construction from [22]. Next we present an $\Omega(m^{2-o(1)})$ lower bound for generalized Büchi objectives.



(a) Reduction 10 applied to $G = (\{a, b, c, d\}, \{(a, b), (b, a), (b, c), (c, a), (c, d), (d, a)\})$.

(b) Reduction 12 applied to $S_1 = \{(1, 0, 0), (1, 1, 1), (0, 1, 1)\}$ and $S_2 = \{(1, 1, 0), (1, 1, 1), (0, 1, 0), (0, 0, 1)\}$.

Figure 1 Illustration of Reductions 10 and 12.

▶ **Theorem 11.** *There is no $O(m^{2-\epsilon})$ or $O(\min_{1 \leq i \leq k} b_i \cdot (k \cdot m)^{1-\epsilon})$-time algorithm (for any $\epsilon > 0$) for generalized Büchi games under Conjecture 5 (i.e., unless OVC & SETH fail).*

The above theorem is by a linear time reduction from OV provided below (cf. Figure 1b).

▶ **Reduction 12.** *Given two sets S_1, S_2 of d-dimensional vectors, we build the following game graph. The vertices V of the graph G are given by a start vertex s, vertices S_1 and S_2 representing the sets of vectors, and vertices $\mathcal{C} = \{c_i \mid 1 \leq i \leq d\}$ representing the coordinates. The edges E of G are defined as follows: the start vertex s has an edge to every vertex of S_1 and every vertex of S_2 has an edge to s; further for each $x \in S_1$ there is an edge to $c_i \in \mathcal{C}$ iff $x_i = 1$ and for each $y \in S_2$ there is an edge from $c_i \in \mathcal{C}$ iff $y_i = 1$. The set of vertices V is partitioned into player 1 vertices $V_1 = S_1 \cup S_2 \cup \mathcal{C}$ and player 2 vertices $V_2 = \{s\}$. Finally, the generalized Büchi objective is given by $\bigwedge_{v \in S_2} \text{Büchi}(T_v)$ with $T_v = \{v\}$.*

▶ **Lemma 13.** *Given two sets S_1, S_2 of d-dimensional vectors and the corresponding graph game $\mathcal{G}$ given by Reduction 12 with $T_v = \{v\}$ for $v \in S_2$, (1) there exist orthogonal vectors $x \in S_1$ and $y \in S_2$ if and only if (2) $s \notin W_1(\mathcal{G}, \bigwedge_{v \in S_2} \text{Büchi}(T_v))$.*

Proof. W.l.o.g. we assume that the 1-vector, i.e., the vector with all coordinates being 1, is contained in S_2 (adding the 1-vector does not change the result of the OV instance), which guarantees that each vertex $c \in \mathcal{C}$ in the construction below has at least 1 outgoing edge. Then a play in the game graph $\mathcal{G}$ proceeds as follows. Starting from s, player 2 chooses a vertex $x \in S_1$; then player 1 first picks a vertex $c \in \mathcal{C}$ and then a vertex $y \in S_2$; then the play goes back to s (at each $y \in S_2$ player 1 has only this choice), starting another cycle of the play. (1)⇒(2): Assume there are orthogonal vectors $x \in S_1$ and $y \in S_2$. Now player 2 can satisfy coBüchi (T_y) by simply going to x whenever the play is in s. Then player 1 can choose some adjacent $c \in \mathcal{C}$ and then some adjacent vertex in S_2, but as x and y are orthogonal, this c is not connected to y. Thus the play will never visit y. (2)⇒(1): By the fact that $W_1 = V \setminus W_2$ [50] we have that (2) is equivalent to $s \in W_2(\mathcal{G}, \bigwedge_{v \in S_2} \text{Büchi}(T_v))$. Assume $s \in W_2(\mathcal{G}, \bigwedge_{v \in S_2} \text{Büchi}(T_v))$ and consider a corresponding strategy for player 2 that satisfies $\bigvee_{v \in S_2} \text{coBüchi}(T_v)$. Notice that the graph is such that player 2 has to visit at least one of the vertices v in S_1 infinitely often. Moreover, for such a vertex v then player 1 can visit all vertices $v' \in S_2$ that correspond to non-orthogonal vectors infinitely often. That is, if v has no orthogonal vector, player 1 can satisfy all the Büchi constraints, a contradiction to our assumption that $s \in W_2(\mathcal{G}, \bigwedge_{v \in S_2} \text{Büchi}(T_v))$. Thus there must be a vector $x \in S_1$ such that there exists a vector $y \in S_2$ that is orthogonal to x. ◀

Let $N = \max(|S_1|, |S_2|)$. The number of vertices in the game graph, constructed by Reduction 12, is $O(N)$, the number of edges m is $O(N \log N)$ (recall that $d \in O(\log N)$), we have $k \in \Theta(N)$ target sets, each of size 1, and the construction can be performed in $O(N \log N)$ time. Thus, if we would have an $O(m^{2-\epsilon})$ or $O(\min_{1 \leq i \leq k} b_i \cdot (k \cdot m)^{1-\epsilon})$ time algorithm for any $\epsilon > 0$, we would immediately get an $O(N^{2-\epsilon})$ algorithm for OV, which contradicts OVC (and thus SETH).

▶ Remark. Notice that the lower bounds apply to instances with $k \in \Theta(n^c)$ for arbitrary $0 < c \leq 1$, although the reductions produce graphs with $k \in \Theta(n)$. This is because of the specific type of the constructed instances, for which each $O((k \cdot f(n,m))^{1-\epsilon})$-time algorithm for $k \in \Theta(n^c)$ also implies an $O((k \cdot f(n,m))^{1-\epsilon})$-time algorithm for $k \in \Theta(n)$.

5 Generalized Reactivity-1 Games

GR(1) games deal with an objective of the form $\bigwedge_{t=1}^{k_1} \text{Büchi}(L_t) \to \bigwedge_{\ell=1}^{k_2} \text{Büchi}(U_\ell)$ and can be solved in $O(k_1 k_2 \cdot m \cdot n)$ time [43] with an extension of the progress measure algorithm of [41] and in $O((k_1 k_2 \cdot n)^{2.5})$ time by combining the reduction to one-pair Streett objectives by [13] with the algorithm of [27]. In this section we develop an $O(k_1 k_2 \cdot n^{2.5})$-time algorithm by modifying the algorithm of [43] to compute dominions. We further use our $O(k \cdot n^2)$-time algorithm for generalized Büchi games with $k = k_1$ as a subroutine.

We first describe a basic, direct algorithm for GR(1) games that is based on repeatedly identifying player-2 dominions in generalized Büchi games. We then show how the progress measure algorithm of [43] can be modified to identify player-2 dominions in generalized Büchi games with k_1 Büchi objectives in time proportional to $k_1 \cdot m$ times the size of the dominion. In the $O(k_1 k_2 \cdot n^{2.5})$-time algorithm we use the modified progress measure algorithm in combination with the hierarchical graph decomposition of [26, 27] to identify dominions that contain up to $\sqrt{n}$ vertices and our $O(k_1 \cdot n^2)$-time algorithm for generalized Büchi games to identify dominions with more than $\sqrt{n}$ vertices. Each time we search for a dominion we might have to consider k_2 different subgraphs.

We denote the sets in the jth-iteration of our algorithms with superscript j, in particular $\mathcal{G}^1 = \mathcal{G}$, where $\mathcal{G}$ is the input game graph, G^j is the graph of $\mathcal{G}^j$, V^j is the vertex set of G^j, $V_1^j = V_1 \cap V^j$, $V_2^j = V_2 \cap V^j$, $L_t^j = L_t \cap V^j$, and $U_\ell^j = U_\ell \cap V^j$.

Basic Algorithm. Similar to generalized Büchi games, the basic algorithm for GR(1) games identifies a player-2 dominion S^j, removes the dominion and its player-2 attractor D^j from the graph, and recurses on the remaining game graph $\mathcal{G}^{j+1} = \mathcal{G}^j \setminus D^j$. If no player-2 dominion is found, the remaining set of vertices V^j is returned as the winning set of player 1. Given the set S^j is indeed a player-2 dominion, the correctness of this approach follows from Lemma 1(3). A player-2 dominion in $\mathcal{G}^j$ is identified as follows: For each $1 \leq \ell \leq k_2$ first the player-1 attractor Y_ℓ^j of U_ℓ^j is temporarily removed from the graph. Then a generalized Büchi game with target sets $L_1^j, \ldots, L_{k_1}^j$ is solved on $\overline{\mathcal{G}^j \setminus Y_\ell^j}$. The generalized Büchi player in this game corresponds to player 2 in the GR(1) game and his winning set to a player-2 dominion in the GR(1) game. Note that $V^j \setminus Y_\ell^j$ is player-1 closed and does not contain U_ℓ^j. Thus if in the game induced by the vertices of $V^j \setminus Y_\ell^j$ player 2 can win w.r.t. the generalized Büchi objective $\bigwedge_{t=1}^{k_1} \text{Büchi}(L_t^j)$, then these vertices form a player-2 dominion in the GR(1) game. Further, we can show that when a player-2 dominion in the GR(1) games on $\mathcal{G}^j$ exists, then for one of the sets U_ℓ^j the winning set of the generalized Büchi game on $\overline{\mathcal{G}^j \setminus Y_\ell^j}$ is non-empty; otherwise we can construct a winning strategy of player 1 for the GR(1) game on $\mathcal{G}^j$. Note that this algorithm computes a player-2 dominion $O(k_2 \cdot n)$ often using our $O(k_1 \cdot n^2)$-time generalized Büchi Algorithm GenBuchiGame.

▶ **Theorem 14.** *The basic algorithm for GR(1) games computes the winning set for player 1 in $O(k_1 \cdot k_2 \cdot n^3)$ time.*

Improved Algorithm. The overall structure of our $O(k_1 k_2 \cdot n^{2.5})$-time algorithm for GR(1) games (see Algorithm GR(1)Game) is the same as for the basic algorithm: We search for a player-2 dominion S^j and if one is found, then its player-2 attractor D^j is determined and removed from the current game graph $\mathcal{G}^j$ (with $\mathcal{G}^1 = \mathcal{G}$) to create the game graph for the next iteration, $\mathcal{G}^{j+1}$. If no player-2 dominion exists, then the remaining vertices are returned as the winning set of player 1. The difference to the basic algorithm lies in the way player-2 dominions are searched. Two different procedures are used for this purpose: First we search for "small" dominions with the subroutine `kGenBüchiDominion`. If no small dominions exist, then we search for player-2 dominions as in the basic algorithm. The guarantee that we find a "large" dominion allows us to bound the number of times the second case can happen.

Progress Measure Algorithm. In the Procedure `kGenBüchiDominion` we use a subroutine that finds in a generalized Büchi game all dominions of the generalized Büchi player that have size at most h (where h is an input parameter). This subroutine is based on a so-called *progress measure* for generalized Büchi objectives which is a special instance of the progress measure for GR(1) objectives presented in [43, Section 3.1], which itself is based on [41]. We modify the progress measure to efficiently identify dominions of size at most h (instead of computing the whole winning set) by restricting the range of allowed values for the progress measure functions similar to [56]. Finally, we give an $O(k \cdot h \cdot m)$-time algorithm for computing the progress measure functions based on [34, 43] (details are provided in the full version).

▶ **Theorem 15.** *For a game graph $\mathcal{G}$ and objective $\psi = \bigwedge_{1 \le \ell \le k} Büchi(T_\ell)$, there is an $O(k \cdot h \cdot m)$ time procedure* `GenBüchiProgressMeasure`$(\mathcal{G}, \psi, h)$ *that either returns a player-1 dominion or the empty set, and, if there is at least one player-1 dominion of size $\le h$ then returns a player-1 dominion containing all player-1 dominions of size $\le h$.*

Procedure `kGenBüchiDominion`**.** The procedure `kGenBüchiDominion` searches for player-2 dominions in the GR(1) game, and returns some dominion if there exists a dominion D with $|Attr_2(\mathcal{G}, D)| \le h_{\max}$. To this end we again consider generalized Büchi games with target sets $L_1^j, \ldots, L_{k_1}^j$, where the generalized Büchi player corresponds to player 2 in the GR(1) game. We use the same hierarchical graph decomposition as for Algorithm GenBuchiGame: Let the incoming edges of each vertex be ordered such that the edges from vertices of V_2 come first; for a given game graph $\mathcal{G}^j$ the graph G_i^j contains all vertices of $\mathcal{G}^j$, for each vertex its first 2^i incoming edges, and for each vertex with outdegree at most 2^i all its outgoing edges. The set Z_i^j contains all vertices of V_1 with outdegree larger than 2^i and all vertices of V_2 that have no outgoing edge in G_i^j. We start with $i = 1$ and increase i by one as long as no dominion was found. For a given i we perform the following operations for each $1 \le \ell \le k_2$: First the player 1 attractor $Y_{i,\ell}^j$ of $U_\ell^j \cup Z_i^j$ is determined. Then we search for player-1 dominions on $\overline{\mathcal{G}_i^j \setminus Y_{i,\ell}^j}$ w.r.t. the objective $\bigwedge_{t=1}^{k_1} \text{Büchi}\,(L_t)$ with the generalized Büchi progress measure algorithm and parameter $h = 2^i$, i.e., by Theorem 15 the progress measure algorithm returns all generalized Büchi dominions in $\overline{\mathcal{G}_i^j \setminus Y_{i,\ell}^j}$ of size at most h.

The following lemma shows how the properties of the hierarchical graph decomposition extend to GR(1) games. The first part is crucial for correctness: Every non-empty set found by the progress measure algorithm on $\overline{\mathcal{G}_i^j \setminus Y_{i,\ell}^j}$ for some i and ℓ is indeed a player-2 dominion in the GR(1) game. The second part is crucial for the runtime argument: Whenever the basic algorithm for GR(1) games would identify a player-2 dominion D with $|Attr_2(\mathcal{G}, D)| \le 2^i$, then D is also a generalized Büchi dominion in $\overline{\mathcal{G}_i^j \setminus Y_{i,\ell}^j}$ for some ℓ.

Algorithm GR(1)Game: Algorithm for GR(1) Games

Input : Game graph $\mathcal{G} = ((V, E), (V_1, V_2))$, Obj. $\bigwedge_{t=1}^{k_1} \text{Büchi}\,(L_t) \to \bigwedge_{\ell=1}^{k_2} \text{Büchi}\,(U_\ell)$
Output: Winning set of player 1

$\mathcal{G}^1 \leftarrow \mathcal{G}$; $\{U_\ell^1\} \leftarrow \{U_\ell\}$; $\{L_t^1\} \leftarrow \{L_t\}$
$j \leftarrow 0$
repeat
  $j \leftarrow j + 1$
  $S^j \leftarrow$ `kGenBüchiDominion`$(\mathcal{G}^j, \{U_\ell^j\}, \{L_t^j\}, \sqrt{n})$
  if $S^j = \emptyset$ **then**
    for $1 \le \ell \le k_2$ **do**
      $Y_\ell^j \leftarrow \mathit{Attr}_1(\mathcal{G}^j, U_\ell^j)$
      $S^j \leftarrow$ `GenBüchiGame`$(\overline{\mathcal{G}^j \setminus Y_\ell^j}, \bigwedge_{\ell=1}^{k_1} \text{Büchi}\left(L_t^j \setminus Y_\ell^j\right))$
      if $S^j \neq \emptyset$ **then break**
  $D^j \leftarrow \mathit{Attr}_2(\mathcal{G}^j, S^j)$
  $\mathcal{G}^{j+1} \leftarrow \mathcal{G}^j \setminus D^j$; $\{U_\ell^{j+1}\} \leftarrow \{U_\ell^j \setminus D^j\}$; $\{L_t^{j+1}\} \leftarrow \{L_t^j \setminus D^j\}$
until $D^j = \emptyset$
return V^j

Procedure `kGenBüchiDominion`$(\mathcal{G}^j, \{U_\ell^j\}, \{L_t^j\}, h_{max})$
  for $i \leftarrow 1$ **to** $\lceil \log_2(h_{max}) \rceil$ **do**
    construct G_i^j
    $Z_i^j \leftarrow \{v \in V_2 \mid \mathit{Outdeg}(G_i^j, v) = 0\} \cup \{v \in V_1 \mid \mathit{Outdeg}(G_i^j, v) > 2^i\}$
    for $1 \le \ell \le k_2$ **do**
      $Y_{i,\ell}^j \leftarrow \mathit{Attr}_1(\mathcal{G}_i^j, U_\ell^j \cup Z_i^j)$
      $X_{i,\ell}^j \leftarrow$ `GenBüchiProgressMeasure`$(\overline{\mathcal{G}_i^j \setminus Y_{i,\ell}^j}, \bigwedge_{\ell=1}^{k_1} \text{Büchi}\left(L_t^j \setminus Y_{i,\ell}^j\right), 2^i)$
      if $X_{i,\ell}^j \neq \emptyset$ **then return** $X_{i,\ell}^j$
  return $\emptyset$

▶ **Lemma 16.** *Let the notation be as in Algorithm* GR(1)Game.

1. *Every $X_{i,\ell}^j \neq \emptyset$ is a player-2 dominion in the GR(1) game on $\mathcal{G}^j$ with $X_{i,\ell}^j \cap U_\ell^j = \emptyset$.*
2. *If for player 2 there exists in $\mathcal{G}^j$ a dominion D w.r.t. the generalized Büchi objective $\bigwedge_{t=1}^{k_1} \mathit{Büchi}(L_t^j)$ such that $D \cap U_\ell^j = \emptyset$ for some $1 \le \ell \le k_2$ and $|\mathit{Attr}_2(\mathcal{G}^j, D)| \le 2^i$, then D is a dominion w.r.t. the generalized Büchi objective $\bigwedge_{t=1}^{k_1} \mathit{Büchi}(L_t^j \setminus Y_{i,\ell}^j)$ in $\mathcal{G}_i^j \setminus Y_{i,\ell}^j$.*

From this we can draw the following two corollaries: (1) When we had to go up to i^* in the graph decomposition to find a dominion, then its attractor has size at least 2^{i^*-1} and (2) when `kGenBüchiDominion` returns an empty set, then all player-2 dominions in the current game graph have more than $h_{\max} = \sqrt{n}$ vertices. In the second case either no player-2 dominion exists or the subsequent call to `GenBüchiGame` returns one with more than $\sqrt{n}$ vertices, which can happen at most $O(\sqrt{n})$ times. Together with (1), this means we can (a) charge the time spent in `kGenBüchiDominion` to the vertices in the dominion identified in this iteration of the repeat-until loop (except for the last iteration) and (b) bound the number of calls to `GenBüchiGame` with $O(\sqrt{n})$.

▶ **Theorem 17.** *Algorithm* GR(1)Game *computes the winning set of player 1 in a GR(1) game in $O(k_1 \cdot k_2 \cdot n^{2.5})$ time.*

6 Conclusion

In this work we present improved algorithms for generalized Büchi and GR(1) objectives, and conditional lower bounds for generalized Büchi objectives. The existing upper bounds

and our conditional lower bounds are tight for (a) for dense graphs, and (b) sparse graphs with constant size target sets. Two interesting open questions are as follows: (1) For sparse graphs with $\theta(n)$ many target sets of size $\theta(n)$ the upper bounds are cubic, whereas the conditional lower bound is quadratic, and closing the gap is an interesting open question. (2) For GR(1) objectives we obtain the conditional lower bounds from generalized Büchi objectives, which are not tight in this case; whether better (conditional) lower bounds can be established also remains open.

References

1 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is Valiant's parser. In *FOCS*, pages 98–117, 2015.

2 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight Hardness Results For LCS and other Sequence Similarity Measures. In *FOCS*, pages 59–78, 2015.

3 Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *FOCS*, pages 434–443, 2014.

4 Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *ICALP 2014, Proceedings, Part I*, pages 39–51, 2014.

5 Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *STOC*, pages 41–50, 2015.

6 Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *SODA*, pages 377–391, 2016.

7 Rajeev Alur and Thomas A. Henzinger. Computer-aided verification, 2004. Unpublished, available at `http://www.cis.upenn.edu/group/cis673/`.

8 Rajeev Alur, Thomas. A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49:672–713, 2002.

9 Rajeev Alur and Salvatore La Torre. Deterministic generators and games for ltl fragments. *ACM Trans. Comput. Log.*, 5(1):1–25, 2004.

10 Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *STOC*, pages 51–58, 2015.

11 Catriel Beeri. On the membership problem for functional and multivalued dependencies in relational databases. *ACM Transactions on Database Systems*, pages 241–259, 1980.

12 Dietmar Berwanger, Anuj Dawar, Paul Hunter, and Stephan Kreutzer. Dag-width and parity games. In *STACS*, pages 524–536, 2006.

13 Roderick Bloem, Krishnendu Chatterjee, Karin Greimel, Thomas A. Henzinger, and Barbara Jobstmann. Robustness in the presence of liveness. In *CAV*, pages 410–424, 2010.

14 Roderick Bloem, Stefan J. Galler, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Martin Weiglhofer. Interactive presentation: Automatic hardware synthesis from specifications: a case study. In *DATE*, pages 1188–1193, 2007.

15 Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *FOCS*, pages 661–670, 2014.

16 Karl Bringmann and Marvin Künnemann. Quadratic Conditional Lower Bounds for String Problems and Dynamic Time Warping. In *FOCS*, pages 79–97, 2015.

17 J. Richard Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6:66–92, 1960.

18 J. Richard Büchi. On a decision method in restricted second-order arithmetic. In E. Nagel, P. Suppes, and A. Tarski, editors, *Proceedings of the First International Congress on Logic, Methodology, and Philosophy of Science 1960*, pages 1–11. Stanford University Press, 1962.

19 J. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the AMS*, 138:295–311, 1969.

20 Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In *IWPEC*, pages 75–85, 2009.

21 Krishnendu Chatterjee, Martin Chmelik, Raghav Gupta, and Ayush Kanodia. Qualitative analysis of pomdps with temporal logic specifications for robotics applications. In *IEEE International Conference on Robotics and Automation, ICRA*, pages 325–330, 2015.

22 Krishnendu Chatterjee, Wolfgang Dvořák, Monika Henzinger, and Veronika Loitzenbauer. Model and objective separation with conditional lower bounds: Disjunction is harder than conjunction. In *LICS*, 2016. To appear, available at `http://arxiv.org/abs/1602.02670`.

23 Krishnendu Chatterjee, Amir Kafshdar Goharshady, Rasmus Ibsen-Jensen, and Andreas Pavlogiannis. Algorithms for algebraic path properties in concurrent systems of constant treewidth components. In *POPL*, pages 733–747, 2016.

24 Krishnendu Chatterjee and Monika Henzinger. Faster and Dynamic Algorithms For Maximal End-Component Decomposition And Related Graph Problems In Probabilistic Verification. In *SODA*, pages 1318–1336, 2011.

25 Krishnendu Chatterjee and Monika Henzinger. An $O(n^2)$ Time Algorithm for Alternating Büchi Games. In *SODA*, pages 1386–1399, 2012.

26 Krishnendu Chatterjee and Monika Henzinger. Efficient and Dynamic Algorithms for Alternating Büchi Games and Maximal End-component Decomposition. *Journal of the ACM*, 61(3):15, 2014.

27 Krishnendu Chatterjee, Monika Henzinger, and Veronika Loitzenbauer. Improved Algorithms for One-Pair and k-Pair Streett Objectives. In *LICS*, pages 269–280, 2015.

28 Krishnendu Chatterjee, Rasmus Ibsen-Jensen, Andreas Pavlogiannis, and Prateesh Goyal. Faster algorithms for algebraic path properties in recursive state machines with constant treewidth. In *POPL*. ACM, 2015.

29 Krishnendu Chatterjee, Marcin Jurdziński, and Thomas A. Henzinger. Simple stochastic parity games. In *CSL*, pages 100–113, 2003.

30 Alonzo Church. Logic, arithmetic, and automata. In *Proceedings of the International Congress of Mathematicians*, pages 23–35. Institut Mittag-Leffler, 1962.

31 Luca de Alfaro and Thomas A. Henzinger. Interface automata. In *FSE'01*, pages 109–120. ACM Press, 2001.

32 David L. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed-independent Circuits*. The MIT Press, 1989.

33 E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy. In *FOCS*, pages 368–377, 1991.

34 Kousha Etessami, Thomas Wilke, and Rebecca A. Schuller. Fair simulation relations, parity games, and state space reduction for büchi automata. *SIAM J. Comput.*, 34(5):1159–1175, 2005.

35 Georgios E. Fainekos, Hadas Kress-Gazit, and George J. Pappas. Temporal logic motion planning for mobile robots. In *IEEE International Conference on Robotics and Automation, ICRA*, pages 2020–2025, 2005.

36 Yashdeep Godhal, Krishnendu Chatterjee, and Thomas A. Henzinger. Synthesis of AMBA AHB from formal specification: A case study. *Journal of Software Tools Technology Transfer*, 2011.

37 Monika Henzinger, Valerie King, and Tandy Warnow. Constructing a Tree from Homeomorphic Subtrees, with Applications to Computational Evolutionary Biology. *Algorithmica*, 24(1):1–13, 1999.

38 Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *STOC*, pages 21–30, 2015.

39 Neil Immerman. Number of quantifiers is better than number of tape cells. *Journal of Computer and System Sciences*, pages 384–406, 1981.

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

41 Marcin Jurdziński. Small Progress Measures for Solving Parity Games. In *STACS*, pages 290–301, 2000.

42 Marcin Jurdziński, Mike Paterson, and Uri Zwick. A Deterministic Subexponential Algorithm for Solving Parity Games. *SIAM Journal on Computing*, 38(4):1519–1532, 2008.

43 Sudeep Juvekar and Nir Piterman. Minimizing generalized büchi automata. In *CAV*, pages 45–58, 2006.

44 Sriram C. Krishnan, Anuj Puri, and Robert K. Brayton. Deterministic Ω automata vis-a-vis deterministic buchi automata. In *ISAAC*, pages 378–386, 1994.

45 Wouter Kuijper and Jaco van de Pol. Computing weakest strategies for safety games of imperfect information. In *TACAS*, pages 92–106, 2009.

46 Orna Kupferman and Moshe Y. Vardi. Freedom, weakness, and determinism: From linear-time to branching-time. In *LICS*, pages 81–92, 1998.

47 Orna Kupferman and Moshe Y. Vardi. From linear time to branching time. *ACM Transactions on Computational Logic*, 6(2):273–294, 2005.

48 François Le Gall. Powers of Tensors and Fast Matrix Multiplication. In *ISSAC*, pages 296–303, 2014.

49 Lillian Lee. Fast context-free grammar parsing requires fast boolean matrix multiplication. *J. ACM*, 49(1):1–15, January 2002.

50 Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975.

51 Robert McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993.

52 Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. In *SODA*, pages 1065–1075, 2010.

53 Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive(1) designs. In *VMCAI*, LNCS 3855, Springer, pages 364–380, 2006.

54 Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190. ACM Press, 1989.

55 P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete-event processes. *SIAM Journal of Control and Optimization*, 25(1):206–230, 1987.

56 Sven Schewe. Solving Parity Games in Big Steps. In *FSTTCS*, pages 449–460, 2007.

57 Mikkel Thorup. All Structured Programs Have Small Tree Width and Good Register Allocation. *Information and Computation*, 142(2):159 – 181, 1998.

58 Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *FOCS*, pages 645–654, 2010.

59 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. Announced at ICALP'04.

60 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *STOC*, pages 664–673, 2014.

61 Ryan Williams. Faster decision of first-order graph properties. In *CSL-LICS*, pages 80:1–80:6, 2014.

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