Proceedings of Machine Learning Research vol :1–23, 2026 39th Annual Conference on Learning Theory

Randomization for Faster Exact Optimization of Discounted Markov Decision Processes

Andrei Graur AGRAUR@STANFORD.EDU
Aaron Sidford SIDFORD@STANFORD.EDU
Ta-Wei Tu TAWEITU@STANFORD.EDU

Editors: Steve Hanneke and Tor Lattimore

Abstract

We provide faster running times for exactly solving discounted Markov Decision Processes (DMDPs) in strongly polynomial time. We obtain our results by efficiently reducing computing optimal values and policies in DMDPs to the easier tasks of policy evaluation and computing approximately optimal values. We provide both a straightforward deterministic reduction and a more efficient randomized variant that, together with advances in approximately solving DMDPs, yield our results.

1. Introduction

Markov decision processes (MDPs) are ubiquitous in learning theory. They are a simple, but expressive, model for decision making under uncertainty that forms the basis for more complex learning theoretic problems. Correspondingly, the problem of solving MDPs—and specifically, γ-*discounted* MDPs (γ-*DMDPs*[1] for short)—is a well-studied problem across computer science, operations research, and machine learning. The problem is foundational in reinforcement learning, which considers complex variations of it (see, e.g., Degris et al. (2006); Sigaud and Buffet (2013); Wei et al. (2017)), and the problem is closely related to canonical combinatorial optimization problems; for example, when $\gamma \to 1$ and transitions are deterministic, the problem encompasses the problem of computing a cycle of minimum-mean cost in a directed graph (Madani et al., 2009).

Given the prominence of the problem, there has been extensive research on efficiently solving γ-DMDPs *approximately*. For example, convergence rates have been provided for a variety of methods, including value iteration, policy iteration (Howard, 1960; Bellman, 1966; Puterman, 1994; Bertsekas and Tsitsiklis, 1995), stochastic variants (in the context of MDPs) such as temporal learning (Sutton, 1988; Tsitsiklis and Roy, 2002) and SARSA (Rummery and Niranjan, 1994; Singh et al., 2000), Q-learning (Watkins and Dayan, 1992) (see Sutton et al. (1998) for a survey on these methods), policy gradient (Williams, 1992; Sutton et al., 1999), etc. There has also been extensive research on understanding the sample complexity of the problem (Kakade, 2003; Azar et al., 2012, 2013; Wang, 2017; Sidford et al., 2018; Wainwright, 2019; Agarwal et al., 2020; Jin and Sidford, 2020; Sidford et al., 2023; Jin et al., 2024, 2025).

Relatively less is known about *exactly* solving γ-DMDPs, i.e., computing an optimal policy π_* and the associated *values* $v^{\mathcal{M}}_{\pi_*}$. In this paper, we focus on the *tabular* setting, where we have explicit access to the γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$ that consists of $\mathcal{S}_{\mathrm{tot}} \stackrel{\mathrm{def}}{=} |\mathcal{S}|$ states and a non-zero, non-uniform number of actions at each state which amount to a total of $\mathcal{A}_{\mathrm{tot}} \stackrel{\mathrm{def}}{=} |\mathcal{A}|$ actions (see

1. We note that DMDP is often also used to refer to deterministic Markov decision processes in the literature.

Section 2 for the formal problem definition).[2] Even in light of the above approximate algorithms, exactly solving γ-DMDPs in this setting is non-trivial, as there is no known target accuracy ε (in terms of the problem parameters) for policies and values that suffices to obtain an optimal solution.

Nevertheless, strikingly, Ye (2005) provided what they call a "combinatorial interior point method (IPM)" that solves γ-DMDPs exactly in $O(\mathcal{A}_{\mathrm{tot}}^4 \log \frac{\mathcal{S}_{\mathrm{tot}}}{1-\gamma})$ time. Additionally, the classic policy iteration method of Howard (1960) was shown to solve γ-DMDPs in $O(\frac{\mathcal{A}_{\mathrm{tot}}}{1-\gamma} \log(\frac{1}{1-\gamma}))$ iterations and can be implemented in $O((\frac{\mathcal{S}_{\mathrm{tot}}^{\omega}\mathcal{A}_{\mathrm{tot}}+\mathcal{S}_{\mathrm{tot}}\mathcal{A}_{\mathrm{tot}}^2}{1-\gamma}) \log(\frac{1}{1-\gamma}))$ time (Ye, 2011; Hansen et al., 2013; Scherrer, 2013), where $\omega < 2.372$ is the matrix multiplication constant (Alman et al., 2025). See Table 1 for prior running times for exactly solving γ-DMDPs.

These exact γ-DMDP solvers are of interest for multiple reasons. First, they provide techniques for deducing that certain actions are not in an optimal policy and can thus be discarded. Second, there have been results for solving MDPs in stochastic settings where exact MDP solvers are used (Brunskill et al., 2009; Xu and Tewari, 2020; Rosenberg and Mansour, 2021) (though perhaps this exactness requirement could be relaxed). Third, they efficiently solve a special class of linear programs in strongly polynomial time (see the discussion on exact algorithms for γ-DMDPs and LPs in Section 1.2). Moreover, interestingly, while policy iteration can be viewed as a type of simplex method, particular simplex methods are only known to run in weakly polynomial time (Kelner and Spielman, 2006).

Correspondingly, the goal of this work is to obtain more efficient exact γ-DMDP solvers. We seek both faster algorithms and general frameworks which enable these algorithms.

1.1. Our Results

In this paper we provide more efficient algorithms for solving γ-DMDPs *exactly*. We provide deterministic algorithms with running times that improve upon the prior state of the art and randomized algorithms that further improve these running times by a factor of $\tilde{\Omega}(\mathcal{A}_{\mathrm{tot}}/\mathcal{S}_{\mathrm{tot}})$.[3] For a comparison between prior running times and our improved running times, see Tables 1 and 2.

Perhaps more importantly, we obtain these results through a straightforward framework (Algorithm 1) that reduces solving γ-DMDPs to the easier tasks of *policy evaluation*[4] (i.e., computing $v_{\pi}^{\mathcal{M}}$ for a policy π) which is solvable in $O(\mathcal{S}_{\mathrm{tot}}^{\omega})$ time, and, essentially, solving a γ-DMDP *approximately*.

Our framework is broadly inspired by Ye (2005) which provided an IPM that discards *suboptimal* (i.e., not part of any optimal policy) actions over time. At a high level, our algorithm proceeds similarly but deviates in exactly how it reasons about suboptimal actions and replaces using an IPM with an arbitrary approximate algorithm for solving γ-DMDPs. We essentially apply an analysis similar to Ye (2011); Scherrer (2013) inside this general framework to perform a straightforward analysis of the iteration count. Ultimately, this allows us to obtain faster deterministic algorithms for solving γ-DMDPs by leveraging state-of-the-art approximate solvers (and essentially recovers the analysis of Ye (2005)). Additionally, we show that it is possible to modify this general framework to incorporate randomization; we show that a randomized variant of the framework allows us

2. Some prior work on solving γ-DMDPs instead suppose that there is a fixed number of actions A at every state and provide running times in terms of $\mathcal{S}_{\mathrm{tot}}A$ rather than $\mathcal{A}_{\mathrm{tot}}$. For uniformity of comparison and to reflect how results often generalize, we state running times for such prior work with $\mathcal{A}_{\mathrm{tot}} = \mathcal{S}_{\mathrm{tot}}A$ even if the work does not claim it.
3. Throughout this paper, we use $\tilde{O}(\cdot)$ and $\tilde{\Omega}(\cdot)$ to hide $\mathrm{poly}\log(\mathcal{A}_{\mathrm{tot}})$ factors (but not $\mathrm{poly}\log(1/(1-\gamma))$).
4. Solving the γ-DMDP that contains only the actions of a policy π, by definition, computes v_{π}.

to prove that more actions can be discarded (in expectation) and thereby to decrease the number of iterations.[5]

From Approximate Values to Exact Policies. Our notion of approximately solving γ-DMDPs is a natural scale-invariant definition of computing δ-approximate values as defined in Definition 1 below (where δ is set in the reductions). This definition simply scales the accuracy of the values computed with $\|r\|_\infty$, where $r \in \mathbb{R}^{\mathcal{A}}$ is the vector whose entries are the rewards of each action.

Definition 1 (γ-DMDP Approximations) *We call $v \in \mathbb{R}^{\mathcal{S}}$* ε-optimal *values of γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$ for accuracy $\varepsilon > 0$ if $\|v - v_*^{\mathcal{M}}\|_\infty \le \varepsilon$ and we call a policy π of $\mathcal{M}$* ε-optimal *if $\|v_\pi^{\mathcal{M}} - v_*^{\mathcal{M}}\|_\infty \le \varepsilon$, where $v_*^{\mathcal{M}}$ is the* optimal values *of $\mathcal{M}$ (see Section 2). We call an algorithm a* δ-approximate γ-DMDP solver *for accuracy $\delta \in (0,1)$ if for input γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$ it outputs $\delta\|r\|_\infty$-optimal values v of $\mathcal{M}$.*

We provide deterministic and randomized reductions to approximate γ-DMDP solvers. To present these results we let $\mathcal{T}_{\mathrm{Det}}(\mathcal{S}_{\mathrm{tot}}, \mathcal{A}_{\mathrm{tot}}, \delta_\gamma)$ be the running time of a deterministic δ_γ-approximate γ-DMDP algorithm on input $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$ and let $\mathcal{T}_{\mathrm{Rnd}}(\mathcal{S}_{\mathrm{tot}}, \mathcal{A}_{\mathrm{tot}}, \delta_\gamma)$ be the expected running time of such a randomized algorithm. See Table 2 for the running times for exactly solving γ-DMDPs that we obtain using our reductions and different approximate γ-DMDP algorithms.

Our first reduction is deterministic. At a high level, it shows that given an arbitrary policy π, as long as it is not optimal, an approximate γ-DMDP solve (plus some additional evaluation of policies) suffices to detect one suboptimal action (specifically, from π) in the γ-DMDP. Discarding that action and repeating thus gives us an $O(\mathcal{A}_{\mathrm{tot}})$-iteration reduction.

Theorem 2 (Deterministic Reduction) *There is a deterministic algorithm that solves a γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$ in $O(\mathcal{S}_{\mathrm{tot}}^{\omega}\mathcal{A}_{\mathrm{tot}} + \mathcal{S}_{\mathrm{tot}}\mathcal{A}_{\mathrm{tot}}^2) + O(\mathcal{A}_{\mathrm{tot}}) \cdot \mathcal{T}_{\mathrm{Det}}(\mathcal{S}_{\mathrm{tot}}, \mathcal{A}_{\mathrm{tot}}, \Theta(1-\gamma))$ time.*

State-of-the-art deterministic δ_γ-approximate γ-DMDP solver include an $\tilde{O}(\mathcal{A}_{\mathrm{tot}}^{\omega_{\mathrm{curr}}} \log(\frac{1}{(1-\gamma)\delta_\gamma}))$ time[6] IPM-based algorithm by van den Brand (2020)[7] (using a reduction in Sidford et al. (2023)), where $\omega_{\mathrm{curr}} \approx 2.371339$ (Duan et al., 2023; Alman and Vassilevska Williams, 2021; Gall, 2024; Vassilevska Williams et al., 2024; Alman et al., 2025) is the current value of the matrix multiplication constant, and the well-known value iteration algorithm of Bellman (1966) that runs in $O(\frac{\mathcal{S}_{\mathrm{tot}}\mathcal{A}_{\mathrm{tot}}}{1-\gamma} \log(\frac{1}{(1-\gamma)\delta_\gamma}))$ time (see, e.g., Jin et al. (2024)). Instantiating Theorem 2 with these algorithms implies the following.

Corollary 3 *There is a deterministic algorithm that solves a γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$ in the minimum of $O(\mathcal{S}_{\mathrm{tot}}^{\omega}\mathcal{A}_{\mathrm{tot}} + \frac{\mathcal{S}_{\mathrm{tot}}\mathcal{A}_{\mathrm{tot}}^2}{1-\gamma} \log(\frac{1}{1-\gamma}))$ and $\tilde{O}(\mathcal{A}_{\mathrm{tot}}^{\omega_{\mathrm{curr}}+1} \log(\frac{1}{1-\gamma}))$ time.*

5. It is conceivable that similar improvements for deterministic algorithms could be obtained by incorporating insights of Ye (2005) into recent IPM results. It is also conceivable that one can incorporate our randomization insights within the framework of Ye (2005). It may be useful that we obtain black-box reductions without appealing to IPMs.
6. The actual running time of van den Brand (2020) is $\tilde{O}((\mathcal{A}_{\mathrm{tot}}^{\omega} + \mathcal{A}_{\mathrm{tot}}^{2.5-\alpha/2+o(1)} + \mathcal{A}_{\mathrm{tot}}^{2+1/6+o(1)}) \log(\frac{1}{(1-\gamma)\delta_\gamma}))$, where ω is the matrix multiplication constant and α is the dual matrix exponent (i.e., largest μ so that multiplying a $n \times n^{\mu}$ matrix by a $n^{\mu} \times n$ matrix can be done in $O(n^{2+\varepsilon})$ time for every $\varepsilon > 0$). For the current values of $\omega = \omega_{\mathrm{curr}} \approx 2.371339$ (Alman et al., 2025) and $\alpha = \alpha_{\mathrm{curr}} \gtrsim 0.321334$ (Gall, 2024; Vassilevska Williams et al., 2024), the running time of van den Brand (2020) simplifies to $\tilde{O}(\mathcal{A}_{\mathrm{tot}}^{\omega_{\mathrm{curr}}} \log(\frac{1}{(1-\gamma)\delta_\gamma}))$.
7. Cohen et al. (2019); Lee et al. (2019) obtain similar running times but their algorithms are randomized. Jiang et al. (2021) also obtains the same running time for the best-known value of ω and improves by a polynomial factor over these works if $\omega = 2$.

Paper	Method	Running time
Ye (2005)	Combinatorial IPM	$O(\mathcal{A}_{\text{tot}}^4 \log(\frac{\mathcal{S}_{\text{tot}}}{1-\gamma}))$
Ye (2011)	Simplex PI	$O((\frac{\mathcal{S}_{\text{tot}}^{\omega+1}\mathcal{A}_{\text{tot}}+\mathcal{S}_{\text{tot}}^2\mathcal{A}_{\text{tot}}^2}{1-\gamma})\log(\frac{1}{1-\gamma}))$
Scherrer (2013)	Howard's PI	$O((\frac{\mathcal{S}_{\text{tot}}^{\omega}\mathcal{A}_{\text{tot}}+\mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}}^2}{1-\gamma})\log(\frac{1}{1-\gamma}))$

Table 1: Prior running times for solving γ-DMDPs exactly.

Approximate solver	Implied running time	Deterministic or randomized
Bellman (1966)	$O(\mathcal{S}_{\text{tot}}^{\omega}\mathcal{A}_{\text{tot}} + \frac{\mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}}^2}{1-\gamma}\log(\frac{1}{1-\gamma}))$	Deterministic
van den Brand (2020)	$\tilde{O}(\mathcal{A}_{\text{tot}}^{\omega_{\text{curr}}+1}\log(\frac{1}{1-\gamma}))$	Deterministic
van den Brand (2020)	$\tilde{O}(\mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}}^{\omega_{\text{curr}}}\log(\frac{1}{1-\gamma}))$	Randomized
van den Brand et al. (2021)	$\tilde{O}((\mathcal{S}_{\text{tot}}^2\mathcal{A}_{\text{tot}} + \mathcal{S}_{\text{tot}}^{3.5})\log(\frac{1}{1-\gamma}))$	Randomized
Jin et al. (2025)	$\tilde{O}(\mathcal{S}_{\text{tot}}^{\omega+1} + (\mathcal{S}_{\text{tot}}^2\mathcal{A}_{\text{tot}} + \frac{\mathcal{S}_{\text{tot}}^{1.5}\mathcal{A}_{\text{tot}}}{1-\gamma})\log^{O(1)}(\frac{1}{1-\gamma}))$	Randomized

Table 2: Improved running times for solving γ-DMDPs exactly.

Randomization for Faster Algorithms. While our deterministic reduction removes the reliance of Ye (2005) on using a specific IPM, our best analysis only bounds that it proceeds in a similar $\Theta(\mathcal{A}_{\text{tot}})$ iterations; in our worst case analysis it may only detect one suboptimal action per approximate solve. We improve this bound by showing that instead of detecting suboptimal actions based on an *arbitrary* policy π, if we pick π *uniformly at random* at each iteration, then asymptotically more actions can be discarded in expectation. In particular, this improves the expected iteration count to $\tilde{O}(\mathcal{S}_{\text{tot}})$ which is lower than $O(\mathcal{A}_{\text{tot}})$ whenever there are sufficiently many actions per state.

Theorem 4 (Randomized Reduction) *There is a randomized algorithm that solves a γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$ in $\tilde{O}(\mathcal{S}_{\text{tot}}^{\omega+1} + \mathcal{S}_{\text{tot}}^2\mathcal{A}_{\text{tot}}) + \tilde{O}(\mathcal{S}_{\text{tot}}) \cdot \mathcal{T}_{\text{Rnd}}(\mathcal{S}_{\text{tot}}, \mathcal{A}_{\text{tot}}, \Theta(1-\gamma))$ expected time.*

Note that Theorem 4 requires the approximate γ-DMDP solver to have an expected running time instead of, e.g., succeeding with constant probability. However, in the following Lemma 5 (proved in Appendix B) we show that we can apply fairly well-known techniques and properties of γ-DMDPs to efficiently convert an approximate γ-DMDP algorithm that succeeds with constant probability into one with an expected running time, with a limited loss in approximation.

Lemma 5 *Suppose there is a randomized algorithm that on input γ-DMDP $\mathcal{M}' = (\mathcal{S}', \mathcal{A}', p', r')$ outputs $\delta'\|r'\|_\infty$-optimal values v of $\mathcal{M}$ in time $\mathcal{T}_{\text{MC}}(\mathcal{S}'_{\text{tot}}, \mathcal{A}'_{\text{tot}}, \delta')$ with constant probability. Then, there is a randomized δ-approximate γ-DMDP algorithm that on input γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$ runs in expected time $\mathcal{T}_{\text{Rnd}}(\mathcal{S}_{\text{tot}}, \mathcal{A}_{\text{tot}}, \delta) = O(\mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}} + \mathcal{T}_{\text{MC}}(\mathcal{S}_{\text{tot}}, \mathcal{A}_{\text{tot}}, \Theta(\delta \cdot (1-\gamma)))$.*

For randomized Monte Carlo δ_γ-approximate γ-DMDPs solvers, the state of the art includes an $\tilde{O}((\mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}} + \frac{\sqrt{\mathcal{S}_{\text{tot}}}\mathcal{A}_{\text{tot}}}{1-\gamma})\log^{O(1)}\frac{1}{(1-\gamma)\delta_\gamma})$ time[8] sampling-based value iteration algorithm (Sidford

8. We note that Jin et al. (2025) only claimed a running time of $\tilde{O}((\mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}} + \frac{\sqrt{\mathcal{S}_{\text{tot}}}\mathcal{A}_{\text{tot}}}{1-\gamma})\log^{O(1)}\frac{1}{(1-\gamma)\delta_\gamma})$ when $\mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}} \leq \mathcal{A}_{\text{tot}}(1-\gamma)^2$. However, when $\mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}} > \mathcal{A}_{\text{tot}}/(1-\gamma)^2$, the $\tilde{O}((\mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}} +$

et al., 2023; Jin et al., 2024, 2025) and an $\tilde{O}((\mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}} + \mathcal{S}_{\text{tot}}^{2.5})\log(\frac{1}{(1-\gamma)\delta_\gamma}))$ time IPM-based algorithm (van den Brand et al., 2021). Combining Lemma 5 with these algorithms (as well as that in van den Brand (2020)) and plugging them into Theorem 4, we obtain the following running times for exactly solving γ-DMDPs.

Corollary 6 *There is a randomized algorithm that solves an input γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$ in the minimum of $\tilde{O}(\mathcal{S}_{\text{tot}}^{\omega+1} + (\mathcal{S}_{\text{tot}}^2\mathcal{A}_{\text{tot}} + \frac{\mathcal{S}_{\text{tot}}^{1.5}\mathcal{A}_{\text{tot}}}{1-\gamma})\log^{O(1)}(\frac{1}{1-\gamma}))$, $\tilde{O}((\mathcal{S}_{\text{tot}}^2\mathcal{A}_{\text{tot}} + \mathcal{S}_{\text{tot}}^{3.5})\log(\frac{1}{1-\gamma}))$, and $\tilde{O}(\mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}}^{\omega_{\text{curr}}}\log(\frac{1}{1-\gamma}))$ expected time.*

Lastly, in Appendix A, we show that instantiating Theorem 4 instead with the classic policy iteration algorithm of Howard (1960) results in a simple randomized policy iteration with a running time that improves upon e.g., Scherrer (2013), but does not improve upon applying our framework with the (fastest) sampling-based value iteration methods of Sidford et al. (2023); Jin et al. (2024, 2025).

1.2. Additional Related Work

Approximate Algorithms for γ-DMDPs. There have been numerous advances in obtaining fast approximate algorithms for γ-DMDPs in different parameter regimes. Beyond classic algorithms such as policy and value iterations (Bertsekas, 2012; Bellman, 1966; Howard, 1960; Tseng, 1990; Littman et al., 1995), there are several works that give sampling-based methods for γ-DMDP, some of them in the context of reinforcement learning (Lattimore and Hutter, 2012; Azar et al., 2012, 2013; Sidford et al., 2023; Jin et al., 2024, 2025). Since γ-DMDPs are instances of linear programs, they can also be solved to high accuracy using weakly polynomial time linear programming (LP) algorithms (e.g., using a reduction of Sidford et al. (2023)). High-accuracy algorithms for linear programs have been extensively studied and there has been recent progress in improving their running times (Lee and Sidford, 2014, 2015; Cohen et al., 2019; van den Brand, 2020; van den Brand et al., 2021; Jiang et al., 2021) some of which yield several state-of-the-arts for γ-DMDPs.

Exact Algorithms for γ-DMDPs and LPs. Our work on exact algorithms for γ-DMDPs builds on a line of work on providing strongly polynomial time algorithms for solving γ-DMDPs when the discount factor γ is a constant. Ye (2005) provided the first analysis of such an algorithm for this variant of the problem, followed by Ye (2011); Hansen et al. (2013); Scherrer (2013). Strongly polynomial algorithms can be defined for different models of computation. In the *real RAM* model, an algorithm runs in strongly polynomial time if the number of arithmetic operations it performs does not depend on the bit complexity of the inputs. In the stricter *Turing model*, on the other hand, it is additionally required that the *space complexity* of the algorithm is bounded by $\text{poly}(L)$, where L is the total bit complexity of the inputs.[9] In most of the paper we do not discuss the space complexity; however, in Theorem 18, we briefly discuss our framework's performance in the Turing model.

$\mathcal{A}_{\text{tot}}/(1-\gamma)^2)\log^{O(1)}\frac{1}{(1-\gamma)\delta_\gamma})$ running of Jin et al. (2024) would also be bounded by $\tilde{O}((\mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}} + \frac{\sqrt{\mathcal{S}_{\text{tot}}}\mathcal{A}_{\text{tot}}}{1-\gamma})\log^{O(1)}\frac{1}{(1-\gamma)\delta_\gamma})$. We additionally note that this running time is better than the standard value iteration (Bellman, 1966) in all regimes of parameters.

9. One classic example is that one can compute 2^{2^n} using n multiplications. However, the space complexity of such an algorithm would be 2^n, so it is considered strongly polynomial in real RAM but not the Turing model.

Related to work on strongly polynomial algorithms for solving γ-DMDPs with constant γ is a long sequence of works (Tardos, 1986; Vavasis and Ye, 1996; Dadush et al., 2020b,a) that obtain exact algorithms for LPs whose running times only depend on certain condition numbers of the constraint matrix $\mathbf{A}$ of the linear program (e.g., $\bar{\chi}_{\mathbf{A}}$ or $\bar{\chi}^*_{\mathbf{A}}$). These algorithms are thus strongly polynomial for certain classes of suitably-conditioned LPs, which constitutes progress towards the outstanding open problem of obtaining a strongly polynomial time algorithm for (general) LPs in the Turing model (Megiddo, 1983) (see Dadush et al. (2024) for further discussion). However, when $\mathbf{A}$ is the constraint matrix in the linear programming formulation of solving γ-DMDPs, the quantities $\chi_{\mathbf{A}}$ and $\chi^*_{\mathbf{A}}$ still depend on the bit complexity of the transition probabilities of the γ-DMDP. Hence, one cannot apply the results in Tardos (1986); Vavasis and Ye (1996); Dadush et al. (2020b,a) black-box to obtain strongly polynomial time algorithms for γ-DMDP, even for bounded values of γ (though there might be ways to closely follow these frameworks and obtain such algorithms). Additionally, there is a line of work (Dadush et al., 2024; Allamigeon et al., 2025; Dadush et al., 2026) on designing IPMs whose running time is tightly related (i.e., up to polynomial factors) to a quantity that Allamigeon et al. (2025) call the *straight line complexity (SLC)* of the linear program. Independently of this paper, Natura and Végh (2025) claims that the SLC of the LP formulation of solving a γ-DMDP is $O(\mathrm{poly}(\mathcal{A}_{\mathrm{tot}}) \log(1/(1-\gamma)))$.

We note that some of these works, e.g., Tardos (1986); Dadush et al. (2020b), analogously to this paper, give a reduction from exact to approximate LP algorithms. In fact, Tardos' framework, when applied to γ-DMDPs, proceeds similarly to our deterministic framework; however, we leverage the particular optimality properties specific to γ-DMDPs to obtain our particular bound on the accuracy required for the approximate solves.

Deterministic γ-DMDPs. γ-DMDP instances where $p(s, a) \in \{0, 1\}$ for all state-action pairs (s, a) (i.e., transitions to a new state are deterministic) are called deterministic γ-DMDPs and are well-studied. In particular, strongly polynomial time algorithms that do not depend on the value of γ are known for deterministic γ-DMDPs (Madani et al., 2009; Karczmarz, 2022). Simplex methods are also known to be strongly polynomial time algorithms on deterministic γ-DMDPs (Post and Ye, 2013; Hansen et al., 2014), yet such variants have worse running times than the specialized methods of Madani et al. (2009); Karczmarz (2022).

Simplex Methods and γ-DMDPs. Given the close relationship between policy iteration algorithms and simplex methods for γ-DMDPs (for instance, the version of policy iteration that only switches the action at a single state every time (Ye, 2011) is one type of simplex method for γ-DMDPs), we note that there have been several results on proving lower bounds for certain variants of the simplex methods. In particular, Melekopoglou and Condon (1994); Fearnley (2010); Friedmann et al. (2011); Hansen (2012); Avis and Friedmann (2017); Disser et al. (2023) show that several (deterministic or randomized) variants of the simplex method applied for γ-DMDPs require subexponential (or exponential) time when $\gamma = 1$. Note that in the rest of the paper we only consider the setting where γ is bounded away for 1, and thus the lower bounds in the works mentioned do not contradict our results.

1.3. Paper Organization

The rest of the paper is organized as follows. In Section 2, we establish the necessary notation and preliminaries. In Section 3, we present our framework for reducing the problem of exactly solving a

γ-DMDP to approximately solving several γ-DMDPs and prove our deterministic reduction (Theorem 2). In Section 4, we instantiate our framework to obtain our randomized reduction (Theorem 4). In Appendix A, we again use the framework and the randomized analysis to present our simple randomized algorithm based on the policy iteration algorithm of Howard (1960). In Appendix B we provide proofs that are omitted from the main body of the paper.

2. Preliminaries

General Notation. We write $u \leq v$ (respectively, $u \geq v$) for $u, v \in \mathbb{R}^d$ to indicate $u_i \leq v_i$ (respectively, $u_i \geq v_i$) for all $i \in [d]$.

γ-DMDPs. This paper considers the problem of solving or optimizing a γ-DMDP for $\gamma \in (0,1)$. We specify a γ-DMDP by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$, where $\mathcal{S}$ is a non-empty finite set of *states*, $\mathcal{A} = \{(s,a) \mid s \in \mathcal{S}\}$ is a finite set of *state-action pairs*, $p : \mathcal{A} \to \Delta^{\mathcal{S}}$ for $\Delta^{\mathcal{S}} \stackrel{\text{def}}{=} \{q \in \mathbb{R}^{\mathcal{S}}_{\geq 0} \mid \|q\|_1 = 1\}$ denotes *transition probabilities*, and $r \in \mathbb{R}^{\mathcal{A}}$ denotes *rewards*. We call $\mathcal{A}_s \stackrel{\text{def}}{=} \{a \mid (s,a) \in \mathcal{A}\}$ the *actions* at $s \in \mathcal{S}$ and assume each $\mathcal{A}_s \neq \emptyset$.

An *agent* interacts with a γ-DMDP in time-steps $t = 0, 1, \ldots$. If the agent takes action $a_t \in \mathcal{A}_{s_t}$ at state $s_t \in \mathcal{S}$ at time-step t then the agent receives reward $r_{(s_t,a_t)} \in \mathbb{R}$ and randomly *transitions* or *moves* to state $s_{t+1} \in \mathcal{S}$ sampled independently from distribution $p(s_t, a_t)$ for time-step $t+1$, i.e., $\Pr[s_{t+1} = \bar{s} \mid s_t] = p(s_t, a_t)_{\bar{s}}$.

In a γ-DMDP we optimize over (deterministic stationary) *policies* π which map states $s \in \mathcal{S}$ to actions at that state, i.e., $\pi(s) \in \mathcal{A}_s$. The value of a policy π at state $s \in \mathcal{S}$, denoted by $[v^{\mathcal{M}}_{\pi}]_s$, is the expected γ-discounted reward received by an agent following π starting from state s, i.e.,

$$[v^{\mathcal{M}}_{\pi}]_s \stackrel{\text{def}}{=} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{s_t,\pi(s_t)}\right] \text{ where } s_0 = s \text{ and } \Pr[s_{t+1} = \bar{s} | s_t, \ldots, s_0] = p(s_t, \pi(s_t))_{\bar{s}},\ \forall t \in \mathbb{N}\,, \tag{1}$$

[10]

where the transitions are drawn independently from each other. Let $[v^{\mathcal{M}}_{*}]_s \stackrel{\text{def}}{=} \max_{\pi} [v^{\mathcal{M}}_{\pi}]_s$ be the *optimal values*. It is known that $v^{\mathcal{M}}_{*} = v^{\mathcal{M}}_{\pi_*}$ for some policy π_* (see, e.g., Puterman (1994)); we call such a policy π_* *optimal*.

Bellman Operator To reason about policies and values, we define the *Bellman on-policy operator* $\mathcal{T}^{\mathcal{M}}_{\pi} : \mathbb{R}^{\mathcal{S}} \to \mathbb{R}^{\mathcal{S}}$ with respect to a policy π by $[\mathcal{T}^{\mathcal{M}}_{\pi}(v)]_s \stackrel{\text{def}}{=} r_{s,\pi(s)} + \gamma \cdot \langle p(s,\pi(s)), v\rangle$, and the *Bellman value operator* $T^{\mathcal{M}}_{*} : \mathbb{R}^{\mathcal{S}} \to \mathbb{R}^{\mathcal{S}}$ that maximizes over all actions instead of a fixed policy, i.e., $[T^{\mathcal{M}}_{*}(v)]_s \stackrel{\text{def}}{=} \max_{a \in \mathcal{A}_s} r_{s,a} + \gamma \cdot \langle p(s,a), v\rangle$. It is known that both $\mathcal{T}^{\mathcal{M}}_{\pi}$ and $T^{\mathcal{M}}_{*}$ are contractions and we use the following folklore contraction bounds for γ-DMDPs throughout the paper.

Lemma 7 (see, e.g., (Sidford et al., 2023, Lemmas 3.4 and 3.5)) *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$ be a γ-DMDP. The unique fixpoint of $\mathcal{T}^{\mathcal{M}}_{*}$ is $v^{\mathcal{M}}_{*}$, and for any values $u, v \in \mathbb{R}^{\mathcal{S}}$ it holds that*

$$\|\mathcal{T}^{\mathcal{M}}_{*}(u) - \mathcal{T}^{\mathcal{M}}_{*}(v)\|_\infty \leq \gamma \|u - v\|_\infty \quad \textit{and} \quad \|u - v^{\mathcal{M}}_{*}\|_\infty \leq \frac{1}{1-\gamma} \left\|u - \mathcal{T}^{\mathcal{M}}_{*}(u)\right\|_\infty .$$

10. For notational convenience, we may omit parentheses in subscripts, e.g., we may write $r_{s,a}$ instead of $r_{(s,a)}$.

Note that the above inequalities apply to $\mathcal{T}_\pi^{\mathcal{M}}$ and $v_\pi^{\mathcal{M}}$ as well by simply considering the γ-DMDP with only actions in π, i.e.,

$$\|\mathcal{T}_\pi^{\mathcal{M}}(u) - \mathcal{T}_\pi^{\mathcal{M}}(v)\|_\infty \leq \gamma \|u - v\|_\infty \quad \text{and} \quad \|u - v_\pi^{\mathcal{M}}\|_\infty \leq \frac{1}{1-\gamma} \left\|u - \mathcal{T}_\pi^{\mathcal{M}}(u)\right\|_\infty . \tag{2}$$

Advantage Function Values. To reason about the suboptimality of values and actions, we define, for values $v \in \mathbb{R}^S$, the *advantage function values* $\Delta^{\mathcal{M}}(v) \in \mathbb{R}^{\mathcal{A}}$ with respect to v as $\Delta^{\mathcal{M}}(v)_{s,a} \stackrel{\text{def}}{=} (r_{s,a} + \gamma \cdot \langle p(s,a), v\rangle) - v_s$ for every $(s,a) \in \mathcal{A}$.[11] For brevity, throughout the paper, we often use *advantage(s)* to refer to advantage function value(s). Note that $\Delta^{\mathcal{M}}(v)_{s,\pi(s)} = [\mathcal{T}_\pi^{\mathcal{M}}(v)]_s - v_s$ by definition. Additionally, since $v_*^{\mathcal{M}}$ is the unique fixpoint of $\mathcal{T}_*^{\mathcal{M}}$ (Lemma 7) and $\mathcal{T}_*^{\mathcal{M}}$ maximizes over all actions, $\Delta^{\mathcal{M}}(v_*^{\mathcal{M}})_{s,a} \leq 0$, i.e., all actions have non-positive advantage at $v_*^{\mathcal{M}}$. Moreover, since $v_\pi^{\mathcal{M}}$ is the unique fixpoint of $\mathcal{T}_\pi^{\mathcal{M}}$, the policy π is optimal if and only if every action has zero advantage with respect to $v_*^{\mathcal{M}}$, i.e., $\Delta^{\mathcal{M}}(v_*^{\mathcal{M}})_{s,\pi(s)} = 0$ for all $s \in \mathcal{S}$.

Discarding Actions. For γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$ and $R \subseteq \mathcal{A}$, we write $\mathcal{M} \setminus R$ as removing the actions R from $\mathcal{M}$. More precisely, it is the γ-DMDP $\mathcal{M}' = (\mathcal{S}, \mathcal{A} \setminus R, p', r')$ such that p' and r' are p and r restricted to $\mathcal{A} \setminus R$, respectively.

Notational Simplification. Throughout the paper, when the γ-DMDP $\mathcal{M}$ is clear from context, we may omit $\mathcal{M}$ in the superscript in our notation (e.g., we may write v_* instead of $v_*^{\mathcal{M}}$).

3. From Approximate Values to Exact Policies

In this section, we present our general framework (Algorithm 1) for solving γ-DMDPs by reducing the problem to policy evaluations and computing approximate values (as mentioned in Section 1.1). We first analyze the framework and then instantiate it to prove Theorem 2.

At a high level, our framework for solving γ-DMDPs iteratively discards *suboptimal actions*, i.e., actions not chosen by any optimal policy. This is a common technique for obtaining strongly polynomial time algorithms. For example, in linear programming, irrelevant constraints are discarded (see, e.g., Dadush et al. (2020b)), and in submodular function minimization, elements that are deduced to not be in any minimizers are discarded (see, e.g., Dadush et al. (2021)). For γ-DMDPs, the IPM of Ye (2005) discards suboptimal actions, and Ye (2011) argues that iterations of policy iteration will no longer consider certain suboptimal actions after a few iterations.

To discard suboptimal actions, our framework starts from a policy π, computes ε-optimal values v, for a suitable value of ε, and then considers the advantage function $\Delta(v)_{s,a} = (r_{s,a} + \gamma \cdot \langle p(s,a), v\rangle) - v_s$ with respect to v to detect such suboptimal actions. In particular, using known arguments for reasoning about γ-DMDPs we can show that if $\Delta(v)_{s,a}$ is sufficiently negative, i.e., less than $-\varepsilon(1+\gamma)$, then no optimal policy π_* contains (s,a). To reason about what value of ε suffices to guarantee that at least one (s,a) satisfies $\Delta(v)_{s,a} < -\varepsilon(1+\gamma)$ and can thus be discarded, we follow an analysis similar to Scherrer (2013) and note that for any policy π, we can relate the most negative $\Delta(v_*)_{s,\pi(s)}$ to the suboptimality $\|v_\pi - v_*\|_\infty$ of π (see, e.g., Lemma 7). Consequently, by relating $\Delta(v_*)_{s,\pi(s)}$ to $\Delta(v)_{s,\pi(s)}$, we observe that it suffices to compute ε-optimal values v for ε on the order of the suboptimality of π up to $\mathrm{poly}(1-\gamma)$ factors (see Lemma 9).

To compute ε-optimal values v for $\varepsilon \leq \|v_\pi - v_*\|_\infty \cdot \mathrm{poly}(1-\gamma)$, we develop and apply an algorithm `MultValApprox`, which can be thought of as an algorithm for computing approximate

11. This coincides with the natural notion of *slack* when expressing γ-DMDPs as a linear program.

values to multiplicative accuracy relative to the suboptimality of π. `MultValApprox` first computes v_π. It is known that evaluating the values v_π can be done in $O(\mathcal{S}_{\mathrm{tot}}^{\omega})$ by solving a linear system.

Fact 8 (Policy Evaluation in Fast Matrix Multiplication Time) *There is an algorithm that, given any policy π in γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$, outputs $v_\pi^{\mathcal{M}}$ in $O(\mathcal{S}_{\mathrm{tot}}^{\omega})$ time.*

Proof It is known that $v_\pi^{\mathcal{M}} = (\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1} r_\pi$, where $\mathbf{P}_\pi \in \mathbb{R}_{\geq 0}^{\mathcal{S} \times \mathcal{S}}$ is such that $[\mathbf{P}_\pi]_{ij} \stackrel{\mathrm{def}}{=} p(i, \pi(i))_j$ for all $i, j \in \mathcal{S}$ and $r_\pi \in \mathbb{R}^{\mathcal{S}}$ is such that $[r_\pi]_i \stackrel{\mathrm{def}}{=} r_{i,\pi(i)}$ for all $i \in \mathcal{S}$ (see, e.g., Puterman (1994)). Thus, $v_\pi^{\mathcal{M}}$ can be computed by solving a linear system which takes $O(\mathcal{S}_{\mathrm{tot}}^{\omega})$ time. ■

After the evaluation, `MultValApprox` discards actions with very negative advantages, meaning advantages more negative than $-\Delta_{\max}^{\mathcal{M}}(v_\pi) \cdot (1+\gamma)$, where

$$\Delta_{\max}^{\mathcal{M}}(v_\pi) \stackrel{\mathrm{def}}{=} \max_{(s,a) \in \mathcal{A}} \Delta^{\mathcal{M}}(v_\pi)_{s,a} \tag{3}$$

as they are not chosen by any optimal policy (see Claim 16). Next, it shifts rewards using v_π by considering a new γ-DMDP $\mathcal{M}'$ where the rewards of remaining actions are replaced by the advantages $\Delta^{\mathcal{M}}(v_\pi)$. Finally, it applies a δ_γ-approximate γ-DMDP algorithm ApxALG for $\delta_\gamma = \mathrm{poly}(1-\gamma)$ to $\mathcal{M}'$, which yields ε-optimal values v for $\mathcal{M}$ for explicit $\varepsilon = O(\Delta_{\max}^{\mathcal{M}}(v_\pi) \cdot \mathrm{poly}(1-\gamma)) = O(\|v_\pi - v_*\|_\infty \cdot \mathrm{poly}(1-\gamma))$ (see Lemma 12).

Algorithm 1 Solving γ-DMDPs Exactly Using an Approximate γ-DMDP Solver

Function `SolveDMDP`($\mathcal{M}$)**:**
- **for** $t = 0, 1, \ldots$ **do**
  - `// In the randomized reductions considered in Section 4, we sample a uniformly random` π.
  - Select a policy π of $\mathcal{M}$.
  - `// Compute` ε`-optimal values for` $\varepsilon \leq \frac{1-\gamma}{3(1+\gamma)} \cdot \|v_\pi - v_*\|_\infty$ `(see Section 3.2).`
  - $(v, \varepsilon) \leftarrow$ `MultValApprox`$(\mathcal{M}, \pi)$.
  - `// Use` v `to detect and discard suboptimal actions,` D.
  - $\mathcal{M} \leftarrow \mathcal{M} \setminus D$ where $D = \{(s,a) \in \mathcal{A} : \Delta^{\mathcal{M}}(v)_{s,a} < -\varepsilon(1+\gamma)\}$. `// discard` D
  - `// If no actions are discarded, the policy must be optimal.`
  - **if** $D = \emptyset$ **then return** π `// For analysis define` $T \stackrel{\mathrm{def}}{=} t$ `on this line`
- **end**

Function `MultValApprox`($\mathcal{M}, \pi$)**:**
- $X^\pi \leftarrow \{(s,a) \in \mathcal{A} : \Delta^{\mathcal{M}}(v_\pi)_{s,a} < -\Delta_{\max}^{\mathcal{M}}(v_\pi) \cdot (1+\gamma)\}$. `// actions to ignore`
- Compute $v' \leftarrow \mathsf{ApxALG}(\mathcal{M}[\Delta^{\mathcal{M}}(v_\pi)] \setminus X^\pi, \delta_\gamma)$ where $\delta_\gamma \stackrel{\mathrm{def}}{=} \frac{1-\gamma}{3(1+\gamma)^2}$, where ApxALG is a δ_γ-approximate γ-DMDP solver (see Definition 1).
- **return** (v, ε) *where* $v \leftarrow v' + v_\pi^{\mathcal{M}}$ *and* $\varepsilon \leftarrow \Delta_{\max}^{\mathcal{M}}(v_\pi) \cdot \frac{1-\gamma}{3(1+\gamma)}$.

In the remainder of this section we break down our analysis of Algorithm 1 (and the use of it to prove Theorem 2) into multiple parts. First, in Section 3.1, we prove Lemma 9, which shows that solving the γ-DMDP to an accuracy proportional to the suboptimality of a policy π, meaning $\|v_* - v_\pi\|_\infty$, suffices to detect actions that have a negative enough advantage and can thus be discarded (as they are provably not part of any optimal solution). In Section 3.2 we then analyze `MultValApprox`. Finally, in Section 3.3, we leverage the two parts to prove the correctness of our reduction framework and bound the running time of each iteration (see Theorem 17). Finally, Theorem 2 follows from this analysis straightforwardly.

3.1. Detecting Suboptimal Actions

In this section, we state and prove Lemma 9 which provides a bound on the accuracy to which it suffices to solve for the approximately optimal values to detect suboptimal actions. Similar bounds and analysis appeared in Scherrer (2013), and we provide a self-contained proof for completeness and to enable our specific algorithms.

Our analysis, Scherrer (2013), and Lemma 9 all connect the suboptimality of a policy π to a key quantity $\Delta^{\mathcal{M}}_{*,\min}(\pi)$ defined as the minimum advantage value of the optimum values for a state-action pair in π, i.e.,

$$\Delta^{\mathcal{M}}_{*,\min}(\pi) \stackrel{\text{def}}{=} \min_{s\in\mathcal{S}} \Delta^{\mathcal{M}}(v^{\mathcal{M}}_*)_{s,\pi(s)} \tag{4}$$

Although we do not compute $\Delta^{\mathcal{M}}_{*,\min}(\pi)$ exactly in Algorithm 1, we use it in our analysis to bound the progress made by our algorithm in discarding actions. In particular, we leverage in the analysis of our framework (Algorithm 1) that when v, π satisfy that $\|v - v^{\mathcal{M}}_*\|_\infty$ is small enough relative to $\|v^{\mathcal{M}}_\pi - v^{\mathcal{M}}_*\|_\infty$, any $(s,a) \in \mathcal{A}$ with $\Delta^{\mathcal{M}}(v^{\mathcal{M}}_*)_{s,a} \le \Delta^{\mathcal{M}}_{*,\min}(\pi)$ will be discarded as it does not belong to the set of optimal actions. Additionally, Lemma 9 implies that solving up to accuracy $\varepsilon < \|v^{\mathcal{M}}_\pi - v^{\mathcal{M}}_*\|_\infty \cdot \frac{1-\gamma}{2(1+\gamma)}$ suffices to be able to discard at least one action this way.

Lemma 9 (Scherrer (2013)) *Let $v \in \mathbb{R}^{\mathcal{S}}$ be ε-optimal for γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$. If $\Delta^{\mathcal{M}}(v)_{s,a} < -\varepsilon(1+\gamma)$ for some $(s,a) \in \mathcal{A}$, no optimal policy π_* of $\mathcal{M}$ has $\pi_*(s) = a$. Moreover, if $\varepsilon < \|v^{\mathcal{M}}_\pi - v^{\mathcal{M}}_*\|_\infty \cdot \frac{1-\gamma}{2(1+\gamma)}$, then any $(s,a) \in \mathcal{A}$ with $\Delta^{\mathcal{M}}(v^{\mathcal{M}}_*)_{s,a} \le \Delta^{\mathcal{M}}_{*,\min}(\pi)$ will satisfy $\Delta^{\mathcal{M}}(v)_{s,a} < -\varepsilon(1+\gamma)$.*

To prove Lemma 9, we provide two helper lemmas, Lemmas 10 and 11, whose proofs we defer to Appendix B.

Lemma 10 *For a γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$ and any $u, v \in \mathbb{R}^{\mathcal{S}}$ and $(s,a) \in \mathcal{A}$,*

$$\left|\Delta^{\mathcal{M}}(u)_{s,a} - \Delta^{\mathcal{M}}(v)_{s,a}\right| \le (1+\gamma)\|u - v\|_\infty.$$

Lemma 11 *For any policy π in γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$, $\Delta^{\mathcal{M}}_{*,\min}(\pi) \le -(1-\gamma)\|v^{\mathcal{M}}_\pi - v^{\mathcal{M}}_*\|_\infty$.*

We are now ready to prove Lemma 9.

Proof By Lemma 10, for any $(s,a) \in \mathcal{A}$, we have

$$|\Delta(v_*)_{s,a} - \Delta(v)_{s,a}| \le (1+\gamma)\|v_* - v\|_\infty \le (1+\gamma)\varepsilon.$$

Hence, $\Delta(v)_{s,a} < -\varepsilon(1+\gamma)$ implies $\Delta(v_*)_{s,a} < 0$, and thus no optimal policy π of $\mathcal{M}$ has $\pi(s) = a$. Next, by Lemma 11, any action (s,a) with $\Delta(v_*)_{s,a} \le \Delta_{*,\min}(\pi)$ has $\Delta(v_*)_{s,a} \le -\eta(1-\gamma)$. This in turns by Lemma 10 shows that $\Delta(v)_{s,a} \le -\eta(1-\gamma) + (1+\gamma)\varepsilon$. For $\varepsilon < \eta \cdot \frac{1-\gamma}{2(1+\gamma)}$, we have $-\eta(1-\gamma) + (1+\gamma)\varepsilon < -\varepsilon(1+\gamma)$ which concludes the proof. ∎

3.2. Approximate Values via Reward Shifting

In this section, we state and prove Lemma 12, which provides the correctness and efficiency guarantee of `MultValApprox`, whose pseudocode was given in Algorithm 1. As discussed, this subroutine is applied to obtain approximate values v of a γ-DMDP to the accuracy required by Lemma 9.

Lemma 12 *Given a policy π of γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$, the `MultValApprox` subroutine in Algorithm 1 computes $\varepsilon \leq \|v_\pi^{\mathcal{M}} - v_*^{\mathcal{M}}\|_\infty \cdot \frac{1-\gamma}{3(1+\gamma)}$ and ε-optimal values v and can be implemented in deterministic $O(\mathcal{S}_{\text{tot}}^\omega + \mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}}) + \mathcal{T}_{\text{Det}}(\mathcal{S}_{\text{tot}}, \mathcal{A}_{\text{tot}}, \Theta(1-\gamma))$ time and can be implemented in randomized $O(\mathcal{S}_{\text{tot}}^\omega + \mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}}) + \mathcal{T}_{\text{Rnd}}(\mathcal{S}_{\text{tot}}, \mathcal{A}_{\text{tot}}, \Theta(1-\gamma))$ expected time.*

To motivate the `MultValApprox` subroutine, note that it is not immediately clear how to efficiently run the approximate γ-DMDP solvers on the original γ-DMDP instance to accuracy $\|v_\pi - v_*\|_\infty \cdot \text{poly}(1-\gamma)$ to enable applying Lemma 9. This is because the ratio $\frac{\|r\|_\infty}{\|v_\pi - v_*\|_\infty}$ can be very large, and the running time of the approximate solver depends on this ratio.

To bypass this obstacle, the `MultValApprox` subroutine appropriately modifies the rewards in the γ-DMDP instance that the solvers run on (as aforementioned). In particular, the first step of `MultValApprox` is to shift the reward so that the actions taken by the policy π become the "baseline", i.e., have value 0. This is done by using the advantage function $\Delta(v_\pi)$ with respect to v_π as the reward instead. Note that given π, $\Delta(v_\pi)_{s,a}$ can be computed in $O(\mathcal{S}_{\text{tot}}^\omega + \mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}})$ time for all $(s, a) \in \mathcal{A}$ using Fact 8. The shifted MDP is formally defined as follows.

Definition 13 (Shifted MDP) *For a γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$ and a policy π of $\mathcal{M}$, $M^\pi \stackrel{\text{def}}{=} (\mathcal{S}, \mathcal{A}, p, \Delta^{\mathcal{M}}(v_\pi^{\mathcal{M}}))$ is the γ-DMDP that has the same states, actions, and transition probabilities as $\mathcal{M}$, but with reward vector $\Delta^{\mathcal{M}}(v_\pi^{\mathcal{M}})$ instead of r.*

To prove Lemma 12, we provide a helper lemma and two claims, whose proofs we defer to Appendix B. We first show that, for any policy π', the value vector of π' in $\mathcal{M}^\pi$ is merely the value vector of π' in $\mathcal{M}$ shifted by v_π.

Lemma 14 *For any policies π and π' of γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$, $v_{\pi'}^{\mathcal{M}} = v_{\pi'}^{\mathcal{M}^\pi} + v_\pi^{\mathcal{M}}$, and therefore $v_*^{\mathcal{M}} = v_*^{\mathcal{M}^\pi} + v_\pi^{\mathcal{M}}$.*

This shows that the $\mathcal{M}^\pi$ has the same optimal policy, and thus we can focus on identifying suboptimal actions in $\mathcal{M}^\pi$. Observe that $\Delta^{\mathcal{M}}(v_\pi)_{s,\pi(s)} = 0$ for all states $s \in \mathcal{S}$. That is, each state has at least one action with reward 0. Recall the definition of $\Delta_{\max}^{\mathcal{M}}(v_\pi)$ in (3) and note that $\Delta_{\max}^{\mathcal{M}}(v_\pi) \geq 0$. Next, we provide Claim 15 and Claim 16, whose proofs we defer to Appendix B.

Claim 15 *For every policy π of γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$, $v_\pi^{\mathcal{M}^\pi} = \mathbf{0}$ and $\|v_*^{\mathcal{M}^\pi}\|_\infty \geq \Delta_{\max}^{\mathcal{M}}(v_\pi)$.*

Claim 15 implies that actions with sufficiently negative rewards do not belong to any optimal policy and can thus be ignored. This allows us to bound the range of the rewards that we need to consider for the approximate value optimization.

Claim 16 *For any policy π of γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$, no optimal policy π_* of $\mathcal{M}$ has $\pi_*(s) = a$ for $\Delta^{\mathcal{M}}(v_\pi)_{s,a} < -\Delta_{\max}^{\mathcal{M}}(v_\pi) \cdot (1+\gamma)$.*

Based on Claim 16, we let $X^\pi \stackrel{\text{def}}{=} \{(s,a) : \Delta^{\mathcal{M}}(v_\pi)_{s,a} < -\Delta^{\mathcal{M}}_{\max}(v_\pi) \cdot (1+\gamma)\}$ be the set of actions that have very negative advantages in $\mathcal{M}$ and can be ignored. We now prove Lemma 12.

Proof [Proof of Lemma 12] We first bound the runtime of `MultValApprox`. `MultValApprox` first computes X^π which can be done in $O(\mathcal{S}^\omega_{\text{tot}} + \mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}})$ time by calculating v_π and $\Delta^{\mathcal{M}}(v_\pi)_{s,a}$ for all $(s,a) \in \mathcal{A}$ (which is also used in computing the return value of $\varepsilon \stackrel{\text{def}}{=} \Delta^{\mathcal{M}}_{\max}(v_\pi) \cdot \frac{1-\gamma}{3(1+\gamma)}$). Since $\frac{1-\gamma}{3(1+\gamma)^2} = \Theta(1-\gamma)$, the call to a δ_γ-approximate solver can be implemented in $\mathcal{T}_{\text{Det}}(\mathcal{S}_{\text{tot}}, \mathcal{A}_{\text{tot}}, \Theta(1-\gamma))$ deterministic time or $\mathcal{T}_{\text{Rnd}}(\mathcal{S}_{\text{tot}}, \mathcal{A}_{\text{tot}}, \Theta(1-\gamma))$ randomized expected time, which establishes the running time of `MultValApprox`.

We now prove the correctness of `MultValApprox`. Let $\mathcal{M}' \stackrel{\text{def}}{=} \mathcal{M}^\pi \setminus X^\pi$, i.e., the γ-DMDP that only contains actions in $\mathcal{A} \setminus X^\pi$ and has the reward vector $\Delta^{\mathcal{M}}(v_\pi)$. Note that since $\delta_\gamma \stackrel{\text{def}}{=} \frac{1-\gamma}{3(1+\gamma)^2} = \frac{\varepsilon}{\Delta^{\mathcal{M}}_{\max}(v_\pi)\cdot(1+\gamma)}$ and the maximum magnitude of the reward in $\mathcal{M}'$ is bounded by $\Delta^{\mathcal{M}}_{\max}(v_\pi) \cdot (1+\gamma)$ by definition of X^π, the value vector v' it outputs satisfies $\|v' - v^{\mathcal{M}'}_*\|_\infty \le \varepsilon$ (see Definition 1 for the definition of an δ-approximate algorithm). Thus, it follows from Lemma 14 and Claim 16 that v is ε-optimal for the original γ-DMDP $\mathcal{M}$. Lastly, note that by Claim 15 and Lemma 14 we know that $\|v^{\mathcal{M}}_* - v^{\mathcal{M}}_\pi\|_\infty \ge \Delta^{\mathcal{M}}_{\max}(v_\pi)$, which implies $\varepsilon \le \|v^{\mathcal{M}}_\pi - v^{\mathcal{M}}_*\|_\infty \cdot \frac{1-\gamma}{3(1+\gamma)}$, so ε satisfies our requirement. This concludes the proof. ■

3.3. Putting Everything Together

Here we use Lemmas 9 and 12 to prove the correctness and efficiency of Algorithm 1, our reduction from exact γ-DMDP solvers to approximate γ-DMDP solvers. This is captured by Theorem 17 below. We then use Theorem 17 to prove Theorem 2.

Theorem 17 (Correctness and Efficiency of Algorithm 1) *Given a γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$, Algorithm 1 computes an optimal policy π of $\mathcal{M}$ and each iteration of the for loop can be implemented in $O(\mathcal{S}^\omega_{\text{tot}} + \mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}}) + \mathcal{T}_{\text{Det}}(\mathcal{S}_{\text{tot}}, \mathcal{A}_{\text{tot}}, \Theta(1-\gamma))$ deterministic time and $O(\mathcal{S}^\omega_{\text{tot}} + \mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}}) + \mathcal{T}_{\text{Rnd}}(\mathcal{S}_{\text{tot}}, \mathcal{A}_{\text{tot}}, \Theta(1-\gamma))$ randomized expected time.*

Proof First, we prove the correctness of Algorithm 1. For this, first note that by Lemma 9, every $(s,a) \in D$, for D defined in Algorithm 1, satisfies that $\pi(s) \neq a$ for all optimal policies π of $\mathcal{M}$ since v is ε-optimal by Lemma 12. Next, we show that if $\Delta^{\mathcal{M}}_{*,\min}(\pi) < 0$ in some iteration of the for loop in Algorithm 1, $D \supseteq D^{\mathcal{M}}(\pi)$, for $D^{\mathcal{M}}(\pi) \stackrel{\text{def}}{=} \{(s,a) \in \mathcal{A} : \Delta^{\mathcal{M}}(v_*)_{s,a} \le \Delta^{\mathcal{M}}_{*,\min}(\pi)\}$. For this, note that if $\Delta^{\mathcal{M}}_{*,\min}(\pi) < 0$, then $\|v_\pi - v_*\|_\infty > 0$, and thus $\varepsilon < \|v_\pi - v_*\|_\infty \cdot \frac{1-\gamma}{2(1+\gamma)}$. By Lemma 9, any (s,a) with $\Delta^{\mathcal{M}}(v_*)_{s,a} \le \Delta^{\mathcal{M}}_{*,\min}(\pi)$ must have $\Delta^{\mathcal{M}}(v)_{s,a} < -\varepsilon(1+\gamma)$. Therefore, in this case we must have $D \supseteq D^{\mathcal{M}}(\pi) = \{(s,a) \in \mathcal{A} : \Delta^{\mathcal{M}}(v_*)_{s,a} \le \Delta^{\mathcal{M}}_{*,\min}(\pi)\}$, and so $|D| \ge 1$. Consequently, if $D = \emptyset$ in some iteration of the for loop in Algorithm 1, we have $\Delta_{*,\min}(\pi) = 0$, since otherwise the $(s, \pi(s))$ with $\Delta(v_*)_{s,\pi(s)} = \Delta_{*,\min}(\pi)$ will be in D. This shows that whenever Algorithm 1 terminates, the policy π must have $\Delta_{*,\min}(\pi) = 0$, i.e., is optimal.

Now, we bound the running time of Algorithm 1. To do so, we bound the running time of each iteration of the for loop in Algorithm 1. Note that selecting a policy π takes $O(\mathcal{A}_{\text{tot}})$ time. $O(\mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}})$ time, respectively. Computing approximate values, by Lemma 12, takes $O(\mathcal{S}^\omega_{\text{tot}} + \mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}}) + \mathcal{T}_{\text{Det}}(\mathcal{S}_{\text{tot}}, \mathcal{A}_{\text{tot}}, \Theta(1-\gamma))$ time or $O(\mathcal{S}^\omega_{\text{tot}} + \mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}}) + \mathcal{T}_{\text{Rnd}}(\mathcal{S}_{\text{tot}}, \mathcal{A}_{\text{tot}}, \Theta(1-\gamma))$ randomized expected time. Computing D in Algorithm 1 takes $O(\mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}})$ time by computing

$\Delta(v)_{s,a}$ for all $(s,a) \in \mathcal{A}$. Thus, overall, the (expected) running time per iteration is bounded by $O(\mathcal{S}_{\text{tot}}^{\omega} + \mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}}) + \mathcal{T}_{\text{Det}}(\mathcal{S}_{\text{tot}}, \mathcal{A}_{\text{tot}}, \Theta(1-\gamma))$ or $O(\mathcal{S}_{\text{tot}}^{\omega} + \mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}}) + \mathcal{T}_{\text{Rnd}}(\mathcal{S}_{\text{tot}}, \mathcal{A}_{\text{tot}}, \Theta(1-\gamma))$. ■

Remark 18 *It is easy to see that our framework itself runs in strongly polynomial time even in the stricter Turing model (i.e., the space complexity of our framework is polynomial of the input bit complexity). In particular, our framework first evaluates v_π, which can be implemented in $O(\mathcal{S}_{\text{tot}}^{\omega})$ strongly polynomial time in the Turing model (see, e.g., Storjohann (2005)). It then only performs a constant number of basic operations (e.g., addition, subtraction, multiplication) involving v_π and the inputs. This includes multiplication with $1/(1-\gamma)$ which has the same bit complexity as the input value γ.*

However, since the framework invokes a black-box approximate solver (e.g., the recent linear programming algorithms), we do not formally claim our end-to-end exact algorithms run in strongly polynomial time in the Turing model, as bounding the bit complexity of the operations performed by these approximate solvers is beyond the scope of this paper. That being said, we remark that there has been some recent work (Ghadiri et al., 2023) that does this for a class of continuous optimization methods and some linear programming algorithms.

We now prove our theorem regarding the deterministic reduction, which we restate below.

Theorem 2 (Deterministic Reduction) *There is a deterministic algorithm that solves a γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$ in $O(\mathcal{S}_{\text{tot}}^{\omega}\mathcal{A}_{\text{tot}} + \mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}}^2) + O(\mathcal{A}_{\text{tot}}) \cdot \mathcal{T}_{\text{Det}}(\mathcal{S}_{\text{tot}}, \mathcal{A}_{\text{tot}}, \Theta(1-\gamma))$ time.*

Proof We prove that Algorithm 1 respects the desired properties. Note that its correctness and running time follow from Theorem 17 provided that the number of iterations T of Algorithm 1 satisfies $T = O(\mathcal{A}_{\text{tot}})$. However, $T = O(\mathcal{A}_{\text{tot}})$ as in every iteration of the for loop in which the algorithm does not terminate, the set D contains at least one action, and therefore the number of actions in $\mathcal{M}$ decreases by at least 1. ■

4. Randomized Reduction

In this section, we instantiate our framework Algorithm 1 to obtain our randomized reduction, with correctness and efficiency guarantees stated in Theorem 4. In particular, instead of choosing an arbitrary policy π in Algorithm 1 of Algorithm 1, we select a policy uniformly at random from the remaining policies (i.e., policies which do not choose actions already discarded), i.e., each remaining policy has an equal probability of being selected. This approach is motivated by the fact that the worst-case running time for the algorithm we give for Theorem 2 is achieved when only one action is discarded at a time, which intuitively corresponds to picking the "most suboptimal" policy π in each iteration. We show that picking a random policy instead decreases the number of iterations of Algorithm 1 in expectation.

To analyze the iteration count when randomly selecting π in Algorithm 1, we consider the number of $(s,a) \in \mathcal{A}$ with $\Delta^{\mathcal{M}}(v_*^{\mathcal{M}})_{s,a} \leq \Delta_{*,\min}^{\mathcal{M}}(\pi) = \min_{s'} \Delta^{\mathcal{M}}(v_*^{\mathcal{M}})_{s',\pi(s')}$ (i.e., all actions with more negative advantages than "worst" action chosen by π) for policy π picked uniformly at random. It is easy to see that obtaining ε-optimal values v for the value of ε chosen in Algorithm 1 allows us, by Lemma 9, to discard all aforementioned actions. To reason about how many such actions we can discard, we let $\Pi(\mathcal{M})$ be the set of all policies of $\mathcal{M}$, define a potential $\Phi(\mathcal{M}) \stackrel{\text{def}}{=} |\Pi(\mathcal{M})| = \prod_s |\mathcal{A}_s|$ to be the number of policies, and let $\pi \sim \Pi(\mathcal{M})$ denote sampling π uniformly at random from $\Pi(\mathcal{M})$. In Lemma 19, we show that, in expectation for $\pi \sim \Pi(\mathcal{M})$, $\Phi(\mathcal{M})$ drops by a factor of two after discarding actions more suboptimal than $\Delta^{\mathcal{M}}_{*,\min}(\pi)$. Consequently, this allows us to reduce the number of calls to our procedure from $\mathcal{A}_{\text{tot}} - \mathcal{S}_{\text{tot}}$ to $O(\mathcal{S}_{\text{tot}} \log \mathcal{A}_{\text{tot}})$ in expectation. The proof of Lemma 19 is deferred to Appendix B.

Lemma 19 *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$ be a γ-DMDP where $\Phi(\mathcal{M}) > 0$. Then, letting $D^{\mathcal{M}}(\pi) \stackrel{\text{def}}{=} \{(s,a) : \Delta^{\mathcal{M}}(v^{\mathcal{M}}_*)_{s,a} \leq \Delta^{\mathcal{M}}_{*,\min}(\pi)\}$ for policy π we have $\mathbb{E}_{\pi\sim\Pi(\mathcal{M})}[\Phi(\mathcal{M} \setminus D^{\mathcal{M}}(\pi))] \leq \frac{1}{2}\Phi(\mathcal{M})$.*

We now use Lemma 19 to prove Theorem 4, the theorem regarding our randomized reduction, which we restate below.

Theorem 4 (Randomized Reduction) *There is a randomized algorithm that solves a γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$ in $\tilde{O}(\mathcal{S}_{\text{tot}}^{\omega+1} + \mathcal{S}_{\text{tot}}^2 \mathcal{A}_{\text{tot}}) + \tilde{O}(\mathcal{S}_{\text{tot}}) \cdot \mathcal{T}_{\text{Rnd}}(\mathcal{S}_{\text{tot}}, \mathcal{A}_{\text{tot}}, \Theta(1-\gamma))$ expected time.*

Proof We prove that Algorithm 1 with uniformly random policy selection respects the desired properties. Note that its correctness and running time follow from Theorem 17 provided that the number of iterations T of Algorithm 1 satisfies $\mathbb{E}[T] = \tilde{O}(\mathcal{S}_{\text{tot}})$.

To this end, note that by Lemma 9 the set D contains $D(\pi)$ as long as $\Delta^{\mathcal{M}}_{*,\min}(\pi) \neq 0$. Thus, by Lemma 19, $\Phi(\mathcal{M})$ decreases by a factor of 2 each iteration in expectation, since clearly Φ is such that $\Phi(\mathcal{M} \setminus D) \leq \Phi(\mathcal{M} \setminus D(\pi))$. When $\Phi(\mathcal{M}) \leq 1$, $\mathcal{M}$ has only one policy which is guaranteed to be optimal, and thus we will get $\Delta^{\mathcal{M}}_{*,\min}(\pi) = 0$ and $D = \emptyset$ at that point. The value of $\Phi(\mathcal{M})$ is upper bounded by $\mathcal{A}_{\text{tot}}^{\mathcal{S}_{\text{tot}}}$, and thus the expected number of iterations T of the for loop until we have an optimal policy is $O(\mathcal{S}_{\text{tot}} \log \mathcal{A}_{\text{tot}})$. By Theorem 17, each iteration takes $O(\mathcal{S}_{\text{tot}}^{\omega} + \mathcal{S}_{\text{tot}} \mathcal{A}_{\text{tot}}) + \mathcal{T}_{\text{Rnd}}(\mathcal{S}_{\text{tot}}, \mathcal{A}_{\text{tot}}, \Theta(1-\gamma))$ randomized expected time, and thus the stated expected runtime of our algorithm follows. ■

5. Conclusion

In this paper we propose a generic framework that reduces solving a γ-DMDP *exactly* to solving a sequence of DMDPs *approximately*. Plugging existing approximate solvers into the framework results in state-of-the-art running times for exactly solving DMDPs. On top of the baseline deterministic framework, we show how by utilizing randomization we can further improve the algorithms.

Acknowledgements

Thank you to anonymous reviewers for their feedback. Thank you to Bento Natura and László Végh for helpful comments and discussions. We also thank LLMs for general writing advice. Andrei Graur was supported in part by the Nakagawa departmental fellowship award from the Management Science and Engineering Department at Stanford University, Microsoft Research Faculty Fellowship, NSF CAREER Award CCF-1844855, and NSF Grant CCF-1955039. Aaron Sidford was supported in part by a Microsoft Research Faculty Fellowship, NSF CAREER Award CCF-1844855, NSF Grant CCF-1955039, and a PayPal research award. Ta-Wei Tu was supported in part by a Microsoft Research Faculty Fellowship and NSF CAREER Award CCF-1844855.

References

Alekh Agarwal, Sham M. Kakade, and Lin F. Yang. Model-based reinforcement learning with a generative model is minimax optimal. In *Conference on Learning Theory, COLT 2020*, volume 125 of *Proceedings of Machine Learning Research*, pages 67–83. PMLR, 2020.

Xavier Allamigeon, Daniel Dadush, Georg Loho, Bento Natura, and László A. Végh. Interior point methods are not worse than simplex. *SIAM J. Comput.*, 54(5):S22–178, 2025.

Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, pages 522–539. SIAM, 2021.

Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025*, pages 2005–2039. SIAM, 2025.

David Avis and Oliver Friedmann. An exponential lower bound for Cunningham's rule. *Math. Program.*, 161(1-2):271–305, 2017.

Mohammad Gheshlaghi Azar, Rémi Munos, and Bert Kappen. On the sample complexity of reinforcement learning with a generative model. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, 2012.

Mohammad Gheshlaghi Azar, Rémi Munos, and Hilbert J. Kappen. Minimax PAC bounds on the sample complexity of reinforcement learning with a generative model. *Mach. Learn.*, 91(3): 325–349, 2013.

Richard Bellman. Dynamic programming. *science*, 153(3731):34–37, 1966.

Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 4. Athena scientific, 2012.

Dimitri P Bertsekas and John N Tsitsiklis. Neuro-dynamic programming: an overview. In *Proceedings of 1995 34th IEEE conference on decision and control*, volume 1, pages 560–564. IEEE, 1995.

Emma Brunskill, Bethany R. Leffler, Lihong Li, Michael L. Littman, and Nicholas Roy. Provably efficient learning with typed parametric models. *J. Mach. Learn. Res.*, 10:1955–1988, 2009.

Michael B. Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019*, pages 938–942. ACM, 2019.

Dan Dadush, László A Végh, and Giacomo Zambelli. Geometric rescaling algorithms for submodular function minimization. *Mathematics of Operations Research*, 46(3):1081–1108, 2021.

Daniel Dadush, Sophie Huiberts, Bento Natura, and László A. Végh. A scaling-invariant algorithm for linear programming whose running time depends only on the constraint matrix. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020*, pages 761–774. ACM, 2020a.

Daniel Dadush, Bento Natura, and László A. Végh. Revisiting Tardos's framework for linear programming: Faster exact solutions using approximate solvers. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 931–942. IEEE, 2020b.

Daniel Dadush, Zhuan Khye Koh, Bento Natura, Neil Olver, and László A. Végh. A strongly polynomial algorithm for linear programs with at most two nonzero entries per row or column. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024*, pages 1561–1572. ACM, 2024.

Daniel Dadush, Haoyuan Ma, Bento Natura, and László A. Végh. Trust region interior point methods: Optimal ℓ_2- and faster wide-neighborhood path following. In *Proceedings of the 58th Annual ACM Symposium on Theory of Computing, STOC 2026*, pages 755–766. ACM, 2026.

Thomas Degris, Olivier Sigaud, and Pierre-Henri Wuillemin. Learning the structure of factored markov decision processes in reinforcement learning problems. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006)*, volume 148 of *ACM International Conference Proceeding Series*, pages 257–264. ACM, 2006.

Yann Disser, Oliver Friedmann, and Alexander V. Hopp. An exponential lower bound for Zadeh's pivot rule. *Math. Program.*, 199(1):865–936, 2023.

Ran Duan, Hongxun Wu, and Renfei Zhou. Faster matrix multiplication via asymmetric hashing. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023*, pages 2129–2138. IEEE, 2023.

John Fearnley. Exponential lower bounds for policy iteration. In *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010*, Lecture Notes in Computer Science, pages 551–562. Springer, 2010.

Oliver Friedmann, Thomas Dueholm Hansen, and Uri Zwick. Subexponential lower bounds for randomized pivoting rules for the simplex algorithm. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011*, pages 283–292. ACM, 2011.

François Le Gall. Faster rectangular matrix multiplication by combination loss analysis. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024*, pages 3765–3791. SIAM, 2024.

Mehrdad Ghadiri, Richard Peng, and Santosh S. Vempala. The bit complexity of efficient continuous optimization. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023*, pages 2059–2070. IEEE, 2023.

Thomas Dueholm Hansen. Worst-case analysis of strategy iteration and the simplex method. 2012.

Thomas Dueholm Hansen, Peter Bro Miltersen, and Uri Zwick. Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. *J. ACM*, 60(1): 1:1–1:16, 2013.

Thomas Dueholm Hansen, Haim Kaplan, and Uri Zwick. Dantzig's pivoting rule for shortest paths, deterministic MDPs, and minimum cost to time ratio cycles. In *Proceedings of the Twenty-Fifth*

Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, pages 847–860. SIAM, 2014.

Ronald A Howard. Dynamic programming and markov processes. 1960.

Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. A faster algorithm for solving general LPs. In *Proceeding of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2021*, pages 823–832. ACM, 2021.

Yujia Jin and Aaron Sidford. Efficiently solving MDPs with stochastic mirror descent. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, volume 119 of *Proceedings of Machine Learning Research*, pages 4890–4900. PMLR, 2020.

Yujia Jin, Ishani Karmarkar, Aaron Sidford, and Jiayi Wang. Truncated variance reduced value iteration. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024*, 2024.

Yujia Jin, Ishani Karmarkar, Aaron Sidford, and Jiayi Wang. Reusing samples in variance reduction. *arXiv preprint arXiv:2509.02526*, 2025.

Sham Machandranath Kakade. *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom), 2003.

Adam Karczmarz. Improved strongly polynomial algorithms for deterministic MDPs, 2VPI feasibility, and discounted all-pairs shortest paths. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, pages 154–172. SIAM, 2022.

Jonathan A. Kelner and Daniel A. Spielman. A randomized polynomial-time simplex algorithm for linear programming. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, STOC 2006*, pages 51–60. ACM, 2006.

Tor Lattimore and Marcus Hutter. PAC bounds for discounted MDPs. In *Algorithmic Learning Theory - 23rd International Conference, ALT 2012*, volume 7568 of *Lecture Notes in Computer Science*, pages 320–334. Springer, 2012.

Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in $\tilde{O}(\sqrt{rank})$ iterations and faster algorithms for maximum flow. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 424–433. IEEE Computer Society, 2014.

Yin Tat Lee and Aaron Sidford. Efficient inverse maintenance and faster algorithms for linear programming. In *56th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2015*, pages 230–249. IEEE Computer Society, 2015.

Yin Tat Lee, Zhao Song, and Qiuyi Zhang. Solving empirical risk minimization in the current matrix multiplication time. In *Conference on Learning Theory, COLT 2019*, Proceedings of Machine Learning Research, pages 2140–2157. PMLR, 2019.

Michael L. Littman, Thomas L. Dean, and Leslie Pack Kaelbling. On the complexity of solving markov decision problems. In *UAI '95: Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*, pages 394–402. Morgan Kaufmann, 1995.

Omid Madani, Mikkel Thorup, and Uri Zwick. Discounted deterministic markov decision processes and discounted all-pairs shortest paths. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009*, pages 958–967. SIAM, 2009.

Nimrod Megiddo. Towards a genuinely polynomial algorithm for linear programming. *SIAM J. Comput.*, 12(2):347–353, 1983.

Mary Melekopoglou and Anne Condon. On the complexity of the policy improvement algorithm for markov decision processes. *INFORMS J. Comput.*, 6(2):188–192, 1994.

Bento Natura and László Végh. Personal communication. October, 2025.

Ian Post and Yinyu Ye. The simplex method is strongly polynomial for deterministic markov decision processes. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013*, pages 1465–1473. SIAM, 2013.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994. ISBN 978-0-47161977-2.

Aviv Rosenberg and Yishay Mansour. Oracle-efficient regret minimization in factored MDPs with unknown structure. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*, pages 11148–11159, 2021.

Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

Bruno Scherrer. Improved and generalized upper bounds on the complexity of policy iteration. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013.*, pages 386–394, 2013.

Aaron Sidford, Mengdi Wang, Xian Wu, Lin Yang, and Yinyu Ye. Near-optimal time and sample complexities for solving markov decision processes with a generative model. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pages 5192–5202, 2018.

Aaron Sidford, Mengdi Wang, Xian Wu, and Yinyu Ye. Variance reduced value iteration and faster algorithms for solving markov decision processes. *Naval Research Logistics (NRL)*, 70(5):423–442, 2023.

Olivier Sigaud and Olivier Buffet. *Markov decision processes in artificial intelligence*. John Wiley & Sons, 2013.

Satinder Singh, Tommi S. Jaakkola, Michael L. Littman, and Csaba Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Mach. Learn.*, 38(3):287–308, 2000.

Arne Storjohann. The shifted number system for fast linear algebra on integer matrices. *J. Complex.*, 21(4):609–650, 2005.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Mach. Learn.*, 3: 9–44, 1988.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Richard S. Sutton, David A. McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, pages 1057–1063. The MIT Press, 1999.

Éva Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Oper. Res.*, 34(2):250–256, 1986.

Paul Tseng. Solving h-horizon, stationary markov decision problems in time proportional to log (h). *Operations Research Letters*, 9(5):287–297, 1990.

John N. Tsitsiklis and Benjamin Van Roy. On average versus discounted reward temporal-difference learning. *Mach. Learn.*, 49(2-3):179–191, 2002.

Jan van den Brand. A deterministic linear program solver in current matrix multiplication time. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020*, pages 259–278. SIAM, 2020.

Jan van den Brand, Yin Tat Lee, Yang P. Liu, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Minimum cost flows, MDPs, and ℓ_1-regression in nearly linear time for dense instances. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2021*, pages 859–869. ACM, 2021.

Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024*, pages 3792–3835. SIAM, 2024.

Stephen A. Vavasis and Yinyu Ye. A primal-dual interior point method whose running time depends only on the constraint matrix. *Math. Program.*, 74:79–120, 1996.

Martin J. Wainwright. Variance-reduced Q-learning is minimax optimal. *arXiv preprint arXiv:1906.04697*, 2019.

Mengdi Wang. Randomized linear programming solves the discounted markov decision problem in nearly-linear running time. *arXiv preprint arXiv:1704.01869*, 2017.

Christopher J. C. H. Watkins and Peter Dayan. Technical note Q-learning. *Mach. Learn.*, 8:279–292, 1992.

Zheng Wei, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. Reinforcement learning to rank with markov decision process. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 945–948. ACM, 2017.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256, 1992.

Ziping Xu and Ambuj Tewari. Reinforcement learning in factored MDPs: Oracle-efficient algorithms and tighter regret bounds for the non-episodic setting. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

Yinyu Ye. A new complexity result on solving the markov decision problem. *Math. Oper. Res.*, 30 (3):733–749, 2005.

Yinyu Ye. The simplex and policy-iteration methods are strongly polynomial for the markov decision problem with a fixed discount rate. *Math. Oper. Res.*, 36(4):593–603, 2011.

Appendix A. A Randomized Policy Iteration Algorithm

In this section, we instantiate our framework (Algorithm 1) to obtain a simple randomized variant of policy iteration that alternates between a "random" step and $O(\log(1/\gamma)/(1-\gamma))$ policy iteration steps, outperforming the deterministic variant (Algorithm 2). In particular, we start from a policy selected uniformly at random in Algorithm 1, as in Section 4, and, to implement a δ_γ-approximate γ-DMDP solver ApxALG, apply the well-known policy iteration algorithm (Howard, 1960) that, for each state s, simultaneously switches the action taken at s to one that has the best advantage with respect to the current policy π. In other words, in each iteration of policy iteration, we update π to π^+ where $\Delta^{\mathcal{M}}(v_\pi)_{s,\pi^+(s)} = \max_a \Delta^{\mathcal{M}}(v_\pi)_{s,a}$ for all $s \in \mathcal{S}$.

Algorithm 2 A randomized policy iteration algorithm

Function `RandomizedPolicyIteration(`$\mathcal{M}$`):`
- **for** $t = 0, 1, \ldots$ **do**
  - Select a uniformly random policy π of $\mathcal{M}$.
  - **for** $x = 0, \ldots, \lceil \frac{\ln(3(1+\gamma)/(1-\gamma)^2)}{1-\gamma} \rceil$ **do**
    - $\pi \leftarrow$ `PolicyIterationStep(`$\mathcal{M}, \pi$`).`
  - **end**
  - $\mathcal{M} \leftarrow \mathcal{M} \setminus D$ where $D = \{(s,a) \in \mathcal{A} : \Delta^{\mathcal{M}}(v)_{s,a} < -\Delta^{\mathcal{M}}_{\max}(v_\pi) \cdot \frac{1-\gamma}{3}\}$.
  - **if** $D = \emptyset$ **then return** π `// For analysis define` $T \stackrel{\text{def}}{=} t$ `on this line`
- **end**

Function `PolicyIterationStep(`$\mathcal{M}, \pi$`):`
- **return** π^+, where $\pi^+(s) \stackrel{\text{def}}{=} \arg\max_{(s,a)\in\mathcal{A}} r_{s,a} + \gamma \cdot \langle p(s,a), v_\pi \rangle$.

To prove the correctness of Algorithm 2, we use the following fact that $\Theta(\log(1/\delta)/(1-\gamma))$ policy iteration steps times suffices to improve the optimality of a policy π by a $\delta < 1$ factor.

Lemma 20 (see, e.g., (Scherrer, 2013, Lemma 2)) *For any γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$, policy π of $\mathcal{M}$, and $\pi^+ \stackrel{\text{def}}{=}$ `PolicyIterationStep(`$\mathcal{M}, \pi$`)` (as in Algorithm 2), $\|v^{\mathcal{M}}_{\pi^+} - v^{\mathcal{M}}_*\|_\infty \le \gamma \|v^{\mathcal{M}}_\pi - v^{\mathcal{M}}_*\|_\infty$.*

Using Lemma 20, we prove the following corollary.

Corollary 21 *Given a γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, p)$, Algorithm 2 computes an optimal policy π of $\mathcal{M}$ in expected $\tilde{O}((\frac{\mathcal{S}_{\text{tot}}^{\omega+1} + \mathcal{S}_{\text{tot}}^2 \mathcal{A}_{\text{tot}}}{1-\gamma}) \log(\frac{1}{1-\gamma}))$ time.*

Proof We show that Algorithm 2 is an instantiation of Algorithm 1, where π in Algorithm 1 is chosen uniformly at random form the remaining policies (as in Section 4) and the for loop in Algorithm 2 is used to implement Algorithm 1 in Algorithm 1, i.e., compute ε-optimal values v for $\varepsilon = \Delta^{\mathcal{M}}_{\max}(v_\pi) \cdot \frac{1-\gamma}{3(1+\gamma)}$. For this, note that Lemma 20 implies that after $\lceil \frac{\ln(3(1+\gamma)/(1-\gamma)^2)}{1-\gamma} \rceil$ iterations of policy iteration—i.e., $\lceil \frac{\ln(3(1+\gamma)/(1-\gamma)^2)}{1-\gamma} \rceil$ iterations of the for loop in Algorithm 2—starting from π, the new policy π' satisfies that $\|v^{\mathcal{M}}_{\pi'} - v^{\mathcal{M}}_{*}\|_\infty \le \frac{(1-\gamma)^2}{3(1+\gamma)} \|v^{\mathcal{M}}_{\pi} - v^{\mathcal{M}}_{*}\|_\infty$. We now show that $(1 - \gamma)\|v^{\mathcal{M}}_{\pi} - v^{\mathcal{M}}_{*}\|_\infty \le \Delta^{\mathcal{M}}_{\max}(v_\pi)$. For this, note that it suffices to show $(1-\gamma)\|v^{\mathcal{M}^\pi}_{*}\|_\infty \le \Delta^{\mathcal{M}}_{\max}(v_\pi)$. This follows immediately from the fact that $\Delta^{\mathcal{M}}_{\max}(v_\pi)$ is the maximum reward in the γ-DMDP $\mathcal{M}^\pi$. Therefore, the loop in Algorithm 2 computes ε-optimal values v for $\varepsilon = \Delta^{\mathcal{M}}_{\max}(v_\pi) \cdot \frac{1-\gamma}{3(1+\gamma)}$. Thus, the correctness of the algorithm follows from Theorem 17.

To bound the running time, note that computing v_π can be implemented in $O(\mathcal{S}^{\omega}_{\text{tot}})$ time (Fact 8). Additionally, $\Delta^{\mathcal{M}}_{\max}(v_\pi)$ and each iteration of the for loop in Algorithm 2 can be implemented in $O(\mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}})$. Moreover, by Lemma 19, the number of iterations T is at most $\tilde{O}(\mathcal{S}_{\text{tot}})$ in expectation, which implies the desired running time. ■

This result opens the door to analyzing a range of randomized variants of the policy iteration algorithm. We chose to analyze this particular algorithm in Algorithm 2 as it is consistent with the framework discussed at the beginning of Section 3. We leave the exploration of alternative randomized variants of the policy iteration algorithm for future work.

Appendix B. Omitted Proofs

In this section, we provide the proofs that are omitted from the main body of the paper. We start with the result regarding converting an approximate γ-DMDP algorithm that succeeds with constant probability into one with an expected running time, which we restate below.

Lemma 5 *Suppose there is a randomized algorithm that on input γ-DMDP $\mathcal{M}' = (\mathcal{S}', \mathcal{A}', p', r')$ outputs $\delta'\|r'\|_\infty$-optimal values v of $\mathcal{M}$ in time $\mathcal{T}_{\text{MC}}(\mathcal{S}'_{\text{tot}}, \mathcal{A}'_{\text{tot}}, \delta')$ with constant probability. Then, there is a randomized δ-approximate γ-DMDP algorithm that on input γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$ runs in expected time $\mathcal{T}_{\text{Rnd}}(\mathcal{S}_{\text{tot}}, \mathcal{A}_{\text{tot}}, \delta) = O(\mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}} + \mathcal{T}_{\text{MC}}(\mathcal{S}_{\text{tot}}, \mathcal{A}_{\text{tot}}, \Theta(\delta \cdot (1-\gamma)))$.*

Proof Consider running the Monte Carlo algorithm on $\mathcal{M}$ with accuracy $\delta' \stackrel{\text{def}}{=} \delta \cdot \frac{1-\gamma}{1+\gamma}$. Let v be the values it returns. In $O(\mathcal{S}_{\text{tot}}\mathcal{A}_{\text{tot}})$ time we can compute $\mathcal{T}_*(v)$ and $\|v - \mathcal{T}_*(v)\|_\infty$. If $\|v - \mathcal{T}_*(v)\|_\infty \le \delta\|r\|_\infty \cdot (1-\gamma)$, then we have by Lemma 7 that $\|v - v_*\|_\infty \le \delta\|r\|_\infty$ and thus we can output v. On the other hand, again by Lemma 7 we know that $\|\mathcal{T}_*(v) - v\|_\infty \le \|\mathcal{T}_*(v) - v_*\|_\infty + \|v_* - v\|_\infty \le (1+\gamma)\|v - v_*\|_\infty$ and thus if v is $\delta'\|r\|_\infty$-optimal then we must have that $\|v - \mathcal{T}_*(v)\|_\infty \le \delta\|r\|_\infty \cdot (1-\gamma)$. Thus, with constant probability this procedure outputs values v with certified optimality. Repeating this procedure until such a v is output yields the result. ■

We now prove the following two lemmas (stated in Section 3.1) which are used to prove Lemma 9.

Lemma 10 *For a γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$ and any $u, v \in \mathbb{R}^{\mathcal{S}}$ and $(s, a) \in \mathcal{A}$,*

$$\left|\Delta^{\mathcal{M}}(u)_{s,a} - \Delta^{\mathcal{M}}(v)_{s,a}\right| \le (1+\gamma)\|u - v\|_\infty.$$

Proof It follows from the definition of $\Delta^{\mathcal{M}}(u)_{s,a}$ that

$$\begin{aligned}|\Delta(u)_{s,a} - \Delta(v)_{s,a}| &= |(r_{s,a} + \gamma \cdot \langle p(s,a), u\rangle - u_s) - (r_{s,a} + \gamma \cdot \langle p(s,a), v\rangle - v_s)| \\ &= |\gamma \cdot \langle p(s,a), u - v\rangle - u_s + v_s| \le \gamma \cdot |\langle p(s,a), u - v\rangle| + |u_s - v_s| \\ &\le (1+\gamma)\|u - v\|_\infty,\end{aligned}$$

where in the last inequality we used that $\langle p(s,a), u - v\rangle \le \|p(s,a)\|_1 \cdot \|u - v\|_\infty = \|u - v\|_\infty$ and that $|u_s - v_s| \le \|u - v\|_\infty$. ■

Lemma 11 *For any policy π in γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$, $\Delta^{\mathcal{M}}_{*,\min}(\pi) \le -(1-\gamma)\|v^{\mathcal{M}}_\pi - v^{\mathcal{M}}_*\|_\infty$.*

Proof Note that $\Delta(v_*)_{s,\pi(s)} = r_{s,\pi(s)} + \gamma \cdot \langle p(s,\pi(s)), v_*\rangle - [v_*]_s = [\mathcal{T}_\pi(v_*)]_s - [v_*]_s$. Consequently, using (2), which, as discussed in Section 2, follows from Lemma 7, we obtain

$$\|v_\pi - v_*\|_\infty \le \frac{1}{1-\gamma}\|v_* - \mathcal{T}_\pi(v_*)\|_\infty = \frac{1}{1-\gamma}\max_{s\in S}|\Delta(v_*)_{s,\pi(s)}|\,.$$

Rearranging and using that $\max_{s\in S}|\Delta(v_*)_{s,\pi(s)}| = -\Delta_{*,\min}(\pi)$ proves the lemma. ■

The following two claims, stated in Section 3.2, were used to prove Lemma 12, our result regarding the correctness and efficiency guarantee of `MultValApprox`.

Claim 15 *For every policy π of γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$, $v^{\mathcal{M}^\pi}_\pi = \mathbf{0}$ and $\|v^{\mathcal{M}^\pi}_*\|_\infty \ge \Delta^{\mathcal{M}}_{\max}(v_\pi)$.*

Proof That $\|v^{\mathcal{M}^\pi}_\pi\|_\infty = 0$ follows by Lemma 14 since each $(s, \pi(s))$ satisfies $\Delta^{\mathcal{M}}(v_\pi)_{s,\pi(s)} = 0$. For the bound on $\|v^{\mathcal{M}^\pi}_*\|_\infty$, consider the action $(s_*, a_*) \in \mathcal{A}$ that maximizes $\Delta^{\mathcal{M}}(v_\pi)_{s_*,a_*}$ which by definition has $\Delta^{\mathcal{M}}(v_\pi)_{s_*,a_*} = \Delta^{\mathcal{M}}_{\max}(v_\pi)$. Let π' be the policy that is the same as π except at state s_* where $\pi'(s_*) = a_*$. Note that since for every s, the reward of $(s, \pi(s))$ is 0 in $\mathcal{M}^\pi$, it follows easily that $[v^{\mathcal{M}^\pi}_{\pi'}]_{s_*} \ge \Delta^{\mathcal{M}}_{\max}(v_\pi)$, as the rewards from following π' are all non-negative. ■

Claim 16 *For any policy π of γ-DMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$, no optimal policy π_* of $\mathcal{M}$ has $\pi_*(s) = a$ for $\Delta^{\mathcal{M}}(v_\pi)_{s,a} < -\Delta^{\mathcal{M}}_{\max}(v_\pi) \cdot (1+\gamma)$.*

Proof By Claim 15 and Lemma 14, we have $\|v_\pi - v_*\| \ge \Delta_{\max}(v_\pi)$. Thus, Lemma 9 asserts that (s,a) with $\Delta(v_\pi)_{s,a} < -\Delta^{\mathcal{M}}_{\max}(v_\pi) \cdot (1+\gamma)$ is not part of any optimal policy. ■

Finally, we prove the result, stated in Section 4, that we use to prove our randomized reduction.

Lemma 19 *Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$ be a γ-DMDP where $\Phi(\mathcal{M}) > 0$. Then, letting $D^{\mathcal{M}}(\pi) \stackrel{\text{def}}{=} \{(s,a) : \Delta^{\mathcal{M}}(v^{\mathcal{M}}_*)_{s,a} \le \Delta^{\mathcal{M}}_{*,\min}(\pi)\}$ for policy π we have $\mathbb{E}_{\pi\sim\Pi(\mathcal{M})}[\Phi(\mathcal{M} \setminus D^{\mathcal{M}}(\pi))] \le \frac{1}{2}\Phi(\mathcal{M})$.*

Proof We proceed by an induction on $\Phi(\mathcal{M})$. For the base case where $\Phi(\mathcal{M}) = 1$, there exists exactly one policy π of $\mathcal{M}$. This means that all actions are optimal, i.e., have advantage zero, and thus $D^{\mathcal{M}}(\pi)$ contains all of them and thus $\Phi(\mathcal{M} \setminus D^{\mathcal{M}}(\pi)) = 0 \le \frac{1}{2}$.

When $\Phi(\mathcal{M}) > 1$, consider an action (s_*, a_*) that minimizes $\Delta^{\mathcal{M}}(v_*^{\mathcal{M}})_{s_*,a_*}$. If $\Delta^{\mathcal{M}}(v_*^{\mathcal{M}})_{s_*,a_*} = 0$, then again all actions are optimal and therefore $\Phi(\mathcal{M} \setminus D^{\mathcal{M}}(\pi)) = 0 \leq \frac{1}{2}\Phi(\mathcal{M})$. We thus assume for the remainder of the proof that $\Delta^{\mathcal{M}}(v_*^{\mathcal{M}})_{s_*,a_*} < 0$. Let $\mathcal{M}' \stackrel{\text{def}}{=} \mathcal{M} \setminus \{(s_*, a_*)\}$ be the γ-DMDP obtained by removing (s_*, a_*) from $\mathcal{M}$. First observe that $(s_*, a_*) \in D^{\mathcal{M}}(\pi)$ regardless of the choice of π and that $\Phi(\mathcal{M}') = \Phi(\mathcal{M}) \cdot \frac{|\mathcal{A}_{s_*}|-1}{|\mathcal{A}_{s_*}|}$ by definition of Φ. Consider the process of selecting a uniformly random policy π of $\mathcal{M}$. We consider the two cases of whether $\pi(s_*) = a_*$ or not and bound $\mathbb{E}[\Phi(\mathcal{M} \setminus D^{\mathcal{M}}(\pi))]$ by

$$\begin{aligned}\mathbb{E}[\Phi(\mathcal{M} \setminus D^{\mathcal{M}}(\pi))] &= \Pr[\pi(s_*) \neq a_*] \cdot \mathbb{E}[\Phi(\mathcal{M} \setminus D^{\mathcal{M}}(\pi)) \mid \pi(s_*) \neq a_*] \\ &\quad + \Pr[\pi(s_*) = a_*] \cdot \mathbb{E}[\Phi(\mathcal{M} \setminus D^{\mathcal{M}}(\pi)) \mid \pi(s_*) = a_*].\end{aligned} \tag{5}$$

To compute the first term, note that the event $\pi(s_*) \neq a_*$ happens with probability $1 - \frac{1}{|\mathcal{A}_{s_*}|}$. The distribution of $\pi \sim \Pi(\mathcal{M})$ conditioned on $\pi(s_*) \neq a_*$ is the same as the distribution $\pi \sim \Pi(\mathcal{M}')$. Moreover, observe that we have $v_*^{\mathcal{M}} = v_*^{\mathcal{M}'}$ since we only remove a suboptimal action, and therefore $\Delta^{\mathcal{M}}(v_*^{\mathcal{M}})_{s,a} = \Delta^{\mathcal{M}'}(v_*^{\mathcal{M}'})_{s,a}$ for all $(s,a) \neq (s_*, a_*)$. In particular this implies $D^{\mathcal{M}'}(\pi) = D^{\mathcal{M}}(\pi) \setminus \{(s_*, a_*)\}$. Because $(s_*, a_*) \in D^{\mathcal{M}}(\pi)$ regardless of the choice of π, we have

$$\mathbb{E}_{\pi\sim\Pi(\mathcal{M})}[\Phi(\mathcal{M} \setminus D^{\mathcal{M}}(\pi)) \mid \pi(s_*) \neq a_*] = \mathbb{E}_{\pi\sim\Pi(\mathcal{M}')}[\Phi(\mathcal{M}' \setminus D^{\mathcal{M}'}(\pi))] \leq \frac{1}{2}\Phi(\mathcal{M}'),$$

where the last inequality follows from the inductive hypothesis. For the second term in (5), again since $D^{\mathcal{M}}(\pi)$ contains (s_*, a_*) we have $\Phi(\mathcal{M} \setminus D^{\mathcal{M}}(\pi)) \leq \Phi(\mathcal{M}')$. Overall, we get

$$\begin{aligned}\mathbb{E}_{\pi\sim\Pi(\mathcal{M})}[\Phi(\mathcal{M} \setminus D^{\mathcal{M}}(\pi))] &\leq \frac{|\mathcal{A}_{s_*}|-1}{|\mathcal{A}_{s_*}|} \cdot \frac{1}{2}\Phi(\mathcal{M}') + \frac{1}{|\mathcal{A}_{s_*}|} \cdot \Phi(\mathcal{M}') = \frac{|\mathcal{A}_{s_*}|^2-1}{2|\mathcal{A}_{s_*}|^2} \cdot \Phi(\mathcal{M}) \\ &\leq \frac{1}{2}\Phi(\mathcal{M}).\end{aligned}$$

■