Algorithms and Conditional Lower Bounds for Planning Problems

Krishnendu Chatterjee
IST Austria
krish.chat@ist.ac.at

Wolfgang Dvořák
TU Wien
dvorak@dbai.tuwien.ac.at

Monika Henzinger
University of Vienna
monika.henzinger@univie.ac.at

Alexander Svozil
University of Vienna
alexander.svozil@univie.ac.at

Abstract

We consider planning problems for graphs, Markov decision processes (MDPs), and games on graphs. While graphs represent the most basic planning model, MDPs represent interaction with nature and games on graphs represent interaction with an adversarial environment. We consider two planning problems where there are k different target sets, and the problems are as follows: (a) the coverage problem asks whether there is a plan for each individual target set; and (b) the sequential target reachability problem asks whether the targets can be reached in sequence. For the coverage problem, we present a linear-time algorithm for graphs, and quadratic conditional lower bound for MDPs and games on graphs. For the sequential target problem, we present a linear-time algorithm for graphs, a sub-quadratic algorithm for MDPs, and a quadratic conditional lower bound for games on graphs. Our results with conditional lower bounds establish (i) model-separation results showing that for the coverage problem MDPs and games on graphs are harder than graphs, and for the sequential reachability problem games on graphs are harder than MDPs and graphs; and (ii) objective-separation results showing that for MDPs the coverage problem is harder than the sequential target problem.

Introduction

Planning models. One of the basic and fundamental algorithmic problems in artificial intelligence is the *planning problem* (LaValle 2006; Russell and Norvig 2010). The classical models in planning are as follows:

- *Graphs.* The most basic planning problems are graph search problems (LaValle 2006; Russell and Norvig 2010).
- *MDPs.* In the presence of interaction with nature, the graph model is extended with probabilities or stochastic transitions, which gives rise to Markov decision processes (MDPs) (Howard 1960; Puterman 1994; Filar and Vrieze 1997; Papadimitriou and Tsitsiklis 1987).
- *Games on graphs.* In the presence of interaction with an adversarial environment, the graph model is extended to AND-OR graphs (or games on graphs) (Mahanti and Bagchi 1985; Hansen and Zilberstein 1998).

Thus graphs, MDPs, and games on graphs are the fundamental models for planning.

Planning objectives. The planning objective represents the goal that the planner seeks to achieve. Some basic planning objectives are as follows:

- *Basic target reachability.* Given a set T of target vertices the planning objective is to reach some target vertex from the starting position.
- *Coverage Objective.* In case of coverage there are k different target sets, namely, $T_1, T_2, \ldots, T_k$, and the planning objective asks whether for each $1 \leq i \leq k$ the basic target reachability with target set T_i can be achieved. The coverage models the following scenarios: Consider that there is a robot or a patroller, and there are k different target locations, and if an event or an attack happens in one of the target locations, then that location must be reached. However, the location of the event or the attack is not known in advance and the planner must be prepared that the target set could be any of the k target sets.
- *Sequential target reachability.* In case of sequential targets there are k different target sets, namely, $T_1, T_2, \ldots, T_k$, and the planning objective asks first to reach T_1, then T_2, then T_3 and so on. This represents the scenario that there is a sequence of tasks that the planner must achieve.

The above are the most natural planning objectives and have been studied in the literature, e.g., in robot planning (Kress-Gazit, Fainekos, and Pappas 2009; Kaelbling, Littman, and Cassandra 1998; Choset 2005).

Planning questions. For the above planning objectives the basic planning questions are as follows: (a) for graphs, the question is whether there exists a plan (or a path) such that the planning objective is satisfied; (b) for MDPs, the basic question is whether there exists a policy such that the planning objective is satisfied almost-surely (i.e., with probability 1); and (c) for games on graphs, the basic question is whether there exists a policy that achieves the objective irrespective of the choices of the adversary. The almost-sure satisfaction for MDPs is also known as the strong cyclic planning in the planning literature (Cimatti et al. 2003), and games on graphs question represent planning in the presence of a worst-case adversary (Mahanti and Bagchi 1985; Hansen and Zilberstein 1998) (aka adversarial planning, strong planning (Maliah et al. 2014), or

Obj/Model	Graphs	MDPs	Games on graphs
Basic target	$O(m)$	$O(m \cdot n^{2/3})$	$O(m)$
Coverage objective	$O(m + \sum_{i=1}^{k} \lvert T_i \rvert)$	$O(m \cdot n^{2/3} + k \cdot m)$ $\boldsymbol{\tilde{\Omega}(k \cdot m)}$ [Thm. 1,2]	$O(k \cdot m)$ $\boldsymbol{\tilde{\Omega}(k \cdot m)}$ [Thm. 3,4]
Sequential target	$\boldsymbol{O(m + \sum_{i=1}^{k} \lvert T_i \rvert)}$ [Thm. 6]	$\boldsymbol{O(m \cdot n^{2/3} + \sum_{i=1}^{k} \lvert T_i \rvert)}$ [Thm. 5]	$O(k \cdot m)$ $\boldsymbol{\tilde{\Omega}(k \cdot m)}$ [Thm. 7,8]

Table 1: Algorithmic bounds where n and m are the number of vertices and edges of the underlying model, and k denotes the number of different target sets. The $\tilde{\Omega}(\cdot)$ bounds are conditional lower bounds (CLBs) under the BMM conjecture and SETH. They establish that polynomial improvements over the given bound are not possible, however, polylogarithmic improvements are not excluded. Note that CLBs are quadratic for $k = \Theta(n)$. The new results are highlighted in boldface.

conformant/contingent planning (Bonet and Geffner 2000; Hoffmann and Brafman 2005; Palacios and Geffner 2007)).

Algorithmic study. In this work, we consider the algorithmic study of the planning questions for the natural planning objectives for graphs, MDPs, and games on graphs. For all the above questions, polynomial-time algorithms exist. When polynomial-time algorithms exist, proving an unconditional lower bound is extremely rare. A new approach in complexity theory aims to establish conditional lower bound (CLB) results based on some well-known conjecture. Two standard conjectures for CLBs are as follows: The (a) *Boolean matrix multiplication (BMM)* conjecture which states that there is no sub-cubic combinatorial algorithm for boolean matrix multiplication; and the (b) *Strong exponential-time hypothesis (SETH)* which states that there is no sub-exponential time algorithm for the SAT problem. Many CLBs have been established based on the above conjectures, e.g., for dynamic graph algorithms, string matching (Abboud and Williams 2014; Bringmann and Künnemann 2015).

Previous results and our contributions. We denote by n and m the number of vertices and edges of the underlying model, and k denotes the number of different target sets. For the basic target reachability problem, while the graphs and games on graphs problem can be solved in linear time (Beeri 1980; Immerman 1981), the current best-known bound for MDPs is $O(m \cdot n^{2/3})$ (Chatterjee and Henzinger 2014; Chatterjee et al. 2016). For the coverage and sequential target reachability, an $O(k \cdot m)$ upper bound follows for graphs and games on graphs, and an $O(m \cdot n^{2/3} + k \cdot m)$ upper bound follows for MDPs. Our contributions are as follows:

1. *Coverage problem:* First, we present an $O(m + \sum_{i=1}^{k} |T_i|)$ time algorithm for graphs; second, we present an $\tilde{\Omega}(k \cdot m)$ lower bound for MDPs and games on graphs, both under the BMM conjecture and the SETH. Note that for graphs our upper bound is linear time, however, if each $|T_i|$ is constant and $k = \theta(n)$, for MDPs and games on graphs the CLB is quadratic.
2. *Sequential target problem:* First, we present an $O(m + \sum_{i=1}^{k} |T_i|)$ time algorithm for graphs; second, we present an $O(m \cdot n^{2/3} + \sum_{i=1}^{k} |T_i|)$ time algorithm for MDPs; and third, we present an $\tilde{\Omega}(k \cdot m)$ lower bound for games on graphs, both under the BMM conjecture and the SETH.

The summary of the results is presented in Table 1. Our most interesting results are the conditional lower bounds for MDPs and game graphs for the coverage problem, the sub-quadratic algorithm for MDPs with sequential targets, and the conditional lower bound for game graphs with sequential targets.

Practical Significance. The sequential reachability and coverage problems we consider are the tasks defined in (Kress-Gazit, Fainekos, and Pappas 2009, Section II. PROBLEM FORMULATION, 3) System Specification), where the problems have been studied for games on graphs (Section IV. DISCRETE SYNTHESIS) and mentioned as future work for MDPs (Section I. INTRODUCTION, A. Related Work). The applications of these problems have been demonstrated in robotics applications. We present a complete algorithmic picture for games on graphs and MDPs, settling open questions related to games and future work mentioned in (Kress-Gazit, Fainekos, and Pappas 2009).

Theoretical Significance. Our results present a very interesting algorithmic picture for the natural planning questions in the fundamental models.

1. First, we establish results showing that some models are harder than others. More precisely,
   - for the basic target problem, the MDP model seems harder than graphs/games on graphs (linear-time algorithm for graphs and games on graphs, and no such algorithms are known for MDPs);
   - for the coverage problem, MDPs, and games on graphs are harder than graphs (linear-time algorithm for graphs and quadratic CLBs for MDPs and games on graphs);
   - for the sequential target problem, games on graphs are harder than MDPs and graphs (linear-time upper bound for graphs and sub-quadratic upper bound for MDPs, whereas quadratic CLB for games on graphs).

   In summary, we establish model-separation results with CLBs: For the coverage problem, MDPs and games on graphs are algorithmically harder than graphs; and for the sequential target problem, games on graphs are algorithmically harder than MDPs and graphs.
2. Second, we also establish objective-separation results. For the model of MDPs consider the different objectives: Both for basic target and sequential target reachability the upper bound is sub-quadratic and in contrast to the coverage problem we establish a quadratic CLB.

Discussion related to other models. In this work, our focus lies on the algorithmic complexity of fundamental plan-

ning problems and we consider *explicit state-space* graphs, MDPs, and games, where the complexities are polynomial. The explicit model and algorithms for it are widely considered: (LaValle 2006)[Chapter 2.1 Discrete Feasible Planning], (Kress-Gazit, Fainekos, and Pappas 2009)[Section IV. DISCRETE SYNTHESIS] and (Chatterjee and Henzinger 2014)[Section 2.1. Definitions. Alternating game graphs.] In other representations such as the factored model, the complexities are higher (NP-complete), and then heuristics are the focus (e.g., (Hansen and Zilberstein 1998)) rather than the algorithmic complexity. Notable exceptions are the work on parameterized complexity of planning problems (see, e.g., (Kronegger, Pfandler, and Pichler 2013)) and Conditional Lower Bounds showing that certain planning problems do not admit subexponential time algorithms (Aghighi et al. 2016; Bäckström and Jonsson 2017).

Preliminaries

Markov Decision Processes (MDPs). A *Markov decision process (MDP)* $P = ((V, E), \langle V_1, V_R \rangle, \delta)$ consists of a finite set of vertices V partitioned into the player-1 vertices V_1 and the random vertices V_R, a finite set of edges $E \subseteq (V \times V)$, and a probabilistic transition function δ. The probabilistic transition function maps every random vertex in V_R to an element of $\mathcal{D}(V)$, where $\mathcal{D}(V)$ is the set of probability distributions over the set of vertices V. A random vertex v has an edge to a vertex $w \in V$, i.e. $(v, w) \in E$ iff $\delta(v)[w] > 0$.

Game Graphs. A game graph $\Gamma = ((V, E), \langle V_1, V_2 \rangle)$ consists of a finite set of vertices V, a finite set of edges E and a partition of the vertices V into player-1 vertices V_1 and the adversarial player-2 vertices V_2.

Graphs. Graphs are a special case of MDPs with $V_R = \emptyset$ as well as special case of game graphs with $V_2 = \emptyset$. Let $Out(v)$ describe the set of successor vertices of v. The set $In(v)$ describes the set of predecessors of the vertex v. More formally $Out(v) = \{w \in V \mid (v, w) \in E\}$ and $In(v) = \{w \in V \mid (w, v) \in E\}$.

Remark 1. *Note that a standard way to define MDPs is to consider finite vertices with actions, and the probabilistic transition function is defined for every vertex and action. In our model, the choice of actions is represented as the choice of edges at player-1 vertices and the probabilistic transition function is represented by the random vertices. This allows us to treat MDPs and game graphs in a uniform way, and graphs can be described easily as a special case of MDPs.*

Plays. A *play* is an infinite sequence $\omega = \langle v_0, v_1, v_2, \ldots \rangle$ of vertices such that each $(v_{i-1}, v_i) \in E$ for all $i \geq 1$. The set of all plays is denoted with Ω. A play is initialized by placing a token on an initial vertex. If the token is on a vertex owned by a player (such as player 1 in MDPs, or player 1 or player 2 in game graphs), then the respective player moves the token along one of the outgoing edges, whereas if the token is at a random vertex $v \in V_R$, then the next vertex is chosen according to the probability distribution $\delta(v)$. Thus an infinite sequence of vertices (or an infinite walk) is formed which is a play.

Policies. Policies are recipes for players to extend finite prefixes of plays. Formally, a player-i *policy* is a function $\sigma_i : V^* \cdot V_i \mapsto V$ which maps every finite prefix $\omega \in V^* \cdot V_i$ of a play that ends in a player-i vertex v to a successor vertex $\sigma_i(\omega) \in V$, i.e., $(v, \sigma_i(\omega)) \in E$. A player-1 policy is *memoryless or stationary* if $\sigma_i(\omega) = \sigma_i(\omega')$ for all $\omega, \omega' \in V^* \cdot V_1$ that end in the same vertex $v \in V_1$, i.e., the policy does not depend on the entire prefix, but only on the last vertex.

Outcome of policies. Outcome of policies are as follows:

- In graphs, given a starting vertex, a policy for player 1 induces a unique play in the graph.
- In game graphs, given a starting vertex v, and policies σ_1, σ_2 for player 1 and player 2 respectively, the outcome is a unique play $\omega(v, \sigma_1, \sigma_2) = \langle v_0, v_1, v_2, \ldots \rangle$, where $v_0 = v$ and for all $i \geq 0$ if $v_i \in V_1$ then $\sigma_1(\langle v_0, \ldots, v_i \rangle) = v_{i+1}$ and if $v_i \in V_2$, then $\sigma_2(\langle v_0, \ldots, v_i \rangle) = v_{i+1}$.
- In MDPs, given a starting vertex v and a policy σ_1 for player 1, there is a unique probability measure over Ω which is denoted as $\Pr_v^{\sigma}(\cdot)$.

Objectives and winning. In general, an *objective* ϕ is a measurable subset of Ω. A play $\omega \in \Omega$ *achieves* the objective if $\omega \in \phi$. We consider the following notion of winning:

- *Almost-sure winning.* In MDPs, a player-1 policy σ is almost-sure (a.s.) winning from a starting vertex $v \in V$ for an objective ϕ iff $\Pr_v^{\sigma}(\phi) = 1$.
- *Winning.* In game graphs a policy σ_1 is *winning for player 1* from a starting vertex v iff the resulting play achieves the objective irrespective of the policy of player 2, i.e., for all σ_2 we have $\omega(v, \sigma_1, \sigma_2) \in \phi$.

Note that in the special case of graphs both of the above winning notions requires that there exists a play from v that achieves the objective.

Remark 2. *In MDPs we consider a.s. winning for which the precise transition probabilities of the transition function δ does not matter, but only the support of the transition function is relevant. The a.s. winning notion we use corresponds to the strong cyclic planning problem. Intuitively, if we visit a random vertex in an MDPs infinitely often then all its successors are visited infinitely often. This represents the local fairness condition (Clarke, Grumberg, and Peled 1999). Therefore, when we consider the MDP question only the underlying graph structure along with the partition is relevant, and the transition function δ can be treated as a uniform distribution over the support.*

We have defined the notion of objectives in general above, and below we consider specific objectives that are natural in planning problems. They are all variants of one of the most fundamental objectives in computer science, namely, reachability objectives.

Basic Target Reachability. For a set $T \subseteq V$ of target set vertices, the basic target reachability objective is the set of infinite paths that contain a vertex of T, i.e., $Reach(T) = \{\langle v_0, v_1, v_2, \ldots \rangle \in \Omega \mid \exists j \geq 0 : v_j \in T\}$.

Coverage Objective. For k different target sets, namely $T_1, T_2, \ldots, T_k$, the coverage objective asks whether for each $1 \leq i \leq k$ the basic target reachability objective $Reach(T_i)$

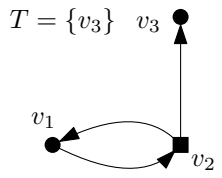

Figure 1: Example illustrating the difference between MDPs and Game Graphs for the reachability objective $Reach(T)$.

can be achieved. More precisely, given a starting vertex v, one asks whether for every $1 \leq i \leq k$ there is a policy σ_1^i to ensure winning (resp., a.s. winning) for the objective $Reach(T_i)$ from v for game graphs (resp., MDPs).

Sequential Target Reachability. For a tuple of vertex sets $\mathcal{T} = (T_1, T_2, \ldots, T_k)$ the sequential target reachability objective is the set of infinite paths that contain a vertex of T_1 followed by a vertex of T_2 and so on up to a vertex of T_k, i.e., $Seq(\mathcal{T}) = \{\langle v_0, v_1, v_2, \ldots\rangle \in \Omega \mid \exists j_1, j_2, \ldots j_k : v_{j_1} \in T_1, v_{j_2} \in T_2, \ldots, v_{j_k} \in T_k \text{ and } j_1 \leq j_2 \leq \cdots \leq j_k\}$.

Difference between MDPs and Game Graphs. Let the graph $G = (V, E)$ be defined as follows: Let $V = \{v_1, v_2, v_3\}$ and $E = \{(v_1, v_2), (v_2, v_1), (v_2, v_3)\}$. Let $T = \{v_3\}$ be a target set. We will now consider $Reach(T)$ for the MDP $P = (G, \langle V_1, V_R\rangle, \delta)$ and the game graph $\Gamma = (G, \langle V_1, V_2\rangle)$. Let $V_1 = \{v_1, v_3\}$ and $V_2 = V_R = \{v_2\}$. The example is illustrated in Figure 1. The adversary always chooses to go to v_1 and the target is never reached from v_1. On the other hand, if v_2 is probabilistic whenever the token is at v_2 it is moved to v_3 with non-zero probability. That is, almost-surely the transition from v_2 to v_3 is taken eventually, i.e. v_3 is reached almost-surely. Thus, reachability in MDPs does not imply reachability in game graphs.

Relevant parameters. We will consider the following parameters: n denotes the number of vertices, m denotes the number of edges and k will either denote the number of target sets in the coverage problem or the size of the tuple of target sets in the sequential target reachability problem.

Algorithmic study. In this work we study the above basic planning objectives for graphs, game graphs (i.e., winning in game graphs), and MDPs (a.s. winning in MDPs). Our goal is to clarify the algorithmic complexity of the above questions with improved algorithms and conditional lower bounds. We define the conjectured lower bounds for conditional lower bounds below.

Conjectured Lower Bounds

Results from classical complexity are based on standard complexity-theoretical assumptions, e.g., **P** $\neq$ **NP**. Similarly, we derive polynomial lower bounds which are based on widely believed, conjectured lower bounds on well studied algorithmic problems. In this work the lower bounds we derive depend on the popular conjectures below:

First of all, we consider conjectures on Boolean Matrix Multiplication (Williams and Williams 2018)[Theorem 6.1] and triangle detection in graphs (Abboud and Williams 2014)[Conjecture 2], which are the basis for lower bounds on dense graphs. A triangle in a graph is a triple x, y, z of vertices such that $(x, y), (y, z), (z, x) \in E$. We will for the rest of this work assume that vertices contain at least one outgoing edge and no self-loops in instances of Triangle. This can be easily established by linear time preprocessing. See Remark 3 for an explanation of the term "combinatorial algorithm".

Conjecture 1 (Combinatorial Boolean Matrix Multiplication Conjecture (BMM))**.** *There is no $O(n^{3-\epsilon})$ time combinatorial algorithm for computing the boolean product of two $n \times n$ matrices for any $\epsilon > 0$.*

Conjecture 2 (Strong Triangle Conjecture (STC))**.** *There is no $O(\min\{n^{\omega-\epsilon}, m^{2\omega/(\omega+1)-\epsilon}\})$ expected time algorithm and no $O(n^{3-\epsilon})$ time combinatorial algorithm that can detect whether a graph contains a triangle for any $\epsilon > 0$, where $\omega < 2.373$ is the matrix multiplication exponent.*

Williams and Williams (2010, Theorem 6.1) showed that BMM is equivalent to the combinatorial part of STC. Moreover, if we do not restrict ourselves to combinatorial algorithms, STC, still gives a super-linear lower bound.

Remark 3 (Combinatorial Algorithms)**.** *"Combinatorial" in Conjecture 2 means that it excludes "algebraic methods" (such as fast matrix multiplication (Williams 2012; Le Gall 2014)), which are impractical due to high associated constants. Therefore the term "combinatorial algorithm" comprises only discrete algorithms. Non-combinatorial algorithms usually have the matrix multiplication exponent ω in the running time. Notice that all algorithms for deciding almost-sure winning conditions in MDPs and winning conditions in games are discrete graph-theoretic algorithms and hence are combinatorial, and thus lower bounds for combinatorial algorithms are of particular interest in our setting. For further discussion consider (Ballard et al. 2012; Henzinger et al. 2015).*

Secondly, we consider the Strong Exponential Time Hypothesis (SETH) used also in (Abboud and Williams 2014)[Conjecture 1] introduced by (Impagliazzo and Paturi 1999; Impagliazzo, Paturi, and Zane 1998) for the satisfiability problem of propositional logic and the Orthogonal Vector Conjecture.

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

Conjecture 3 (Strong Exponential Time Hypothesis (SETH))**.** *For each $\epsilon > 0$ there is a k such that k-CNF-SAT on n variables and m clauses cannot be solved in $O(2^{(1-\epsilon)n} poly(m))$ time.*

Conjecture 4 (Orthogonal Vectors Conjecture (OVC))**.** *There is no $O(N^{2-\epsilon})$ time algorithm for the Orthogonal Vectors Problem for any $\epsilon > 0$.*

Williams (2005)[Theorem 5] SETH implies OVC, which is an implications of a result in (Williams 2005) and an explicit reduction is given in the survey article by Vassilevska-Williams (2018, Theorem 3.1). Whenever a problem is provably hard assuming OVC it is thus also hard when assuming SETH. For example, in (Bringmann and Künnemann

2015)[Preliminaries, A. Hardness Assumptions, OVH] the OVC is assumed to prove conditional lower bounds for the longest common subsequence problem. To the best of the author's knowledge, there is no connection between the former two and the latter two conjectures.

Remark 4. *The conjectures that no polynomial improvements over the best-known running times are possible do not exclude improvements by sub-polynomial factors such as polylogarithmic factors or factors of, e.g.,* $2^{\sqrt{\log n}}$.

Basic Previous Results

In this section, we recall the basic algorithmic results about MDPs and game graphs known in the literature that we later use in our algorithms.

Basic result 1: Maximal End-Component Decomposition. Given an MDP P, an *end-component* is a set of vertices $X \subseteq V$ s.t. (1) the subgraph induced by X is strongly connected (i.e., $(X, E \cap X \times X)$ is strongly connected) and (2) all random vertices have their outgoing edges in X, i.e., X is closed for random vertices, formally described as: for all $v \in X \cap V_R$ and all $(v, u) \in E$ we have $u \in X$. A *maximal end-component* (MEC) is an end-component which is maximal under set inclusion. The importance of MECs is as follows: (i) first it generalizes strongly connected components (SCCs) in graphs (with $V_R = \emptyset$) and closed recurrent sets of Markov chains (with $V_1 = \emptyset$); and (ii) in a MEC X from all vertices $u \in X$ every vertex $v \in X$ can be reached almost-surely. The MEC-decomposition of an MDP is the partition of the vertex set into MECs and the set of vertices which do not belong to any MEC. While MEC-decomposition generalizes SCC decomposition of graphs, and SCC decomposition can be computed in linear time (Tarjan 1972, Theorem 13), there is no linear-time algorithm for MEC-decomposition computation. The current best-known algorithmic bound for MEC-decomposition is $O(\min(n^2, m^{1.5}) = O(m \cdot n^{2/3}))$ (Chatterjee and Henzinger 2014, Theorem 3.6, Theorem 3.10).

Basic result 2: Reachability in MDPs. Given an MDP P and a target set T, the set of starting vertices from which T can be reached almost-surely can be computed in $O(m)$ time given the MEC-decomposition of P (Chatterjee et al. 2016, Theorem 4.1). Moreover, for the basic target reachability problem the current best-known algorithmic bounds are the same as the MEC-decomposition problem, i.e., $O(\min(n^2, m^{1.5})) = O(m \cdot n^{2/3})$ (Chatterjee and Henzinger 2014, Theorem 3.6, Theorem 3.10), and any improvement for the MEC-decomposition algorithm also carries over to the basic target reachability problem.

Basic result 3: Reachability in game graphs. Given a game graph Γ and a target set T, the set of starting vertices from which player 1 can ensure to reach T against all polices of player 2, is called *player-1 attractor* to T and can be computed in $O(m)$ time (Beeri 1980; Immerman 1981).

The above basic results from the literature explain the result of the first row of Table 1.

Coverage Problem

In this section, we consider the coverage problem. First, we present the algorithms, which are simple, and then focus on the conditional lower bounds for MDPs and game graphs, which establish that the existing algorithms cannot be (polynomially) improved under the STC and OV conjectures.

Algorithms

We present a linear-time algorithm for graphs, and quadratic time algorithm for MDPs and game graphs. The results below present the upper bounds of the second row of Table 1.

Planning in Graphs. For the coverage problem in graphs we are given a graph $G = (V, E)$, a vertex $s \in V$ and target sets $T_1, T_2, \ldots, T_k$. The algorithmic problem is to find out if starting from an initial vertex v the basic target reachability, i.e., $Reach(T_i)$, can be achieved for all $1 \leq i \leq k$. The algorithmic solution is as follows: Compute the BFS tree starting from s and check if all the targets are contained in the resulting BFS tree.

Planning in MDPs and Games. For both MDPs and game graphs with k target sets, the basic algorithm performs k basic reachability computations, i.e., for each target set T_i, $1 \leq i \leq k$, the basic target reachability for target set T_i is computed. (1) For game graphs, using the $O(m)$-time attractor computation (see Basic result 3), we have an $O(k \cdot m)$-time algorithm. (2) For MDPs, the MEC-decomposition followed by k many $O(m)$-time almost-sure reachability computation (see Basic result 2), gives an $O(k \cdot m + \text{MEC})$ time algorithm.

Conditional Lower Bounds

We present conditional lower bounds for the coverage problem in MDPs and game graphs (i.e., the CLBs of the second row of Table 1). For MDPs and game graphs the conditional lower bounds complement the quadratic algorithms from the previous subsection. The conditional lower bounds are due to reductions from OV and Triangle.

Sparse MDPs. For sparse MDPs we present a conditional lower bound based on OVC. To do that we reduce the OV problem to the coverage problem in MDPs.

Reduction 1. *Given two sets S_1, S_2 of d-dimensional vectors, we build the MDP P as follows.*

- *The vertices V of the MDP are given by a start vertex s, sets of 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 of the vectors in the OVC instance.*
- *The edges E of P are defined as follows: The start vertex s has an edge to every vertex of S_1. Furthermore for each $x_i \in S_1$ there is an edge to $c_j \in C$ iff $x_i[j] = 1$ and for each $y_i \in S_2$ there is an edge from $c_j \in S_2$ to y_i iff $y_i[j] = 1$.*
- *The set of vertices is partitioned into player-1 vertices $V_1 = S_1 \cup \mathcal{C} \cup S_2$ and random vertices $V_R = \{s\}$.*

The reduction is illustrated in Figure 2 (the dashed edges will be used later for the sequential target lower bounds).

Lemma 1. *Let $P = (V, E, \langle V_1, V_R \rangle, \delta)$ be the MDP given by Reduction 1 with target sets $T_i = \{y_i\}$ for $i = 1 \ldots N$.*

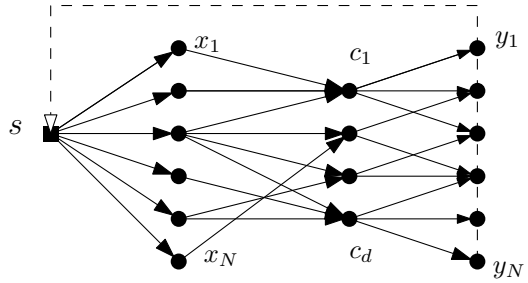

Figure 2: Reduction from OV

There exist orthogonal vectors $x \in S_1$, $y \in S_2$ iff there is no a.s. winning policy from s for the coverage objective.

Proof. Notice that when starting from s the token is randomly moved to one of the vertices x_i and thus player 1 can reach each y_j almost surely from s iff it can reach each y_j from each x_i. The MDP P is constructed in such a way that there is no path between vertex x_i and y_j iff the corresponding vectors are orthogonal in the OV instance: If x_i is orthogonal to y_j, the outgoing edges lead to no vertex which has an incoming edge to y_j as either $x_i[k] = 0$ or $y_j[k] = 0$. One the other hand, if there is no path from x_i to y_j we again have by the construction of the underlying graph that for all $1 \leq k \leq d : x_i[k] = 0$ or $y_j[k] = 0$. This is the definition of orthogonality for x_i and y_j. Thus, player 1 can reach all the target sets a.s. from s iff there are no orthogonal vectors in S_1 and S_2. □

The MDP P has only $O(N)$ many vertices and Reduction 1 can be performed in $O(N \log N)$ time (recall that $d = \omega(\log N)$). The number of edges m is $O(N \log N)$ and the number of target sets $k \in \theta(N)$. Thus the theorem below follows immediately.

Theorem 1. *There is no $O(m^{2-\epsilon})$ or $O((k \cdot m)^{1-\epsilon})$ (for any $\epsilon > 0$) algorithm to check if a vertex v has an a.s. winning policy for the coverage problem in MDPs under Conjecture 4 (i.e., unless OVC and SETH fail).*

Dense MDPs. For dense MDPs we present a conditional lower bound based on boolean matrix multiplication (BMM). Therefore we reduce the Triangle problem to the coverage problem in MDPs.

Reduction 2. *Given an instance of triangle detection, i.e., a graph $G = (V, E)$, we build the following MDP $P = (V', E', \langle V'_1, V'_R \rangle, \delta)$.*

- *The vertices V' are given as four copies V_1, V_2, V_3, V_4 of V and a start vertex s.*
- *The edges E' of P are defined as follows: There is an edge from s to every $v_{1i} \in V_1$ for $i = 1 \dots n$. In addition for $1 \leq j \leq 3$ there is an edge from v_{ji} to $v_{(j+1)k}$ iff $(v_i, v_k) \in E$.*
- *The set of vertices V' is partitioned into player-1 vertices $V'_1 = \emptyset$ and random vertices $V'_R = \{s\} \cup V_1 \cup V_2 \cup V_3 \cup V_4$.*

The reduction is illustrated in Figure 3 (the dashed edges will be used later for the sequential target lower bounds).

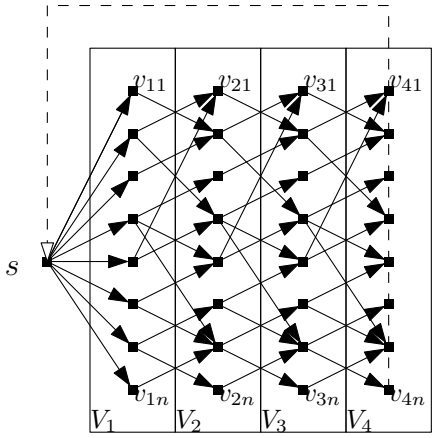

Figure 3: Reduction from Triangle

Lemma 2. *Let P be the MDP given by Reduction 2 with n target sets $T_1, \dots, T_n$. The target set $T_i = V_1 \setminus \{v_{1i}\} \cup V_4 \setminus \{v_{4i}\}$ for $i = 1 \dots n$. A graph G has a triangle iff player-1 has an a.s. winning policy from v for the coverage objective.*

Proof. Notice that there is a triangle in the graph G iff there is a path from some vertex v_{1i} in the first copy of G to the same vertex in the fourth copy of G, v_{4i}. Also, a path starting in s satisfies the coverage objective, i.e., reaches all target sets a.s., unless it visits a vertex v_{1i} and also v_{4i}. As each of these paths has non-zero probability player 1 wins almost-surely from v iff there is no such path iff there is no triangle in the original graph. □

Moreover, the size and the construction time of the MDP P are linear in the size of the original graph G and we have $k = \theta(n)$ target sets. Thus the theorem below follows immediately.

Theorem 2. *There is no combinatorial $O(n^{3-\epsilon})$ or $O((k \cdot n^2)^{1-\epsilon})$ algorithm (for any $\epsilon > 0$) to check if a vertex has an a.s. winning policy for the coverage objective in MDPs under Conjecture 2 (i.e., unless STC and BMM fail). The bounds hold for dense MDPs with $m = \theta(n^2)$.*

Next, we describe how the results for MDPs can be extended to game graphs.

Sparse Game Graphs. The random starting vertex in the reduction is changed to a player-2 vertex. The rest of the reduction stays the same. The proof then proceeds as before with the adversary player 2 now overtaking the role of the random choices.

Reduction 3. *Given two sets S_1, S_2 of d-dimensional vectors, we build the following game graph $\Gamma = (V, E, \langle V_1, V_2 \rangle)$.*

- *The vertices V of the game graph are given by a start vertex s, sets of 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 Γ are defined as follows: the start vertex s has an edge to every vertex of S_1. Furthermore for each $x_i \in S_1$ there is an edge to $c_j \in C$ iff $x_i[j] = 1$ and*

for each $y_i \in S_2$ there is an edge from $c_j \in S_2$ to y iff $y_i[j] = 1$.
- *The set of vertices is partitioned into player-1 vertices $V_1 = S_1 \cup \mathcal{C} \cup S_2$ and player-2 vertices $V_2 = \{s\}$.*

The reduction is illustrated in Figure 2 (the dashed edges will be used later for the sequential target lower bounds).

Lemma 3. *Let Γ be the game graph given by Reduction 3 with target sets target sets $T_i = \{y_i\}$ for $i = 1 \dots N$. There exist orthogonal vectors $x \in S_1$, $y \in S_2$ iff there is no winning policy from start vertex s for the coverage objective.*

Proof. Notice that when starting from s the token is moved to one of the vertices x_i and thus player 1 can reach each y_j from s iff it can reach each y_j from each x_i. If there is one y_j which cannot be reached from an x_i, player 2 will choose x_i as successor and win. The game graph Γ is constructed in such a way that there is no path between vertex x_i and y_j iff the corresponding vectors are orthogonal in the OV instance: If x_i is orthogonal to y_j, the outgoing edges lead to no vertex which has an incoming edge to y_j as either $x_i[k] = 0$ or $y_j[k] = 0$. One the other hand, if there is no path from x_i to y_j we again have by the construction of the underlying graph that for all $1 \leq k \leq d : x_i[k] = 0$ or $y_j[k] = 0$. This is the definition of orthogonality for x_i and y_j. Thus, player 1 can reach all the target sets from starting vertex s iff there are no orthogonal vectors in S_1 and S_2. □

The game graph Γ has only $O(N)$ many vertices and Reduction 1 can be performed in $O(N \log N)$ time (recall that $d = \omega(\log N)$). The number of edges m is $O(N \log N)$ and the number of target sets $k \in \theta(N)$. Thus the theorem below follows immediately.

Theorem 3. *There is no $O(m^{2-\epsilon})$ or $O((k \cdot m)^{1-\epsilon})$ algorithm (for any $\epsilon > 0$) to check if a vertex has a winning policy for the coverage objective with k reachability objectives in game graphs under Conjecture 4 (i.e., unless OVC and SETH fail).*

Dense Game Graphs. The random vertices in the reduction are now player-2 vertices. Notice that the resulting game graph Γ has only player-2 vertices. Now if there is a path starting from s that is not in the defined coverage objective then player 2 would simply choose that one and thus player 1 still wins iff there is no such path, i.e., there is no triangle in the original graph.

Reduction 4. *Given an instance of triangle detection, i.e., a graph $G = (V, E)$, we build the following game graph $\Gamma = (V', E', \langle V_1', V_2' \rangle)$.*
- *The vertices V' are given as four copies V_1, V_2, V_3, V_4 of V and a start vertex s.*
- *The edges E' are defined as follows: There is an edge from s to every $v_{1i} \in V_1$ for $i = 1 \dots n$. In addition for $1 \leq j \leq 3$ there is an edge from v_{ji} to $v_{(j+1)k}$ iff $(v_i, v_k) \in E$.*
- *The set of vertices V' is partitioned into player-1 vertices $V_1' = \emptyset$ and player-2 vertices $V_2' = \{s\} \cup V_1 \cup V_2 \cup V_3 \cup V_4$.*

Lemma 4. *Let Γ be the game graph given by Reduction 4 with n target sets $T_1, \dots, T_n$. The target set $T_i = V_1 \setminus \{v_{1i}\} \cup V_4 \setminus \{v_{4i}\}$ for $i = 1 \dots n$. A graph G has a triangle iff player 1 has a winning policy from s for the coverage objective.*

Proof. Notice that there is a triangle in the graph G iff there is a path from some vertex v_{1i} in the first copy of G to the same vertex in the fourth copy of G, v_{4i}. Also, a path starting in s satisfies the coverage objective, i.e., reaches all target sets unless it visits a vertex v_{1i} and also v_{4i}. Player 1 wins from v iff there is no such path as player 2 choose it. Such a path exists as proved above iff there is no triangle in the original graph. □

Moreover, the size and the construction time of game graph Γ are linear in the size of the original graph G and we have $k = \theta(n)$ target sets. Thus the theorem below follows immediately.

Theorem 4. *There is no combinatorial $O(n^{3-\epsilon})$ or $O((k \cdot n^2)^{1-\epsilon})$ algorithm (for any $\epsilon > 0$) to check whether a vertex v has a winning policy for the coverage objective in game graphs under Conjecture 2 (i.e., unless STC and BMM fail). The bounds hold for dense game graphs with $m = \theta(n^2)$.*

Sequential Target Problem

We consider the sequential target problem in graphs, MDPs and game graphs. In contrast to the quadratic CLB for the coverage problem, quite surprisingly we present a subquadratic algorithm for MDPs, which as a special case gives a linear-time algorithm for graphs. For games, we present a quadratic algorithm and a quadratic CLB.

Algorithms

The results below present the upper bounds of the third row of Table 1.

Planning in MDPs. We first calculate the MEC-decomposition of the MDP. Then each MECs is collapsed into a single vertex which we set to be a player-1 vertex. In the target sets, all the vertices of the MEC are replaced by this new vertex. This does not change the reachability conditions of the resulting MDP: Every vertex in th MEC can be reached almost surely starting from every other vertex in the same MEC, regardless of their type (player-1, random). Thus it suffices to give an algorithm for a MEC-free MDP $P = (V, E, \langle V_1, V_R \rangle, \delta)$ with tuple of target sets $(T_1, \dots, T_k)$.

The vertices in S are the vertices that are not processed yet and S is initialized with V. Initially, vertices with no outgoing edges are added to a queue Q. Throughout the algorithm, the queue Q contains the vertices which have not been processed so far but whose successors are already processed.

While the queue Q is not empty, a vertex from the queue is processed. When a vertex v is processed the function `ProcessVertex`(v) is called. The function calculates the label ℓ_v of the vertex v and updates variables $best_w$ and $count_w$ of the other vertices. The label ℓ_v means vertex v has an almost-sure winning policy for the objective $Seq(\mathcal{T}_{\ell_v})$ where $\mathcal{T}_{\ell_v} = (T_{\ell_v}, \dots, T_k)$. Note that this means that vertices with label 1 have an almost-sure winning policy for the

Algorithm 1: Sequential target Reachability for MEC-free MDPs.

Input: MEC-free MDP $P = (V, E, \langle V_1, V_R \rangle, \delta)$ and a tuple of target sets $\mathcal{T} = (T_1, \ldots, T_k)$.
Output: All vertices with a policy for $Seq(\mathcal{T})$.

```
1  L_v ← {i | v ∈ T_i : i = 1...k} ∀v ∈ V;
2  A[i] ← 0 for 1 ≤ i ≤ k;
3  count_v ← number of outgoing edges of v;
4  best_v ← null, ℓ_v ← null for v ∈ V;
5  best_v ← k + 1 for v ∈ V with no outgoing edges;
6  S ← V;
7  Queue Q ← {v ∈ V | v has no outgoing edges};
8  while S ≠ ∅ do
9      if Q ≠ ∅ then
10         v = Q.pop();
11         ProcessVertex(v);
12     else
13         v ← argmax_{v∈V_R∩S} best_v;
14         ProcessVertex(v);
15 return {v ∈ V | ℓ_v = 1};
16 function ProcessVertex(Vertex v)
17     for i ∈ L_v do A[i] ← 1 ;
18     ℓ_v ← best_v;
19     while A[ℓ_v − 1] = 1 ∧ ℓ_v > 1 do
20         ℓ_v ← ℓ_v − 1;
21     for i ∈ L_v do A[i] ← 0;
22     S ← S \ {v};
23     for w ∈ {w : (w, v) ∈ E} do
24         if w ∈ V_1 then
25             best_w ← min(best_w, ℓ_v)
26         else
27             best_w ← max(best_w, ℓ_v)
28         count_w ← count_w − 1;
29         if count_w = 0 ∧ w ∈ S then
30             Q.push(w)
```

objective $Seq(\mathcal{T})$ where $\mathcal{T} = \{T_1, T_2, \ldots T_k\}$. The variables $best_v$ are used to store the maximum (for $v \in V_R$) / minimum (for $v \in V_1$) label of the already processed successors of v.

Now when Q is empty, then the algorithm has to process a vertex where not all successors have been processed yet. In that case, one considers all the random vertices for which at least one successor has already been processed and chooses the random vertex with maximum $best_v$ to process next. Notice that the function argmax ignores arguments with *null* values. One can show that, as the graph has no MECs, whenever Q is empty (and S is not) there exist such a random vertex. Moreover, whenever Q is empty, all vertices in the set of unprocessed vertices S have a policy that satisfies $Seq(\mathcal{T}_m)$ for $m = \max_{v \in V_R \cap S} best_v$. Intuitively, this is due to the fact that all vertices $v \in S$ can reach the set of already processed vertices and in the worst case the reached vertex v' has $\ell_{v'} = m$ and thus a strategy for $Seq(\mathcal{T}_m)$. For the selected vertex v all its successors w will, thus, finally have a label ℓ_w of at most m, and, as the current value of $best_v$ is m, there is a successor w with $\ell_w = m$. Thus, as $v \in V_R$, we have that also the final value of $best_v$ must be m. Hence, one can already process v without knowing the labels of all the successors.

Proposition 1 (Correctness)**.** *Given an MDP P and a sequential target objective $Seq(\mathcal{T})$ with targets $\mathcal{T} = \{T_1, \ldots, T_k\}$, Algorithm 1 decides whether there is a player-1 policy at a start vertex s for the objective $Seq(\mathcal{T})$.*

We next state invariants of the while loop (see Line 8) that we will use later to show a loop invariant that will establish the correctness of the algorithm.

Lemma 5. *The following statements are invariants of the while loop in Line 8.*

1. $count_v = |Out(v) \cap S|$;
2. $v \in Q$ *iff* $v \in S$ *and* $Out(v) \cap S = \emptyset$;
3. $best_v = k + 1$, *for all* $v \in V$ *with* $Out(v) = \emptyset$.
4. $best_v = \begin{cases} \min_{w \in Out(v) \setminus S} \ell_w & v \in V_1 \\ \max_{w \in Out(v) \setminus S} \ell_w & v \in V_R \end{cases}$, *for all* $v \in V$ *with* $Out(v) \neq \emptyset$;
   In particular $\ell_w \neq$ *null for all* $w \in V \setminus S$.
5. *If* $S \neq \emptyset$ *and* $Q = \emptyset$ *there is a* $v \in S \cap V_R$ *such that* $best_v \neq$ *null.*

The above invariants state that (a) the variables have the intended meaning, (b) Q contains all the unprocessed vertices whose successors are already processed, and (c) that the function argmax is well-defined whenever called. These are three important ingredients to show the correctness of Algorithm 1.

Proof. 1. The counters $count_v$ are initialized as $|Out(v)|$ and S is initialized as V. Thus the claim holds when first entering the while loop.
Assume the claim holds at the beginning of the iteration where vertex v is processed. The set S is only changed in Line 22 where only v is removed from the set while the counters are only changed in Line 28, where all counters of vertices w with $v \in Out(w)$ are decreased by one (notice that $v \in Out(w)$ iff $w \in In(v)$). That is, $count_v = |Out(v) \cap S|$ also after this iteration of the loop and the claim follows.

2. In the initial phase S is set to V and Q is set to $\{v \in V \mid Out(v) = \emptyset\}$. Thus the claim holds when first entering the while loop.
Assume the claim holds at the beginning of the iteration where vertex v is processed. The set S is only changed in Line 22 where v is removed.
First consider a vertex $w \in Q \setminus \{v\}$. As w is not removed from the set S and no vertex is added to S the claim is still true for w. Now consider a vertex $w \in S \setminus Q$ that might be added during the iteration of the loop. This can only happen in Line 30 and the if conditions ensures that $w \in S$ and $Out(v) \cap S = \emptyset$ (by the previous invariant). Thus the claim also holds for the newly added vertices.

3. For $v \in V$ with $Out(v) = \emptyset$ the variables $best_v$ are initialized with $k+1$ and $best_v$ is only changed in Line 25 or Line 27, when a successor of the vertex is processed. As v has no successor, $best_v$ is not changed during the algorithm.
4. Note that we define the $\max$ or $\min$ over the empty set to be null. As all ℓ_v are initialized as null and $best_v$ with $Out(v) \neq \emptyset$ are initialized as *null* the claim holds when the algorithm enters the loop.
   Now consider the iteration of vertex v and assume the claim is true at the beginning. The set S is only changed in Line 22 where v is removed. Let S_{old} be the set at the beginning of the iteration and $S_{new} = S_{old} \setminus \{v\}$ the updated set. First notice that $best_v \neq$ *null* as it is either chosen by (a) as element of Q or (b) by argmax. In the former case it was either initially set to $k+1$ or it was added to Q when processing a vertex $w \in Out(v)$, which would have set $best_v$. In the latter case $best_v \neq$ *null* by the definition of argmax. Now, as $best_v \neq$ *null* the assignment in Line 18 ensures that also $\ell_v \neq$ *null*. For a vertex $w \in In(v) \cap V_1$ the value $best_w$ is updated to $\min(best_w, \ell_v)$ (Line 25) which by assumption is equal to $\min_{x \in (Out(w) \setminus S_{old}) \cup v} \ell_x = \min_{x \in (Out(w) \setminus S_{new})} \ell_x$, i.e., the equation holds. For a vertex $w \in In(v) \cap V_R$ the value $best_w$ is updated to $\max(best_w, \ell_v)$ (Line 27) which by assumption is equal to $\max_{x \in (Out(w) \setminus S_{old}) \cup v} \ell_x = \max_{x \in (Out(w) \setminus S_{new})} \ell_x$, i.e., the equation holds. For vertices $w \notin In(v)$ both $best_w$ as well as the right hand side of the equation are unchanged. Hence, the claim holds also after the iteration.
5. The input graph has a vertex v with $Out(v) = \emptyset$. Towards a contradiction assume that no such vertex exists. Then an SCC C where every vertex in C has only edges to other vertices in C exists. Such SCCs are called *bottom SCCs*. Bottom SCCs are MECs (Chatterjee and Henzinger 2014) and we assumed that there are no MECs in the MDP, a contradiction. Thus, Q is non-empty after the initialization and the claim holds after the initialization. Now consider the iteration of vertex v and assume the claim is true at the beginning and $Q = \emptyset$. Notice that $best_w$ is set for vertices as soon as one vertex $w \in Out(v)$ was processed. Towards a contradiction assume that all vertices in $S \cap V_R$ have $best_w =$ *null*, i.e., no vertex $v \in S \cap V_R$ has a successor in $V \setminus S$. Each $v \in S$ has at least one successor in S as otherwise, v would be in Q. That is S is either empty or has a non-trivial bottom SCC that has no random outgoing edges. Again such an SCC would be a MEC and thus we obtain our desired contradiction.

□

From the following invariant, we obtain the correctness of our algorithm.

Lemma 6. *The following statements are invariants of the while loop in Line 8 or all $v \in V \setminus S$:*

1. *there exists a player 1 policy σ s.t. $\Pr_v^\sigma(Seq(\mathcal{T}_{\ell_v})) = 1$; and*
2. *there is no player 1 policy σ s.t. $\Pr_v^\sigma(Seq(\mathcal{T}_{\ell_v - 1})) = 1$.*

where $\mathcal{T}_{\ell_v} = \{T_{\ell_v}, \ldots, T_k\}$ or $\ell_v > k$.

Proof. As S is initialized as set V the two statements hold after the initialization.

Now consider the iteration where vertex v is processed and assume the invariants hold at the beginning of the iteration. We first introduce the following notation

$$be(v) = \begin{cases} \min_{w \in Out(v)} \ell_w & v \in V_1 \\ \max_{w \in Out(v)} \ell_w & v \in V_R \end{cases}$$

We distinguish the case where Q is non-empty and the case where Q is empty.

- *Case $Q \neq \emptyset$*: By Lemma 5 we have $Out(v) \cap S = \emptyset$ and thus also $\ell_w \neq$ *null* for all $w \in Out(v)$. Thus by Lemma 5(4) we have $be(v) = best_v$ and ℓ_v can be computed. By the while loop in Line 19 we have L_v contains $\ell_v, \ldots, best_v - 1$ but does not contain $\ell_v - 1$.
  1) Thus we can easily obtain a policy σ with $\Pr_v^\sigma(Seq(\mathcal{T}_{\ell_v})) = 1$ as follows. If $v \in V_1$ pick the vertex w that corresponds to $be(v)$ and then player 1 can follow the existing policy for vertex w. If $v \in V_R$ then which ever vertex $w \in Out(v)$ is randomly chosen follow the existing policy for w. In both cases, the claim follows from the inductive assumption on w.
  2) We next show that there is no policy for $Seq(\mathcal{T}_{\ell_v - 1})$. If $v \in V_1$ we have that the current vertex is not in the set $T_{\ell_v - 1}$ and no successor w has a policy σ with $\Pr_w^\sigma(Seq(\mathcal{T}_{\ell_v - 1})) = 1$ as by the inductive assumption $T_{\ell_v - 1}$ cannot be reached a.s. from any successor of v and thus there is also no policy σ for $\Pr_v^\sigma(Seq(\mathcal{T}_{\ell_v - 1})) = 1$. Similar for $v \in V_R$ we have that the current vertex is not in the set $T_{\ell_v - 1}$ and there is at least one successor w where there is no policy σ with $\Pr_w^\sigma(Seq(\mathcal{T}_{\ell_v - 1})) = 1$ and thus there is also no policy σ with $\Pr_v^\sigma(Seq(\mathcal{T}_{\ell_v - 1})) = 1$ as there is a non-zero chance that a vertex w is picked that, by the inductive assumption, cannot reach a node in $T_{l_v - 1}$.
- *Case $Q = \emptyset$*: As shown in the proof of Lemma 5(4,5) $best_w$ is not *null* for all $w \in S$ that have an edge to vertices in $V \setminus S$ and there is at least one vertex in $V_R \cap S$ that has an edge to $V \setminus S$. That is, the operator in Line 13 returns an argument v, where $best_v \neq$ *null* by the choice of v, and thus ℓ_v can be computed. Let $best_{max} = max_{v \in V_R \cap S}\ best_v$.
  1) As we have no MEC (in S), there is a policy σ, so that the play almost surely leaves S by using one of the outgoing edges of a random node: The policy σ can be arbitrary, except that for a player-1 vertex $x \in S$ with an edge (x, y) where $y \in V \setminus S$ we choose $\sigma(x) \in S$ (which must exist as x would be in Q otherwise). As there are no MECs (in S) the policy σ_1 will eventually go to $V \setminus S$ using a random node. This implies that from each vertex in S player 1 has a policy to reach a vertex in $V \setminus S$ coming from a random vertex. By inductive assumption each successor of such random vertex has a policy to satisfy $Seq(\mathcal{T}_{best_{max}})$. Thus it follows that from each vertex in S player 1 has a policy to satisfy $Seq(\mathcal{T}_{best_{max}})$. Now consider the random vertex v that was chosen by the algorithm as $\mathrm{argmax}_{v \in V_R \cap S}\ best_v$. By the above all successors have a policy to satisfy $Seq(\mathcal{T}_{best_{max}})$ almost-surely.

Now as L_v contains $\ell_v, \dots, best_v - 1$ but does not contain $\ell_v - 1$ we obtain a policy σ with $\Pr_v^\sigma(Seq(\mathcal{T}_{\ell_v})) = 1$.
2) By the choice of v there is also a successor (that is chosen with non-zero probability) that, by assumption, has no policy for $Seq(\mathcal{T}_{best_{max}-1})$ and, moreover, L_v does not contain $\ell_v - 1$. Thus, when starting in v each policy will fail to satisfy $Seq(\mathcal{T}_{best_{max}-1})$ with non-zero probability, i.e., there is no policy σ for $\Pr_v^\sigma(Seq(\mathcal{T}_{\ell_v-1})) = 1$.

□

Proposition 2 (Running Time)**.** *Algorithm 1 runs in* $O(m \log n + \sum_{i=0}^{k} |T_i|)$ *time.*

Proof. Initializing the algorithm takes $O(m + \sum_{i=0}^{k} |T_i|)$ time. This is due to the fact that we calculate L_v in $O(n + m + k)$ time at Line 1. The other initialization steps take only $O(m)$ time(lines 2-6). Now consider the while loop. Every vertex $v \in V$ is processed once. The costly operations are the call of the `ProcessVertex` function and the evaluation of the argmax function. Evaluating `ProcessVertex(v)` takes time linear in the number of incoming edges of v plus $|L_v|$. Summing up over all vertices we obtain a $O(m + \sum_{i=0}^{k} |T_i|)$ bound. To compute argmax efficiently we have to maintain a priority queue containing all not yet finished random vertices. As we have $O(m)$ updates this costs only $O(m \log n)$ for one of the standard implementations of priority queues. Summing up this yields a $O(m \log n + \sum_{i=0}^{k} |T_i|)$ running time for Algorithm 1 . □

By considering also the time MEC for the MEC decomposition we obtain the desired bound and the following theorem.

Theorem 5. *Given an MDP P, a starting vertex s and a tuple of targets $\mathcal{T} = (T_1, \dots, T_k)$, we can calculate whether there is a player-1 policy σ_1 at s for the objective $Seq(\mathcal{T})$ in* $O(\text{MEC} + m \log n + \sum_{i=0}^{k} |T_i|)$ *time.*

Planning in Graphs.

The algorithm for graphs works identically to the algorithm for MDPs but it does not need the priority queue. This is due to the fact that Q is always non-empty and the MEC decomposition reduces to computing SCCs. We thus obtain a running time of $O(m + \sum_{i=1}^{n} |T_i|)$. The resulting Algorithm for graphs is given as Algorithm 2.

Proposition 3 (Correctness)**.** *Given a DAG $D = (V, E)$ and a sequential reachability objective $Seq(\mathcal{T})$ with target sets $\mathcal{T} = \{T_1, \dots, T_k\}$, Algorithm 2 determines whether a start vertex s has a path for the objective $Seq(\mathcal{T})$.*

Observation 1. *The input graph has a vertex v with $Out(v) = \emptyset$ and thus Q is non-empty after the initialization.*

Proof. Note that there is always a vertex $v \in V$ where $Out(v) = \emptyset$ because we assumed that D is a DAG. □

The invariants below state that (a) the variables have the intended meaning, (b) Q contains all the unprocessed vertices whose successors are already processed and (c) that the queue contains vertices as long as S is not empty.

Algorithm 2: Backward Label Propagation Algorithm

```
Input: DAG D = (V, E) and target sets
       T = {T_1, ..., T_k}
1  L_v ← {i | v ∈ T_i : i = 1...k} ∀v ∈ V;
2  A[i] ← 0 for 1 ≤ i ≤ k;
3  count_v ← |Out(v)| for v ∈ V;
4  best_v ← null, ℓ_v ← null for all v ∈ V;
5  best_v ← k + 1 for v ∈ V with Out(v) = ∅;
6  S ← V, Queue Q ← {v ∈ V | Out(v) = ∅};
7  while S ≠ ∅ do
8      v = Q.pop();
9      ProcessVertex(v);
10 return {v ∈ V | ℓ_v = 1};
11 function ProcessVertex(Vertex v)
12     for i ∈ L_v do A[i] ← 1 ;
13     ℓ_v ← best_v;
14     while A[ℓ_v − 1] = 1 ∧ ℓ_v > 1 do
15         ℓ_v ← ℓ_v − 1;
16     for i ∈ L_v do A[i] ← 0 ;
17     S ← S \ {v};
18     for w ∈ In(v) do
19         best_w ← min(best_w, ℓ_v);
20         count_w ← count_w − 1;
21         if count_w = 0 ∧ w ∈ S then
22             Q.push(w);
```

Lemma 7. *The following statements are invariants of the while loop at Line 7.*

1. $count_v = |Out(v) \cap S|$
2. *$v \in Q$ iff. $v \in S$ and all $Out(v) \cap S = \emptyset$.*
3. *If S is not empty then the queue Q is not empty.*
4. $best_v = k + 1$ *for all $v \in V$ with $Out(v) = \emptyset$.*
5. $best_v = \min_{w \in Out(v) \setminus S} \ell_w$, *for all $v \in V$ with $Out(v) \neq \emptyset$. In particular $\ell_w \neq null$ for all $w \in V \setminus S$.*

Proof. 1. The counters $count_v$ are initialized as $|Out(v)|$ and S is initialized as V. Thus the claim holds when first entering the while loop.
Assume the claim holds at the beginning of the iteration where vertex v is processed. The set S is only changed in Line 17. There v is removed from the set. The counters are only changed in Line 20: All counters of vertices w with $v \in Out(w)$ are decreased by one (notice that $v \in Out(w)$ iff $w \in In(v)$). Consequently $count_v = |Out(v) \cap S|$ also after this iteration of the loop and the claim follows.

2. In the initial phase S is set to V and Q is set to $\{v \in V \mid Out(v) = \emptyset\}$. Thus the claim holds when first entering the while loop.
Assume the claim holds at the beginning of the iteration where vertex v is processed. The set S is only changed in Line 17 where v is removed.
First consider a vertex $w \in Q \setminus \{v\}$. As w is not removed from the set S and no vertex is added to S the claim is

still true for w. Now consider a vertex w that might be added during the iteration of the loop. This can only happen in Line 22 and the if conditions ensure that $w \in S$ and $Out(v) \cap S = \emptyset$ (by the previous invariant) and thus the claim also holds for the newly added vertices.

3. Due to Observation 1 the claim holds when first entering the while loop.
   Assume the claim holds at the beginning of the iteration, where vertex v is processed. The vertex v is removed from S in Line 17 and if the set S is empty now, the claim follows trivially. On the other hand, if S is non-empty and Q is also non-empty the claim follows again. In the third case S is non-empty and Q is empty. Assume for contradiction that no vertex is added at line 22. By invariant (2), every vertex $v \in S$ has a successor in S as otherwise, v would be in Q. That implies that there exists an SCC which is a contradiction because we assumed that D is a DAG.
4. For $v \in V$ with $Out(v) = \emptyset$ the variables $best_v$ are initialized with $k+1$ and $best_v$ is only changed in Line 19 when a successor of the vertex is processed. As v has no successor, $best_v$ is not changed during the algorithm.
5. Note that we define the $\max$ or $\min$ over the empty set to be null. As all ℓ_v are initialized as null and $best_v$ with $Out(v) \neq \emptyset$ are initialized as *null* the claim holds when the algorithm enters the loop.
   Now consider the iteration of vertex v and assume the claim is true at the beginning. The set S is only changed in Line 17 where v is removed. Let S_{old} be the set at the beginning of the iteration and $S_{new} = S_{old} \setminus \{v\}$ the updated set. First notice that $best_v \neq$ *null* as v was in Q. It was either initially set to $k+1$ or it was added to Q when processing a vertex $w \in Out(v)$, which would have set $best_v$. Now, as $best_v \neq$ *null* the assignment in Line 13 ensures that also $\ell_v \neq$ *null*. For a vertex $w \in In(v)$, the value $best_w$ is updated to $\min(best_w, \ell_v)$ (Line 19) which by assumption is equal to $\min_{x \in (Out(w) \setminus S_{old}) \cup v} \ell_x = \min_{x \in (Out(w) \setminus S_{new})} \ell_x$, i.e., the equation holds. For vertices $w \notin In(v)$ both $best_w$ as well as the right hand side of the equation are unchanged. Hence, the claim holds also after the iteration.

□

From the following invariants we obtain the correctness of our algorithm.

Lemma 8. *The following statements are invariants of the while loop at Line 7 for all $v \in V \setminus S$:*

1. *For all $v \in V$ there exists a path $p_v \in Seq(\mathcal{T}_{\ell_v})$.*
2. *For all $v \in V$ there exists no path $p_v \in Seq(\mathcal{T}_{\ell_v - 1})$.*

where $\mathcal{T}_{\ell_v} = \{T_{\ell_v}, \ldots, T_k\}$ or $\ell_v > k$.

Proof. As S is initialized with the set of vertices V the two statements trivially hold after the initialization.

Now consider the iteration where vertex v is processed and assume the invariants hold at the beginning of the iteration. We first introduce the following notation $be(v) = \min_{w \in Out(v)} \ell_w$. By Lemma 7 we have $Out(v) \cap S = \emptyset$ and thus also $\ell_w \neq$ *null* for all $w \in Out(v)$. Thus by Lemma 7(5) we have $be(v) = best_v$ and ℓ_v can be computed. By the while loop in Line 14 we have L_v contains $\ell_v, \ldots, best_v - 1$ but does not contain $\ell_v - 1$.

1. We next show that there is a path p_v in $Seq(\mathcal{T}_{\ell_v})$: Let $w = be(v)$. A path for vertex w where $p_w \in Seq(\mathcal{T}_{\ell_w})$, exists by induction hypothesis. The targets $\{T_{\ell_v}, \ldots, T_{\ell_{w-1}}\}$ are visited by starting from v. The path is obtained as follows: $p_v = v, p_w$, which proves the claim.
2. We next show that there is no path p_v in $Seq(\mathcal{T}_{\ell_v - 1})$. The current vertex v is not in the set $T_{\ell_v - 1}$ and no successor w has a path p_w with $p_w \in Seq(\mathcal{T}_{\ell_v - 1})$ because $\ell_v - 1 < \ell_w$. Thus there is also no path $p_v \in Seq(\mathcal{T}_{\ell_v - 1})) = 1$ which concludes the proof.

□

Proposition 4 (Running Time)**.** *Algorithm 2 has running time $O(m + \sum_{i=1}^{n} |T_i|)$.*

Proof. The initialization of A takes $O(k)$ time. Initializing the sets L_v costs $\sum_{i=1}^{n} |T_i|$ time. We process every vertex $v \in V$ with the function $ProcessVertex(\, Vertex\ v)$ at line 9 because we assume the input graph D is a DAG. In the function, all incoming edges of v are processed once (line 15). When processing an edge, we do constant work in lines (18-22). The vertices have total work $O(\sum_{i=1}^{n} |T_i|)$ to do: Each of them goes through the list of their label three times (lines 12,14-15,16). This yields a total running time of $O(m + \sum_{i=1}^{n} |T_i|)$. □

Theorem 6. *Given a graph $G = (V, E)$, a starting vertex s and a tuple of targets $\mathcal{T} = (T_1, \ldots, T_k)$, we can calculate whether there is a player-1 policy σ_1 at a start vertex s for the objective $Seq(\mathcal{T})$ in $O(m + \sum_{i=1}^{k} |T_i|)$ time.*

Planning in Games. For game graphs with the tuple $\mathcal{T} = (T_1, \ldots T_k)$ and starting vertex s, the basic algorithm performs k player-1 attractor computations, starting with computing the attractor $S_k = Attr_1(T_k)$ of T_k, then computing $S_\ell = Attr_1(S_{\ell+1} \cap T_\ell)$ for $1 \leq \ell < k$, and finally returning S_1. This gives an $O(k \cdot m)$-time algorithm.

Conditional Lower Bounds

We present CLBs for game graphs based on the conjectures STC, SETH and OVC, which establish the CLBs for the third row of Table 1.

Sparse Game Graphs. For sparse game graphs, we present conditional lower bounds based on OVC. The reduction is an extension of Reduction 1, where we (a) produce player-2 vertices instead of random vertices and (b) also every vertex of S_2 has an edge back to s. The reduction is illustrated in Figure 2.

Reduction 5. *Given two sets S_1, S_2 of d-dimensional vectors, we build the following game graph Γ.*

- *The vertices V of the game graph are given by a start vertex s, sets of 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 of the vectors in the OVC instance.*

- *The edges E of Γ 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 back to s; furthermore for each $x_i \in S_1$ there is an edge to $c_j \in C$ iff $x_i[j] = 1$ and for each $y_i \in S_2$ there is an edge from $c_j \in S_2$ to y iff $y_i[j] = 1$.*
- *The set of vertices is partitioned into player-1 vertices $V_1 = S_1 \cup \mathcal{C} \cup S_2$ and player-2 vertices $V_2 = \{s\}$.*

Lemma 9. *Let Γ be the game graph given by Reduction 5 with a tuple of target sets $\mathcal{T} = (T_1, \ldots, T_k)$ where $T_i = \{y_i\}$ for $i = 1 \ldots N$. There exist orthogonal vectors $x_i \in S_1$, $y_j \in S_2$ iff s has no player-1 policy σ_1 to ensure winning for the objective $Seq(\mathcal{T})$.*

Proof. Notice that the game graph Γ is constructed in such a way that there is no path between x_i and y_j iff they are orthogonal in the OV instance. Notice that each play starting at s revisits s every four steps and if there is no path between x_i and y_j then player 2 can disrupt player 1 from visiting a target T_j by moving the token to x_i whenever the token is in s. However, if there is no such x_i and y_j, player 2 cannot disrupt player 1 from s because no matter which vertex x_i player 2 chooses, player 1 has a policy to reach the next target set. If s has no player-1 policy σ_1 to ensure winning for the objective $Seq(\mathcal{T})$ there must be a target player 1 cannot reach. This must be due to the fact that there is no path between some x_i and y_j and player 2 always chooses x_i. □

The number of vertices in Γ, constructed by Reduction 1 is $O(N)$ and the construction can be performed in $O(N \log N)$ time (recall that $(d = \omega(\log N))$. The number of edges m is $O(N \log N)$ (thus we consider G to be a sparse graph) and the number of target sets $k \in \theta(N) = \theta(m/\log N)$.

Theorem 7. *There is no $O(m^{2-\epsilon})$ or $O((k \cdot m)^{1-\epsilon})$ algorithm (for any $\epsilon > 0$) to check if a vertex v has a winning policy for sequential reachability objectives in game graphs under Conjecture 4 (i.e., unless OVC and SETH fail).*

Dense Game Graphs. For dense game graphs, we present a conditional lower bound based on BMM. The reduction extends Reduction 2, where we (a) again produce player-2 vertices instead of random vertices and (b) every vertex in the fourth copy has an edge back to s. The reduction is illustrated in Figure 3.

Reduction 6. *Given an instance of triangle detection, i.e., a graph $G = (V, E)$, we build the following game graph $\Gamma = (V', E', \langle V_1', V_2' \rangle)$.*
- *The vertices V' are given as four copies V_1, V_2, V_3, V_4 of V and a start vertex s.*
- *The edges E' are defined as follows: There is an edge from s to every $v_{1i} \in V_1$ where $i = 1 \ldots n$. In addition for $1 \leq j \leq 3$ there is an edge from v_{ji} to $v_{(j+1)k}$ iff $(v_i, v_k) \in E$. Furthermore there are edges from every $v_{4i} \in V_4$ to the start vertex s.*
- *The set of vertices V' is partitioned into player-1 vertices $V_1' = \emptyset$ and player-2 vertices $V_2' = \{s\} \cup V_1 \cup V_2 \cup V_3 \cup V_4$.*

Lemma 10. *Let Γ' be the game graphs given by Reduction 6 with a tuple of target sets $\mathcal{T} = (T_1, T_2, \ldots, T_k)$ where $T_i = V_1 \setminus \{v_{1i}\} \cup V_4 \setminus \{v_{4i}\}$ for $i = 1 \ldots k$. A graph G has a triangle iff there is no policy σ_1 to ensure winning for the objective $Reach(\mathcal{T})$ from start vertex s.*

Proof. For the correctness of the reduction notice that there is a triangle in the graph G iff there is a path from some vertex v_{1i} in the first copy of G to the same vertex in the fourth copy of G, v_{4i} in P. Player 2 then has a policy to always visit only v_{1i} from the first copy and only v_{4i} from the fourth copy which prevents player 1 from visiting target T_i. □

The size and the construction time of graph Γ, given by Reduction 2, are linear in the size of the original graph G and we have $k = \theta(n)$ target sets.

Theorem 8. *There is no combinatorial $O(n^{3-\epsilon})$ or $O((k \cdot n^2)^{1-\epsilon})$ algorithm (for any $\epsilon > 0$) to check if a vertex v has a winning policy for sequential reachability objectives in game graphs under Conjecture 2 (i.e., unless STC and BMM fail). The bounds hold for dense game graphs with $m = \theta(n^2)$.*

Discussion and Conclusion

In this work, we study several natural planning problems in graphs, MDPs, and game graphs, which are basic algorithmic problems in artificial intelligence. Our main contributions are a sub-quadratic algorithm for sequential target in MDPs, and quadratic conditional lower bounds. Note that graphs are a special case of both MDPs and game graphs, and the algorithmic problems are simplest for graphs, and in all cases, we have linear-time upper bounds. The key highlight of our results is an interesting separation of MDPs and game graphs: for basic target reachability, MDPs are harder than game graphs; for the coverage problem, both MDPs and game graphs are hard (quadratic CLBs); for sequential target reachability, game graphs are harder than MDPs.

Remark 5. *Note that in Table 1 in the upper bounds for MDPs (second column) the term $m \cdot n^{2/3}$ appears consistently, which is the current best-known bound for the MEC-decomposition problem. For all the upper bound results, any improvement for the MEC-decomposition bound also carries over and improves the $m \cdot n^{2/3}$ term in all entries of Table 1. Quite interestingly, for the coverage problem the CLB shows that the $k \cdot m$ term, which is present alongside the MEC-decomposition term, cannot be improved (this gives quadratic CLB), whereas for the sequential target problem, we present a sub-quadratic upper bound for MDPs.*

In this work, we clarified the algorithmic landscape of basic planning problems with CLBs and better algorithms. An interesting direction of future work would be to consider CLBs for other polynomial-time problems in planning and AI in general. For MDPs with sequential targets, we establish sub-quadratic upper bounds, and hence the techniques of the paper that establish quadratic CLBs are not applicable. Other CLB techniques for this problem are an interesting topic to investigate as future work.

Acknowledgments

The authors are grateful to the anonymous referees for their valuable comments and suggestions to improve the presentation of the paper.

A. S. is fully supported by the Vienna Science and Technology Fund (WWTF) through project ICT15-003, the other authors are partially supported by that grant. K.C. is also supported by the Austrian Science Fund (FWF) NFN Grant No S11407-N23 (RiSE/SHiNE) and an ERC Starting grant (279307: Graph Games). For M.H 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.

References

[Abboud and Williams 2014] Abboud, A., and Williams, V. V. 2014. Popular conjectures imply strong lower bounds for dynamic problems. In *FOCS*, 434–443.

[Aghighi et al. 2016] Aghighi, M.; Bäckström, C.; Jonsson, P.; and Ståhlberg, S. 2016. Refining complexity analyses in planning by exploiting the exponential time hypothesis. *Annals of Mathematics and Artificial Intelligence* 78(2):157–175.

[Bäckström and Jonsson 2017] Bäckström, C., and Jonsson, P. 2017. Time and space bounds for planning. *Journal of Artificial Intelligence Research* 60:595–638.

[Ballard et al. 2012] Ballard, G.; Demmel, J.; Holtz, O.; and Schwartz, O. 2012. Graph expansion and communication costs of fast matrix multiplication. *J. ACM* 59(6):32:1–32:23.

[Beeri 1980] Beeri, C. 1980. On the membership problem for functional and multivalued dependencies in relational databases. *ACM Transactions on Database Systems (TODS)* 5(3):241–259.

[Bonet and Geffner 2000] Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In *AIPS*, 52–61.

[Bringmann and Künnemann 2015] Bringmann, K., and Künnemann, M. 2015. Quadratic conditional lower bounds for string problems and dynamic time warping. In *FOCS*, 79–97.

[Chatterjee and Henzinger 2014] Chatterjee, K., and Henzinger, M. 2014. Efficient and dynamic algorithms for alternating büchi games and maximal end-component decomposition. *J. ACM* 61(3):15:1–15:40.

[Chatterjee et al. 2016] Chatterjee, K.; Dvořák, W.; Henzinger, M.; and Loitzenbauer, V. 2016. Model and objective separation with conditional lower bounds: Disjunction is harder than conjunction. In *LICS*, 197–206.

[Choset 2005] Choset, H. M. 2005. *Principles of robot motion: theory, algorithms, and implementation*. MIT press.

[Cimatti et al. 2003] Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence* 147(1):35–84.

[Clarke, Grumberg, and Peled 1999] Clarke, Jr., E. M.; Grumberg, O.; and Peled, D. A. 1999. *Model Checking*. Cambridge, MA, USA: MIT Press.

[Filar and Vrieze 1997] Filar, J. A., and Vrieze, K. 1997. *Competitive Markov Decision Processes*. Springer.

[Hansen and Zilberstein 1998] Hansen, E. A., and Zilberstein, S. 1998. Heuristic search in cyclic and/or graphs. In *AAAI*, 412–418.

[Henzinger et al. 2015] Henzinger, M.; Krinninger, S.; Nanongkai, D.; and Saranurak, T. 2015. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, 21–30. New York, NY, USA: ACM.

[Hoffmann and Brafman 2005] Hoffmann, J., and Brafman, R. 2005. Contingent planning via heuristic forward search with implicit belief states. In *ICAPS*, 71–88.

[Howard 1960] Howard, H. 1960. *Dynamic Programming and Markov Processes*. MIT Press.

[Immerman 1981] Immerman, N. 1981. Number of quantifiers is better than number of tape cells. *Journal of Computer and System Sciences* 22(3):384–406.

[Impagliazzo and Paturi 1999] Impagliazzo, R., and Paturi, R. 1999. Complexity of k-sat. In *CCC*, 237–240.

[Impagliazzo, Paturi, and Zane 1998] Impagliazzo, R.; Paturi, R.; and Zane, F. 1998. Which problems have strongly exponential complexity? In *FOCS*, 653–662.

[Kaelbling, Littman, and Cassandra 1998] Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1):99–134.

[Kress-Gazit, Fainekos, and Pappas 2009] Kress-Gazit, H.; Fainekos, G. E.; and Pappas, G. J. 2009. Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics* 25(6):1370–1381.

[Kronegger, Pfandler, and Pichler 2013] Kronegger, M.; Pfandler, A.; and Pichler, R. 2013. Parameterized complexity of optimal planning: A detailed map. In *IJCAI*, 954–961.

[LaValle 2006] LaValle, S. M. 2006. *Planning algorithms*. Cambridge University Press.

[Le Gall 2014] Le Gall, F. 2014. Powers of tensors and fast matrix multiplication. In *ISSAC*, 296–303.

[Mahanti and Bagchi 1985] Mahanti, A., and Bagchi, A. 1985. AND/OR graph heuristic search methods. *J. ACM* 32(1):28–51.

[Maliah et al. 2014] Maliah, S.; Brafman, R.; Karpas, E.; and Shani, G. 2014. Partially observable online contingent planning using landmark heuristics. In *ICAPS*, 163–171.

[Palacios and Geffner 2007] Palacios, H., and Geffner, H. 2007. From conformant into classical planning: Efficient translations that may be complete too. In *ICAPS*, 264–271.

[Papadimitriou and Tsitsiklis 1987] Papadimitriou, C. H., and Tsitsiklis, J. N. 1987. The complexity of Markov

decision processes. *Mathematics of Operations Research* 12:441–450.

[Puterman 1994] Puterman, M. L. 1994. *Markov Decision Processes*. John Wiley and Sons.

[Russell and Norvig 2010] Russell, S. J., and Norvig, P. 2010. *Artificial Intelligence - A Modern Approach (3rd ed.)*. Pearson Education.

[Tarjan 1972] Tarjan, R. 1972. Depth-first search and linear graph algorithms. *SIAM Journal on Computing* 1(2):146–160.

[Vassilevska-Williams 2018] Vassilevska-Williams, V. 2018. On some fine-grained questions in algorithms and complexity. In *ICM*, to appear.

[Williams and Williams 2010] Williams, V. V., and Williams, R. 2010. Subcubic equivalences between path, matrix and triangle problems. In *FOCS*, 645–654.

[Williams and Williams 2018] Williams, V. V., and Williams, R. 2018. Subcubic equivalences between path, matrix and triangle problems. *J. ACM*. to appear. preliminary version available at `http://people.csail.mit.edu/virgi/tria-mmult-jv.pdf`.

[Williams 2005] Williams, R. 2005. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science* 348(2):357 – 365.

[Williams 2012] Williams, V. V. 2012. Multiplying matrices faster than coppersmith-winograd. In *STOC*, 887–898.