Lower Bound on Howard Policy Iteration for Deterministic Markov Decision Processes

Ali Asadi[1] **Krishnendu Chatterjee**[1] **Jakob de Raaij**[2]

[1]Institute of Science and Technology Austria (ISTA)
[2]Harvard University

Abstract

Deterministic Markov Decision Processes (DMDPs) are a mathematical framework for decision-making where the outcomes and future possible actions are deterministically determined by the current action taken. DMDPs can be viewed as a finite directed weighted graph, where in each step, the controller chooses an outgoing edge. An objective is a measurable function on runs (or infinite trajectories) of the DMDP, and the value for an objective is the maximal cumulative reward (or weight) that the controller can guarantee. We consider the classical mean-payoff (aka limit-average) objective, which is a basic and fundamental objective.
Howard's policy iteration algorithm is a popular method for solving DMDPs with mean-payoff objectives. Although Howard's algorithm performs well in practice, as experimental studies suggested, the best known upper bound is exponential and the current known lower bound is as follows: For the input size I, the algorithm requires $\widetilde{\Omega}(\sqrt{I})$ iterations, where $\widetilde{\Omega}$ hides the poly-logarithmic factors, i.e., the current lower bound on iterations is sub-linear with respect to the input size. Our main result is an improved lower bound for this fundamental algorithm where we show that for the input size I, the algorithm requires $\widetilde{\Omega}(I)$ iterations.

1 INTRODUCTION

Deterministic Markov Decision Processes. Deterministic Markov Decision Processes (DMDPs) [Puterman, 1994] are a mathematical framework for sequential decision-making where an agent interacts with a fully deterministic environment. They are modeled as a graph, where in every step the controller chooses a successor vertex from the neighbors of the current vertex. This repeated process generates an infinite sequence of vertices (called a run). Policies for the controller provide the successor vertex choice at every vertex. A payoff function assigns a real value to every run. We consider a classical and well-studied function: the mean-payoff (or limit-average) payoff function [Puterman, 1994, Filar and Vrieze, 2012]. Every edge of the graph is assigned an integer weight, and the payoff of a run is the long-run average of the weights of the run.

Applications. This formalism is particularly relevant in settings where system behaviors are fully known, such as controlled robotic environments or algorithmic planning tasks [Blondel and Tsitsiklis, 2000]. DMDPs also appear in formal verification and synthesis, where deterministic transitions allow for tractable analysis of safety, liveness, and temporal logic specifications [Baier and Katoen, 2008]. For example, in autonomous systems, DMDPs are a basic model for synthesizing controllers that guarantee correct behaviors [Alur, 2015]. Besides the practical applications, DMDPs can demonstrate fundamental computational limits, e.g., NP-hardness of optimal planning [Littman, 1997]. Furthermore, this model corresponds to classical directed weighted graphs, which have many applications such as network routing, travel planning, etc [Cormen et al., 2022].

Motivation. One of the main algorithm in the area of planning and sequential decision-making is Howard's policy iteration [Howard, 1960]. Dasdan [2004] compared various algorithms and showed that Howard's algorithm works well in practice as compared to other algorithms. Furthermore, lower and upper bounds for policy iteration algorithms have deep theoretical impact, e.g., lower bounds for policy iteration lead to lower bounds for pivoting approaches in linear programming [Friedmann et al., 2011]. Hence better theoretical understanding of Howard's algorithm is an interesting problem. Our work aims at the theoretical understanding of this fundamental algorithm for DMDPs with mean-payoff objectives. We first recall the previous results from the literature.

Previous Work on Howard's Policy Iteration. Howard's policy iteration has been extensively studied for general MDPs (not necessarily DMDPs) with discounted-sum and mean-payoff objectives. For MDPs with mean-payoff objectives, the best-known upper bound is exponential [Puterman, 1994]. Fearnley [2010], inspired by the work of Friedmann [2009], showed that Howard's algorithm requires exponential time for a specific family of MDPs. For discounted-sum objectives, better upper bounds are known in special cases of the discount factor: Post and Ye [2015] established a strongly polynomial running time when the discount factor is constant or represented in unary, and Hansen et al. [2013] improved Post's bounds and extended these results to 2-player settings. However, the exponential lower bound from Fearnley [2010] still holds for arbitrary discount factors. Related problems, such as MDPs with total-reward objectives (analogous to the stochastic shortest path), also exhibit exponential lower bounds for strategy iteration [Fearnley, 2010]. While exponential lower bounds are established for stochastic models [Fearnley, 2010], game-theoretic models [Friedmann, 2009], and linear programming pivoting rules [Friedmann et al., 2011], identifying lower bounds specifically for Howard's policy iteration in simpler deterministic graph models has remained a compelling open question. Our contribution addresses this by presenting improved lower bounds for this simplest model, complementing the more general cases already explored in the literature.

Recent Results on Howard's Policy Iteration. There are several recent advancements on Howard's Policy Iteration in the literature, which highlight ongoing research in this direction. Loff and Skomra [2024] demonstrated polynomial-time smoothed complexity for deterministic models (DMDPs and turn-based games with mean-payoff and discounted-sum objectives). However, Christ and Yannakakis [2023] presented a sub-exponential lower bound for the smoothed complexity of Howard's algorithm for stochastic MDPs with mean-payoff objectives. Moreover, Asadi et al. [2024] recently improved complexity results for two-player turn-based discounted games with unary weights through a new analysis for Howard's policy iteration.

Previous Work on DMDPs. DMDPs have been studied extensively in the literature [Arora et al., 2012, Boone and Gaujal, 2023, Castro, 2020, Madani, 2002, Madani et al., 2009]. Karp [1978] presented an algorithm for solving DMDPs with mean-payoff objectives in $O(mn)$ time (where m is the number of edges and n is the number of vertices), while Young et al. [1991] proposed an $O(mn + n^2 \log n)$-time algorithm that often performs better in practice despite its slightly worse time complexity. Although Howard's policy iteration works well in practice [Dasdan, 2004], the known theoretical upper and lower bounds for DMDPs with mean-payoff objectives are as follows: The best-known upper bound is exponential [Puterman, 1994]. Moreover, a better parametric bound of $O(n^3W)$ on the number of iterations can be obtained, since by Howard [1960], the number of policy iteration steps is at most the number of value iteration steps, which by Zwick and Paterson [1996] is bounded by $O(n^3W)$, where W is the maximum absolute weight. Hansen and Zwick [2010] presented a lower bound, giving DMDPs with $2n$ vertices, m edges, and edge weights of $O(n^{n^2})$, on which the algorithm requires $m-n+1$ iterations to find an optimal policy. For example, (a) with $m = O(n)$, this result shows that on input size of $\widetilde{O}(n^3)$, the algorithm requires $\Omega(n)$ iterations, or (b) with $m = O(n^2)$, this shows that on input size of $\widetilde{O}(n^4)$, the algorithm requires $\Omega(n^2)$ iterations. In particular, given the input description of an DMDP with I bits, the results shows that the lower bound on iterations is $\widetilde{\Omega}(\sqrt{I})$. In computer science establishing and improving lower bounds are challenging, and whether this lower bound can be improved is a fundamental problem which we address in this work.

Our Contributions. The above motivates the study of Howard's policy iteration for DMDPs with mean-payoff objective. In this work, we construct a family of DMDPs with $2n$ vertices, $O(n^2)$ edges, and edge weights of $O(n^2)$, on which Howard's algorithm requires $\Omega(n^2)$ iterations to find an optimal policy. Hence, the improved lower bound is as follows. Given the input description of an DMDP with I bits, the required number of iterations is $\widetilde{\Omega}(I)$. Table 1 summarizes the results.

Significance. As compared to the work of Hansen and Zwick [2010], the significance of our result is twofold. First, with respect to the input size of I, we improve the lower bound from $\widetilde{\Omega}(\sqrt{I})$ to $\widetilde{\Omega}(I)$. Second, there is an important implication with respect to the $O(n^3W)$ parametric bound by Zwick and Paterson [1996]. For the family of examples considered in Hansen and Zwick [2010], the best known upper bound is exponential as the weights are exponential. Thus, the examples of Hansen and Zwick [2010] belong to a class where the upper bound is exponential and a sub-linear lower bound is presented. In contrast, since the weights are polynomial for our class of examples, the upper bound on the number of iterations is polynomial (namely, quadratic with respect to the input size) and we present an almost-linear lower bound.

Technical Novelty. In the examples of Hansen and Zwick [2010], Howard's policy iteration goes through $\Theta(n^2)$ "good" cycles and due to the structure of the graph only ever finds a slightly better cycle. Each cycle weight is determined by a "key" edge with largest weight. To achieve the lower bound for policy iteration, the algorithm requires the next key edge weight to differ from the previous one by a multiplicative factor of n (or the cycle length, which can be up to n), leading to exponential edge weights. In contrast, our examples avoid this, inspired by the results of Fried-

Table 1: Comparison of lower bounds for Howard's policy iteration. $|V|$, $|E|$, and W correspond to the number of vertices, number of edges, and maximum absolute weight, respectively.

	$\lvert V\rvert$	$\lvert E\rvert$	W	Size	# Iterations
Hansen and Zwick [2010]	$2n$	m	$O(n^{n^2})$	$O(mn^2 \log n)$	$m - n + 1$
Ours	$2n$	$O(n^2)$	$O(n^2)$	$O(n^2 \log n)$	$\Omega(n^2)$

mann [2009], Fearnley [2010], using a similar concept to their "deceleration lane". Our DMDPs only have n "good" cycles that do not overlap edgewise. Thus, each cycle only needs to have a weight of 1 more than the previous one. The "deceleration lane" technique forces the algorithm to perform $\Omega(i)$ iterations to find the ith cycle after having found the $i-1$th cycle. This structure ensures that (a) Howard's algorithm finds the cycles in the right order and (b) performs $\Omega(n^2)$ iterations to find an optimal policy.

2 PRELIMINARIES

We present standard notations and definitions related to deterministic markov decision processes.

Deterministic Markov Decision Processes. A deterministic markov decision process (DMDP) is a finite directed weighted graph $P = (V, E, w)$ consisting of

- the set of vertices V, of size n;
- the set of edges $E \subseteq V \times V$, of size m, such that for all $v \in V$, the set $E(v) := \{u \mid (v, u) \in E\}$ is non-empty; and
- the weight function $w : E \to \mathbb{Z}$ that assigns a weight $w(v, u)$ for all edges $(v, u) \in E$.

We denote the largest absolute weight by $W := \max\{|w(v, u)| \mid (v, u) \in E\}$. The size of P is defined as $|P| := n + m + \sum_{(v,u)\in E} \lceil \log_2 |w(v, u)| \rceil$. The vertices are indexed and have an ordering.

Steps and Runs. Given an initial vertex $v_0 \in V$, the process proceeds as follows. In each step, the controller chooses the next vertex from the set $E(v)$. A *run* is an infinite sequence of vertices $\omega = \langle v_0, v_1, \ldots\rangle$ where for every step $t \geq 0$, the vertex $v_{t+1} \in E(v_t)$. We denote by Ω the set of all runs, and by Σ_v the set of all runs $\omega = \langle v_0, v_1, \ldots\rangle$ where $v_0 = v$.

Mean-payoff Objectives. An objective is a measurable function that assigns a real number to all runs. For a run $\omega = \langle v_0, v_1, \ldots\rangle$, the average for t steps is $\mathrm{Avg}_t(\omega) := \frac{1}{t}\sum_{i=0}^{t-1} w(v_i, v_{i+1})$. The $\liminf$ average is $\mathrm{LimAvg}(\omega) := \liminf_{t\to\infty} \mathrm{Avg}_t(\omega)$. The objective of controller is to maximize the $\liminf$ average of the run.

Positional Policies. Policies are recipes that specify how to choose the next vertex. A *positional* policy $\sigma \colon V \to V$ for the controller is a policy which chooses a vertex $\sigma(v) \in E(v)$ whenever the run visits vertex v. We denote by Σ^P the set of all positional policies. In general, policies can depend on past history and not only the current vertex. However, for mean-payoff objectives, positional policies are as powerful as general policies [Puterman, 1994]. Hence, in the sequel, every policy is positional.

Runs Given Policies in DMDPs. We define P^σ as the restricted DMDP where the controller follows the policy σ. Note that once the controller has fixed their policy, we obtain a graph where each vertex has exactly one outgoing edge. Given an initial vertex v, we obtain a run $P_v^\sigma = \langle v_0, v_1, \ldots\rangle$ such that $v_0 = v$, and for any step $t \geq 0$, $v_{t+1} = \sigma(v_t)$. The obtained run P_v^σ is a *lasso-shaped* run that consists in a finite cycle-free path $\mathcal{P} := \langle v_0, \ldots, v_p\rangle$ followed by a simple cycle $\mathcal{C} := \langle v_p, \ldots, v_{p+c-1}\rangle$ repeated forever, where v_p is the *head* of the cycle (the vertex with the least index in the cycle). The mean-payoff of the policy σ is defined as

$$\mathrm{val}^\sigma(v) := \mathrm{LimAvg}(P_v^\sigma) = \frac{1}{c}\sum_{i=0}^{c-1} w(v_{p+i}, v_{p+i+1})\,.$$

We define the potential function as

$$\mathrm{pot}^\sigma(v) := \sum_{i=0}^{p-1} \left(w(v_i, v_{i+1}) - \mathrm{val}^\sigma(v)\right)\,.$$

In words, the payoff $\mathrm{val}^\sigma(v)$ is the mean-payoff the controller obtains, in case they follow the policy σ, and the potential $\mathrm{pot}^\sigma(v)$ is the relative distance from v to v_p, where the weight of each edge is subtracted by the mean-payoff.

Value and Optimal Policies. The mean-payoff value for a vertex v is defined as $\mathrm{val}(v) := \max_{\sigma\in\Sigma^P} \mathrm{val}^\sigma(v)$. A policy σ for the controller is *optimal* for mean-payoff objectives if, for all vertices $v \in V$, we have $\mathrm{val}^\sigma(v) = \mathrm{val}(v)$.

Bellman Operator. Given a policy σ, we define the *appraisal* of an edge (v, u) as a tuple

$$\mathcal{A}^\sigma(v, u) := (\mathrm{val}^\sigma(u), w(v, u) - \mathrm{val}^\sigma(u) + \mathrm{pot}^\sigma(u))\,.$$

The Bellman operator, which is an operator from Σ^P to Σ^P, is defined as

$$\mathcal{B}(\sigma)(v) := \arg\max_{u\in E(v)} \mathcal{A}^\sigma(v, u)\,.$$

The appraisals are compared lexicographically, and ties are resolved by first favoring $u = \sigma(v)$, then vertices with the least index. For increased legibility, we will refer to the second term of the appraisal as

$$\mathcal{A}_2^\sigma(v,u) \coloneqq w(v,u) - \mathrm{val}^\sigma(u) + \mathrm{pot}^\sigma(u)\,.$$

Howard's Policy Iteration. Howard's policy iteration is a classical algorithm for computing the optimal policies in DMDPs with mean-payoff objectives. The algorithm starts with an arbitrary policy σ_0. In each iteration, the algorithm locally improves the current policy: Starting with σ_k at iteration k, the algorithm computes the payoff and the potential of the policy σ_k. Using these, it updates the policy using the Bellman operator defined above by setting $\sigma_{k+1} = \mathcal{B}(\sigma_k)$. The algorithm terminates if $\sigma_{k+1} = \sigma_k$, meaning no update to the policy was made. The correctness of Howard's algorithm is shown in Derman [1970], Puterman [1994].

3 OVERVIEW OF RESULTS

A natural question on Howard's policy iteration is that how many iteration it takes to find an optimal policy. In the following, we state a long-standing conjecture on the upper bound for Howard's policy iteration.

Conjecture 3.1 ([Hansen, 2012, Conjecture 6.1.1])**.** *The number of iterations performed by Howard's algorithm, when applied to a DMDP, is at most the number of edges.*

Hansen and Zwick [2010] constructed a family of DMDPs with n vertices and m edges on which Howard's algorithm performs $m-n+1$ iterations. However, the size of DMDPs is $O(mn^2 \log n)$ due to exponential weights. In this work, we present a family of DMDPs with $2n$ vertices and $O(n^2)$ edges on which Howard's algorithm performs $\Omega(n^2)$ iterations to find an optimal policy. The weights are bounded by $O(n^2)$. Hence, the size of our DMDPs is $O(n^2 \log n)$, which improves the dependency on the number of edges from linear to constant. Our main result is stated as follows.

Theorem 3.2 (Main Result)**.** *Let n be a positive integer. There exists a DMDP with $2n$ vertices, $\frac{3n^2+n}{2}$ edges, and size of $O(n^2 \log n)$ on which Howard's algorithm performs $\frac{n^2+7n-6}{2}$ iterations to find an optimal policy.*

4 IMPROVED LOWER BOUND

In this section, we construct a family of DMDPs with $2n$ vertices of size $O(n^2 \log n)$ on which Howard's algorithm performs $\Omega(n^2)$ iterations.

4.1 DMDP CONSTRUCTION

Given a positive integer n, we construct a DMDP $P_n = (V_n, E_n, w_n)$. We denote the set of vertices by

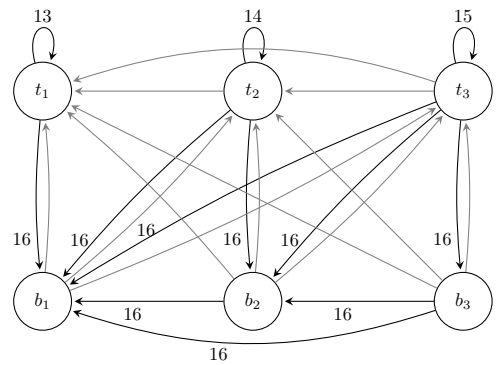


Figure 1: Our running example with $n = 3$. Unlabeled (gray) edges have weight 0.

$V_n = \{b_1, \ldots, b_n, t_1, \ldots, t_n\}$. The ordering of vertices is $(t_1, b_1, \ldots, b_n, t_2, \ldots, t_n)$. We denote the set of edges by

$$\begin{aligned} E_n \coloneqq &\{(b_i, b_j) \mid 1 \le j < i \le n\} \\ &\cup \{(b_i, t_j) \mid 1 \le i, j \le n\} \\ &\cup \{(t_i, b_j) \mid 1 \le j \le i \le n\} \\ &\cup \{(t_i, t_j) \mid 0 \le j \le i \le n\}\,. \end{aligned}$$

We now define the weight function as

$$w_n(v,u) \coloneqq \begin{cases} (n+1)^2 & v = b_i \wedge u = b_j \\ & \text{for all } 1 \le j < i \le n \\ (n+1)^2 & v = t_i \wedge u = b_j \\ & \text{for all } 1 \le j \le i \le n \\ 0 & v = b_i \wedge u = t_j \\ & \text{for all } 1 \le i, j \le n \\ 0 & v = t_i \wedge u = t_j \\ & \text{for all } 1 \le j < i \le n \\ n(n+1) + i & v = t_i \wedge u = t_i \\ & \text{for all } 1 \le i \le n \end{cases}$$

Figure 1 illustrates an example of the DMDP with $n = 3$.

4.2 POLICIES

We describe three families of policies that appear in Howard's algorithm. We first define the family of policies π_i for $1 \le i \le n$ as

$$\pi_i(v) \coloneqq \begin{cases} b_k & v = t_k & \text{for all } 1 \le k \le i-2 \\ t_k & v = t_k & \text{for all } k \in \{i-1, i\} \\ t_i & \text{otherwise} \end{cases}$$

We now define another family of policies $\sigma_{i,j}$ for $1 \leq i \leq n$ and $1 \leq j \leq i+1$ as

$$\sigma_{i,j}(v) \coloneqq \begin{cases} t_i & v = t_i \\ b_k & v = t_k \quad \text{for all } (1 \leq k \leq j \wedge k \neq i) \\ & \quad\quad\quad\quad \text{or } (k \leq i-2 \wedge j = 1) \\ t_i & v = b_1 \\ b_{k-1} & v = b_k \quad \text{for all } 2 \leq k \leq j \\ b_j & \text{otherwise} \end{cases}$$

Finally, we define the last family of policies τ_i for $1 \leq i \leq n$ as

$$\tau_i(v) \coloneqq \begin{cases} t_k & v = t_k \quad \text{for all } k \in \{i, i+1\} \\ b_k & v = t_k \quad \text{for all } 1 \leq k < i \\ t_i & v = b_1 \\ b_{k-1} & v = b_k \quad \text{for all } 2 \leq k \leq i+2 \\ b_{i+2} & \text{otherwise} \end{cases}$$

For more intuition, the policies π_2, $\sigma_{2,1}$, $\sigma_{2,3}$, and τ_2 over our running example are illustrated in Figure 2. An illustration of all policies for the running example can be found in Appendix A. Moreover, the outline of policies that appear in the general DMDP P_n is illustrated in Appendix B.

4.3 HOWARD'S ALGORITHM ON P_n

Given the DMDP P_n, we show that the sequence of policies that appear in Howard's algorithm is as follows.

$$\begin{aligned} &\pi_1 \to \sigma_{1,1} \to \sigma_{1,2} \to \tau_1 \\ &\to \pi_2 \to \sigma_{2,1} \to \sigma_{2,2} \to \sigma_{2,3} \to \tau_2 \\ &\vdots \\ &\to \pi_{n-1} \to \sigma_{n-1,1} \to \ldots \to \sigma_{n-1,n} \to \tau_{n-1} \\ &\to \pi_n \to \sigma_{n,1} \to \ldots \to \sigma_{n,n-1} \end{aligned} \tag{1}$$

Intuition. The n highest mean-payoff cycles in the graph are the self-loops of t_1 through t_n. At a high level, the algorithm "finds" those cycles one after another, taking roughly i iterations to find the cycle at t_i after finding the cycle at t_{i-1}. This happens because whenever the next-best cycle is found, all "progress" the algorithm made so far in the rest of the graph is lost. In policies π_i, the lasso-shaped runs of all vertices (except t_{i-1}) end up in the cycle at t_i. Now, in the $(\sigma_{i,j})_j$ chain of policies, the vertices keep adding an edge of weight $(n+1)^2$ to the path of their run by including more of the b-vertices (the "deceleration lane"). Since the weight of the deceleration lane edges is greater than the weight of the self-loops, the t-vertices only "realize" they can do better than their current run by using their self-loop when they can no longer improve their path using the deceleration lane. However, by the tie-breaking rule, the vertices only ever add one additional vertex of the deceleration lane to their path, so it takes all $i+1$ iterations from the $(\sigma_{i,j})_j$ chain until the best improvement by appraisal for vertex t_{i+1} is to use its self-loop, which happens in the iteration to τ_i (vertices $t_{i+2}, ..., t_n$ can still do more deceleration lane improvements so don't use their self-loop yet). In the next iteration, all vertices in the deceleration lane as well as $t_{i+2}, ..., t_n$ "realize" that in their current run they do not end up in the highest mean-payoff cycle, since a new, better cycle formed at t_{i+1}. Thus, instead of doing the next improvement in the deceleration lane, they all choose their edge directly to the now-best cycle at t_{i+1}. Thus, all "progress" in the deceleration lane is lost. This continues until the cycle at t_n is formed, when the algorithm does a final run through the deceleration lane before halting in the optimal policy.

Formal Argument. In the following, we formally show that given a policy in the sequence, the Bellman operator returns the next policy in the sequence. The initial policy is π_1, because t_1 is the vertex with lowest index in the ordering and all vertices have outgoing edges to t_1.

Lemma 4.1. *The following assertions hold:*

1. *for $1 \leq i \leq n$, it holds that $\mathcal{B}(\pi_i) = \sigma_{i,1}$;*
2. *for $1 \leq i \leq n$ and $1 \leq j \leq \begin{cases} i & 1 \leq i \leq n-1 \\ n-2 & i = n \end{cases}$, it holds that $\mathcal{B}(\sigma_{i,j}) = \sigma_{i,j+1}$;*
3. *for $1 \leq i \leq n-1$, it holds that $\mathcal{B}(\sigma_{i,i+1}) = \tau_i$;*
4. *for $1 \leq i \leq n-1$, it holds that $\mathcal{B}(\tau_i) = \pi_{i+1}$; and*
5. *$\mathcal{B}(\sigma_{n,n-1}) = \sigma_{n,n-1}$.*

Proof. We formally prove the first two items of the result. We abbreviate the proofs of the rest, where details can be filled in similarly. For notational simplicity, we denote by

$$w_i = w_n(t_i, t_i) = n(n+1) + i$$

the weight of the self-looping edge of vertex t_i and by

$$\bar{w} = (n+1)^2$$

the weight of all other non-zero edges. The following inequalities will be essential throughout the proof:

- it holds that

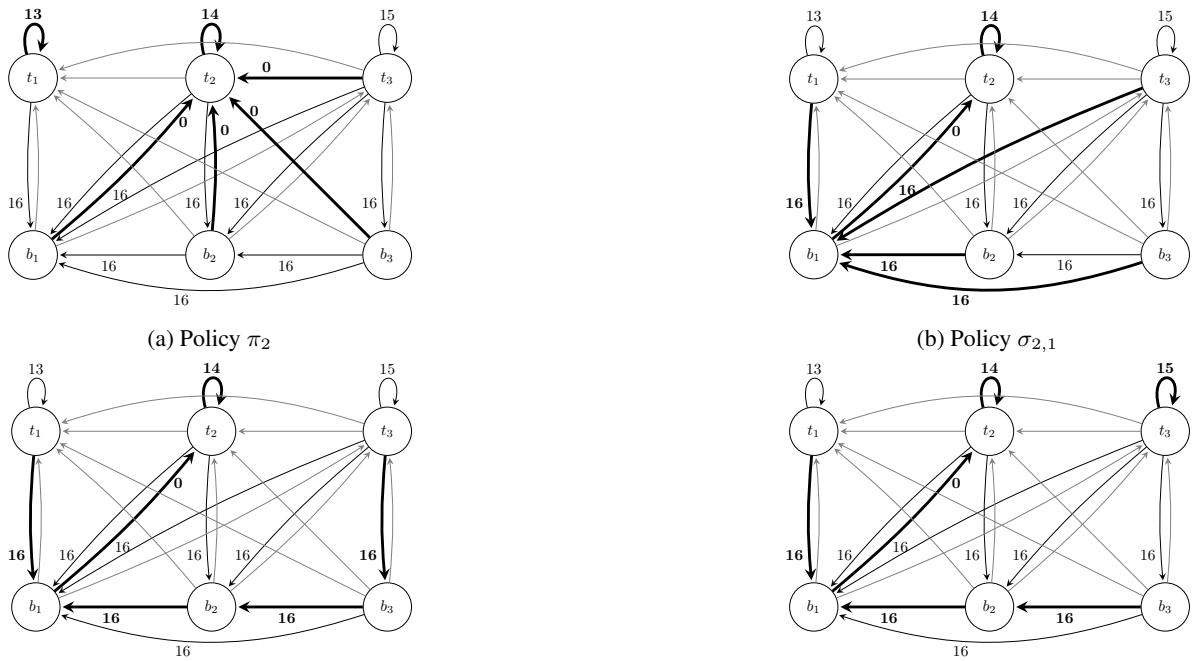
$$\bar{w} > w_n > \cdots > w_i > \cdots > w_1 > 0\,; \tag{2}$$

- for all $0 \leq k \leq n$ and $1 \leq i \leq n$, it holds that

$$k\bar{w} < (k+1)w_i\,; \tag{3}$$

- for all $k \leq n+1$ and $1 \leq i \leq n$, it holds that

$$(k-1)\bar{w} - kw_i < k\bar{w} - (k+1)w_i\,. \tag{4}$$



(a) Policy π_2

(b) Policy $\sigma_{2,1}$

(c) Policy $\sigma_{2,3}$

(d) Policy τ_2

Figure 2: Examples of policies. Thick lines correspond to policy choices. Unlabeled (gray) edges have weight 0.

Proof of Item 1. We first give a quick intuitive rationale: In policy π_i, all vertices that have an edge to directly move to t_i, the highest-value cycle of value w_i, use it. t_{i-1} loops to itself and the vertices $t_k, k \leq i-2$, that have no edge directly to t_i, use the edge to b_k (which uses the edge to t_i). Since t_i is the highest-value cycle, all vertices will pick their next edge so that they end up there, and between their options choose to maximize the weight of the path to t_i, minus w_i times the path length. All the vertices that currently use the edge directly to t_i, and t_{i-1}, can improve this quantity of the path to t_i by using one of their edges to any of the b-vertices, thus adding an edge of weight $\bar{w} > w_i$ to their path. By the tie breaking rule they pick the edge to b_1. The remaining vertices, t_k, for $k \leq i-2$, in this iteration cannot increase the average weight of their path to t_i, so do not change their used edge.

We now formalize this intuition. First, we compute the mean-payoff for the policy π_i. For all vertices $v \in V \setminus \{t_{i-1}\}$, the cycle of the lasso-shaped play $P_v^{\pi_i}$ is of form $\langle t_i \rangle$. Since $\pi_i(t_{i-1}) = t_{i-1}$, therefore the cycle of $P_{t_{i-1}}^{\pi_i}$ is of form $\langle t_{i-1} \rangle$. Therefore, the mean-payoff for the policy π_i is

$$\mathrm{val}^{\pi_i}(v) = \begin{cases} w_{i-1} & v = t_{i-1} \\ w_i & \text{otherwise}\,. \end{cases} \tag{5}$$

We now compute the potential function pot^{π_i}. Fo vertices t_{i-1} and t_i, the potential is 0, because the path of their lasso-shaped play is of length 0. For $1 \leq k \leq i-2$, the path of the play $P_{t_k}^{\pi_i}$ is of form $\langle t_k, b_k, t_i \rangle$. Therefore, we have

$$\begin{aligned} \mathrm{pot}^{\pi_i}(t_k) &= (\bar{w} - w_i) + (0 - w_i) \\ &= \bar{w} - 2w_i\,. \end{aligned}$$

For $1 \leq k \leq n$, the path of the play $P_{b_k}^{\pi_i}$ is of form $\langle b_k, t_i \rangle$. Since the weight of the edge (b_k, t_i) is 0, the potential of the vertex b_k is $\mathrm{pot}^{\pi_i}(b_k) = -w_i$. For $i < k$, the path of the play $P_{t_k}^{\pi_i}$ is of form $\langle t_k, t_i \rangle$. Since the weight of the edge (t_k, t_i) is 0, the potential of the vertex t_k is $\mathrm{pot}^{\pi_i}(t_k) = -w_i$. To consolidate the previous computations, the potential function for the policy π_i is

$$\mathrm{pot}^{\pi_i}(v) = \begin{cases} 0 & v = t_k \quad \forall k \in \{i-1, i\} \\ \bar{w} - 2w_i & v = t_k \quad \forall k \leq i-2 \\ -w_i & v = b_k \quad \forall k \leq n \\ -w_i & v = t_k \quad \forall k > i \end{cases} \tag{6}$$

We are now ready to show that $\mathcal{B}(\pi_i)(v) = \sigma_{i,1}(v)$ for all $v \in V_n$. Recall that the appraisal for edge (v, u) under π_i is:

$$\mathcal{A}^{\pi_i}(v, u) = (\mathrm{val}^{\pi_i}(u),\ w_n(v, u) - \mathrm{val}^{\pi_i}(u) + \mathrm{pot}^{\pi_i}(u))\,,$$

compared lexicographically with tie-breaking as described in Section 2. By Equation (5) we know that the first element of the appraisal is maximized by all $u \in E(v) \setminus \{t_{i-1}\}$. Thus, the second element of the appraisal, $\mathcal{A}_2^{\pi_i}(v, u)$, is deciding. We now precede by, unfortunately quite tediously but inevitability, calculating $\mathcal{A}_2^{\pi_i}(v, u)$ for all $v \in V$ and $u \in E(v) \setminus \{t_{i-1}\}$.

For all $j \leq i-2$, $v = t_j$, we get that

$$\mathcal{A}_2^{\pi_i}(v,u) = \begin{cases} \bar{w} - 3w_i & u = t_k, k \leq j-1 \\ \bar{w} + w_j - 3w_i & u = t_j \\ \bar{w} - 2w_i & u = b_k, k \leq j \end{cases}.$$

This is maximized by $u = b_k$ for any $k \leq j$. By the tie-breaking rule t_j favors $\pi_i(t_j) = b_j$ if possible, so that $\mathcal{B}(\pi_i)(t_j) = b_j = \sigma_{i,1}(t_j)$.

For $v = t_{i-1}$, we get that

$$\mathcal{A}_2^{\pi_i}(v,u) = \begin{cases} \bar{w} - 3w_i & u = t_k, k \leq i-2 \\ \bar{w} - 2w_i & u = b_k, k \leq i-1 \end{cases}.$$

This is maximized by $u = b_k$ for any $k \leq i-1$. Since $\pi_i(t_{i-1}) = t_{i-1}$, the tie-breaking rule will favor based on the vertex ordering, so that $\mathcal{B}(\pi_i)(t_{i-1}) = b_1 = \sigma_{i,1}(t_{i-1})$.

For $v = t_i$, we get that

$$\mathcal{A}_2^{\pi_i}(v,u) = \begin{cases} \bar{w} - 3w_i & u = t_k, k \leq i-2 \\ 0 & u = t_i \\ \bar{w} - 2w_i & u = b_k, k \leq i \end{cases}.$$

This is maximized by $u = t_i$. Thus, it holds that $\mathcal{B}(\pi_i)(t_i) = t_i = \sigma_{i,1}(t_i)$.

For all $j > i$, $v = t_j$, we get that

$$\mathcal{A}_2^{\pi_i}(v,u) = \begin{cases} \bar{w} - 3w_i & u = t_k, k \leq i-2 \\ -w_i & u = t_i \\ -2w_i & u = t_k, i < k < j \\ w_j - 2w_i & u = t_j \\ \bar{w} - 2w_i & u = b_k, k \leq j \end{cases}.$$

This is maximized by $u = b_k$ for any $k \leq j$. Since $\pi_i(t_j) = t_i$, the tie-breaking rule will favor based on the vertex ordering, so that $\mathcal{B}(\pi_i)(t_j) = b_1 = \sigma_{i,1}(t_j)$.

For all $1 \leq j \leq n$, $v = b_j$, we get that

$$\mathcal{A}_2^{\pi_i}(v,u) = \begin{cases} \bar{w} - 3w_i & u = t_k, k \leq i-2 \\ -w_i & u = t_i \\ -2w_i & u = t_k, k > i \\ \bar{w} - 2w_i & u = b_k, k < j \end{cases}.$$

For $j \geq 2$, this is maximized by $u = b_k$ for any $k < j$. Since $\pi_i(t_j) = t_i$, the tie-breaking rule will favor based on the vertex ordering, so that $\mathcal{B}(\pi_i)(b_j) = b_1 = \sigma_{i,1}(b_j)$. For $j = 1$, it is maximized for $u = t_i$, so $\mathcal{B}(\pi_i)(b_1) = t_i = \sigma_{i,1}(b_1)$. Thus, we conclude that $\mathcal{B}(\pi_i) = \sigma_{i,1}$.

Proof of Item 2. We first give a quick intuitive rationale: In $\sigma_{i,j}$, all paths end up in the same cycle, the self-loop at t_i with value w_i. Thus, in the iteration to $\sigma_{i,j+1}$, the other vertices choose their next edge solely to maximize the weight of their path to t_i, minus w_i times the path length. The optimal path for that is using as much of the deceleration lane $\langle b_n, ..., b_1 \rangle$ as possible, since its edges have weight $\bar{w} > w_i$. $b_1, ..., b_{j+1}$ already move to their predecessor, so for all vertices that can (except t_i), it is optimal to pick the edge to to b_{j+1} (which they do, by the tie-breaking rule) to maximize the edges in the deceleration lane in their path. Thus, in $\sigma_{i,j+1}$ the vertices that can, all pick the edge to b_{j+1} (except t_i), while the other vertices cannot further improve their path and hence pick the same edge as in $\sigma_{i,j}$.

Formally, we consider policy $\sigma_{i,j}$ for any $1 \leq i \leq n$ and $1 \leq j \leq \begin{cases} i & 1 \leq i \leq n-1 \\ n-2 & i = n \end{cases}$. For all vertices $v \in V$, the cycle of the lasso-shaped play $P^{\sigma_{i,j}}$ is $\langle t_i \rangle$. Therefore, for all vertices $v \in V$, it holds that $\text{val}^{\sigma_{i,j}}(v) = w_i$. Hence, $\mathcal{B}(\sigma_{i,j})(v)$ solely depends on $\mathcal{A}_2^{\sigma_{i,j}}(v,u)$. We precede by calculating $\mathcal{A}_2^{\sigma_{i,j}}(v,u)$. for all $v \in V$ and $u \in E(v)$. We will first assume $j > 1$, since $\sigma^{i,j}$ is slightly different at $j = 1$. We will treat this special case at the end.

For $v = t_i$, we get that

$$\mathcal{A}_2^{\sigma_{i,j}}(v,u) = \begin{cases} k\bar{w} - (k+2)w_i & u = t_k, k \leq j \\ j\bar{w} - (j+2)w_i & u = t_k, j < k < i \\ 0 & u = t_i \\ k\bar{w} - (k+1)w_i & u = b_k, k \leq j \\ (j+1)\bar{w} - (j+2)w_i & u = b_k, j < k \leq i \end{cases}.$$

By Equations (2) to (4), we know that this is maximized by $u = t_i$ so that $\mathcal{B}(\sigma_{i,j})(t_i) = t_i = \sigma_{i,j+1}(t_i)$.

For $j < h < i$, $v = t_h$, we get that

$$\mathcal{A}_2^{\sigma_{i,j}}(v,u) = \begin{cases} k\bar{w} - (k+2)w_i & u = t_k, k \leq j \\ j\bar{w} - (j+2)w_i & u = t_k, j < k < h \\ j\bar{w} + w_h - (j+2)w_i & u = t_h \\ k\bar{w} - (k+1)w_i & u = b_k, k \leq j \\ (j+1)\bar{w} - (j+2)w_i & u = b_k, j < k \leq h \end{cases}.$$

By Equations (2) to (4), this is maximized by $u = b_k$ for all $j < k \leq h$. Since $\sigma_{i,j}(v) = b_j$, tie-breaking is done by vertex order so that $\mathcal{B}(\sigma_{i,j})(t_h) = b_{j+1} = \sigma_{i,j+1}(t_h)$.

For $h \leq j, h < i$, $v = t_h$, we get that

$$\mathcal{A}_2^{\sigma_{i,j}}(v,u) = \begin{cases} k\bar{w} - (k+2)w_i & u = t_k, k < h \\ h\bar{w} + w_h - (h+2)w_i & u = t_h \\ k\bar{w} - (k+1)w_i & u = b_k, k \leq h \end{cases}.$$

By Equations (2) to (4), this is maximized by $u = b_h$ so that $\mathcal{B}(\sigma_{i,j})(t_h) = b_h = \sigma_{t_h}$.

For $i < h$, $v = t_h$, we get that

$$\mathcal{A}_2^{\sigma_{i,j}}(v,u) = \begin{cases} k\bar{w} - (k+2)w_i & u = t_k, k \leq j \\ j\bar{w} - (j+2)w_i & u = t_k, j < k < h, k \neq i \\ -w_i & u = t_i \\ j\bar{w} + w_h - (j+2)w_i & u = t_h \\ k\bar{w} - (k+1)w_i & u = b_k, k \leq j \\ (j+1)\bar{w} - (j+2)w_i & u = b_k, j < k \leq h \end{cases}.$$

By Equations (2) to (4), this is maximized by $u = b_k$ for all $j < k \leq h$. Since $\sigma_{i,j}(v) = b_j$, tie-breaking is done by vertex order so that $\mathcal{B}(\sigma_{i,j})(t_h) = b_{j+1} = \sigma_{i,j+1}(t_h)$.

For $1 \leq h \leq n$, $v = b_h$, we get that

$$\mathcal{A}_2^{\sigma_{i,j}}(v,u) = \begin{cases} k\bar{w} - (k+2)w_i & u = t_k, k \leq j \\ j\bar{w} - (j+2)w_i & u = t_k, j < k, k \neq i \\ -w_i & u = t_i \\ k\bar{w} - (k+1)w_i & u = b_k, k \leq j, k < h \\ (j+1)\bar{w} - (j+2)w_i & u = b_k, j < k < h \end{cases}.$$

By Equations (2) to (4), for $j + 1 \geq h$, this is maximized by $u = b_{h-1}$, so that $\mathcal{B}(\sigma_{i,j})(b_h) = b_{h-1} = \sigma_{i,j+1}(b_h)$. For $j + 1 < h$, this is maximized by b_k for all $j < k < h$. Since $\sigma_{i,j}(b_h) = b_j$, tie-breaking is done by vertex order so that $\mathcal{B}(\sigma_{i,j})(b_h) = b_{j+1} = \sigma_{i,j+1}(b_h)$.

Consider $j = 1$. $\mathcal{A}_2^{\sigma_{i,j}}(v,u)$ generally is unchanged from the case $j > 1$, except for $u = t_k$ for $1 < k \leq i - 2$. In these cases, due to the one extra edge in the path of run $P_v^{\sigma_{i,j}}$, $\mathcal{A}_2^{\sigma_{i,j}}(v,u)$ is greater by $\bar{w} - w_i$. This does not change which u's maximize $\mathcal{A}_2^{\sigma_{i,j}}(v,u)$, so the results for $\mathcal{B}(\sigma_{i,j})$ outlined above for $j > 1$ hold for $j = 1$ as well.

We conclude that $\mathcal{B}(\sigma_{i,j}) = \sigma_{i,j+1}$.

Proof of Item 3. This case is highly similar to Item 2. In the iteration on $\sigma_{i,i+1}$, all vertices except t_{i+1} follow the same pattern as for the iteration from $\sigma_{i,j}$ to $\sigma_{i,j+1}$ for $j \leq i$. Vertex t_{i+1}, though, can no longer use an edge to a higher b_k to add another edge of weight $\bar{w} > w_i$ to its path to t_i. However, (differently to the vertices t_k for $k < i$), the edge to itself has a value higher than the value of its current run's cycle $\langle t_i \rangle$, i.e. $w_{i+1} > w_i$, so it will use this edge to add it to its path. Thus, in the iteration from $\sigma_{i,i+1}$ to τ_i, all vertices will behave as in Item 2, except for t_{i+1}, which picks its self-loop. Formally, for all $v, u \in V$, $\mathcal{A}^{\sigma_{i,j}}(v,u)$ for $j = i + 1$ follows the same pattern as for $j \leq i$. We can check that for all vertices $v \neq t_{i+1}$, also the pattern of which u's maximize it stays unchanged, so that $\mathcal{B}(\sigma_{i,i+1})(v) = \sigma_{i,i+2}(v) = \tau_i(v)$ [1] for $v \neq t_{i+1}$. For $v = t_{i+1}$, by Equations (2) to (4) $\mathcal{A}_2^{\sigma_{i,i+1}}(t_{i+1},u)$ (and thus $\mathcal{A}^{\sigma_{i,i+1}}(t_{i+1},u)$) is maximized by $u = t_{i+1}$ so that $\mathcal{B}(\sigma_{i,i+1})(t_{i+1}) = t_{i+1} = \tau_i(t_{i+1})$.

Proof of Item 4. Intuitively, in τ_i, the best cycle is $\langle t_{i+1} \rangle$ but currently no vertex other than t_{i+1} ends up there. Thus, t_{i+1} has the uniquely highest value, so all vertices with edges to t_{i+1} will pick them. For vertices that do not have an edge to t_{i+1}, those are t_k for $k \leq i + 1$, this iteration step still is identical to the iteration from $\sigma_{i,i}$ to $\sigma_{i,i+1}$ in which they did not change the edge they use. Formally, for all vertices $v \in V \setminus \{t_{i+1}\}$, the cycle of the lasso-shaped play $P_v^{\tau_i}$ is of form $\langle t_i \rangle$. The cycle of $P_{t_{i+1}}^{\tau_i}$ is of form $\langle t_{i+1} \rangle$. Hence, the mean-payoff for the policy τ_i is

$$\mathrm{val}^{\tau_i}(v) = \begin{cases} w_{i+1} & v = t_{i+1} \\ w_i & \text{otherwise}. \end{cases}$$

Since $\mathrm{val}^{\tau_i}(v)$ is uniquely maximized by $v = t_{i+1}$, $\mathcal{B}(\tau_i)(v) = t_{i+1}$ for all v where $t_{i+1} \in E(v)$, i.e., for $v = b_k, 1 \leq k \leq n$ and $v = t_k, i + 1 \leq k \leq n$. For the remaining vertices, $v = t_k, 1 \leq k \leq i$, their appraisal function $\mathcal{A}^{\tau_i}(v,u)$ is still the same as $\mathcal{A}^{\sigma_{i,i}}(v,u)$. Thus, $\mathcal{B}(\tau_i)(t_k) = \mathcal{B}(\sigma_{i,i})(t_k) = \sigma_{i,i+1}(t_k) = \pi_{i+1}(t_k)$.

Proof of Item 5. Again, this case is very similar to Item 2. In the general case for the iteration from $\sigma_{i,j}$ to $\sigma_{i,j+1}$, only vertices t_k with $k \geq j + 1, k \neq i$ and b_k with $k \geq j + 2$ change the edge they use. Since we reached the end of the deceleration lane and the highest-value self-loop is being used, there are no more vertices that match these criteria, so no vertex changes the edge it uses. Formally, $\mathcal{A}^{\sigma_{i,j}}(v,u)$ for $i = j + 1 = n$ follows the same pattern as for all values of i, j considered in Item 2. For all vertices $v \in V$, the pattern of which u's maximize it also stays unchanged, so that $\mathcal{B}(\sigma_{n,n-1}) = \sigma_{n,n} = \sigma_{n,n-1}$. □

Proof of Theorem 3.2. The constructed DMDP has $2n$ vertices and $\frac{3n^2+n}{2}$ edges. The absolute value of weights is $O(n^3)$. Therefore, the size of the DMDP is $O(n^2 \log n)$.

Lemma 4.1 shows that if Howard's algorithm starts with the policy π_1, it iterates over all policies in the sequence shown in Equation (1). Therefore, the length of the sequence is

$$2n + \sum_{i=1}^{n}(i+1) - 3 = \frac{n^2 + 7n - 6}{2},$$

where the equality follows from the sum of arithmetic series and algebraic rearrangement of terms, which yields the result. □

[1] Note that policy $\sigma_{i,i+2}$ does not appear in the algorithm, but we use it here to illustrate this point.

4.4 EXPERIMENTAL EVALUATION

To validate our lower bound example, we implemented both Howard's policy iteration and the example in Python. The experimental evaluation confirms the sequence of policies shown by the theoretical analysis. The full implementation is publicly available at `https://doi.org/10.5281/zenodo.14823415`.

5 EXTENSIONS

We discuss several extensions of Theorem 3.2.

Policy Initialization. The number of iterations that Howard's Policy Iteration algorithm performs depends on the choice of the initial policy σ_0. Two natural choices are for each vertex to use (a) the edge to it's lowest-index neighbor or (b) the highest-weight outgoing edge (breaking ties by vertex indices). Option (b) is the most common in literature and was used in Howard [1960]. While our lower bound proof above uses option (a) to get the starting policy, it is noteworthy that our lower bound also holds for (b). In particular, we let

$$\sigma_0(v) \coloneqq \arg\max_{u \in E(v)} w(v,u) = \begin{cases} t_1 & v = b_1 \\ b_1 & \text{otherwise} \end{cases}.$$

One iteration of Howard's algorithm on σ_0 leads to $\sigma_{1,2}$, from which on the algorithm proceeds as described above, iterating over the policies in the sequence shown in Equation (1) starting at $\sigma_{1,2}$. Compared to using π_1 as an initial policy, Howard's algorithm with initial policy σ_0 only performs one iteration less. Thus, the number of iterations for this policy initialization is still $\Omega(n^2)$.

Discounted-sum Objectives. In discounted-sum objectives, every edge is assigned an integer weight and the payoff is the discounted sum of these weights. Although Theorem 3.2 is stated for mean-payoff objectives, it extends to discounted-sum objectives with a discount factor sufficiently close to 1 as a function of n, because as the discount factor approaches 1, discounting diminishes and the sum converges to the mean-payoff value. Furthermore, by Blackwell optimality [Blackwell, 1962], an optimal policy optimal for the mean-payoff objectives remains optimal for all discount factors sufficiently near 1.

It is an open question whether our techniques can be extended to obtain a similar lower bound for a discounted-sum objective with a constant discount factor. In this setting, a lower bound of $\Omega(n^2)$ on the number of iterations would be tight up to a factor of $\log n$ to the upper bound due to Hansen et al. [2013], which applies not only to DMDPs, but also stochastic MDPs and in the 2-player setting.

6 CONCLUSION AND FUTURE WORK

In this work, we studied Howard's policy iteration algorithm for DMDPs with mean-payoff objectives and constructed a family of examples with $2n$ vertices and $O(n^2)$ edges where the algorithm requires $\Omega(n^2)$ iterations and improved the lower bound on the number of iterations to $\widetilde{\Omega}(I)$ with respect to the input size I. There are several interesting directions for future work. In particular, Hansen's conjecture [Hansen, 2012] on the number of iterations remains a major open problem. Furthermore, the practical performance of Howard's policy iteration, despite its high theoretical worst-case complexity, raises relevant and interesting questions. While our focus is to establish an improved theoretical lower bound for this classical algorithm, these practical concerns highlight important directions for future research.

Acknowledgements

This research was partially supported by the ERC CoG 863818 (ForM-SMArt) grant and Austrian Science Fund (FWF) 10.55776/COE12.

References

R Alur. *Principles of Cyber-Physical Systems*. The MIT Press, 2015.

Raman Arora, Ofer Dekel, and Ambuj Tewari. Deterministic MDPs with adversarial rewards and bandit feedback. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pages 93–101, 2012.

Ali Asadi, Krishnendu Chatterjee, Jakub Svoboda, and Raimundo Saona Urmeneta. Deterministic subexponential algorithm for discounted-sum games with unary weights. In *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 1–12, 2024.

Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

David Blackwell. Discrete dynamic programming. *The Annals of Mathematical Statistics*, pages 719–726, 1962.

Vincent D Blondel and John N Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274, 2000.

Victor Boone and Bruno Gaujal. Identification of blackwell optimal policies for deterministic MDPs. In *International Conference on Artificial Intelligence and Statistics*, pages 7392–7424. PMLR, 2023.

Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic Markov Decision Processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10069–10076, 2020.

Miranda Christ and Mihalis Yannakakis. The smoothed complexity of policy iteration for markov decision processes. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1890–1903, 2023.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

Ali Dasdan. Experimental analysis of the fastest optimum cycle ratio and mean algorithms. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 9 (4):385–418, 2004.

Cyrus Derman. *Finite state Markovian decision processes*. Academic Press, Inc., 1970.

John Fearnley. Exponential lower bounds for policy iteration. In *Automata, Languages and Programming: 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part II 37*, pages 551–562. Springer, 2010.

Jerzy Filar and Koos Vrieze. *Competitive Markov decision processes*. Springer Science & Business Media, 2012.

Oliver Friedmann. An exponential lower bound for the parity game strategy improvement algorithm as we know it. In *2009 24th Annual IEEE Symposium on Logic In Computer Science*, pages 145–156. IEEE, 2009.

Oliver Friedmann, Thomas Dueholm Hansen, and Uri Zwick. Subexponential lower bounds for randomized pivoting rules for the simplex algorithm. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 283–292, 2011.

Thomas Dueholm Hansen. *Worst-case analysis of strategy iteration and the simplex method*. PhD thesis, Aarhus Universitet, Datalogisk Institut, 2012.

Thomas Dueholm Hansen and Uri Zwick. Lower bounds for howard's algorithm for finding minimum mean-cost cycles. In *International Symposium on Algorithms and Computation*, pages 415–426. Springer, 2010.

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. *JACM*, 60(1):1–16, 2013.

Ronald A Howard. Dynamic programming and markov processes. *MIT Press google schola*, 2:39–47, 1960.

Richard M Karp. A characterization of the minimum cycle mean in a digraph. *Discrete mathematics*, 23(3):309–311, 1978.

Michael L Littman. Probabilistic propositional planning: Representations and complexity. *AAAI/IAAI*, pages 748–754, 1997.

Bruno Loff and Mateusz Skomra. Smoothed analysis of deterministic discounted and mean-payoff games. In *51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*, pages 147–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024.

Omid Madani. Polynomial value iteration algorithms for deterministic MDPs. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 311–318, 2002.

Omid Madani, Mikkel Thorup, and Uri Zwick. Discounted deterministic Markov decision processes and discounted all-pairs shortest paths. In *SODA*, pages 958–967. SIAM, 2009.

Ian Post and Yinyu Ye. The simplex method is strongly polynomial for deterministic Markov decision processes. *Mathematics of Operations Research*, 40(4):859–868, 2015.

Martin L. Puterman. *Markov Decision Processes*. John Wiley and Sons, 1994.

Neal E Young, Robert E Tarjant, and James B Orlin. Faster parametric shortest path and minimum-balance algorithms. *Networks*, 21(2):205–221, 1991.

Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1-2):343–359, 1996.

Lower Bound on Howard Policy Iteration for Deterministic Markov Decision Processes
(Supplementary Material)

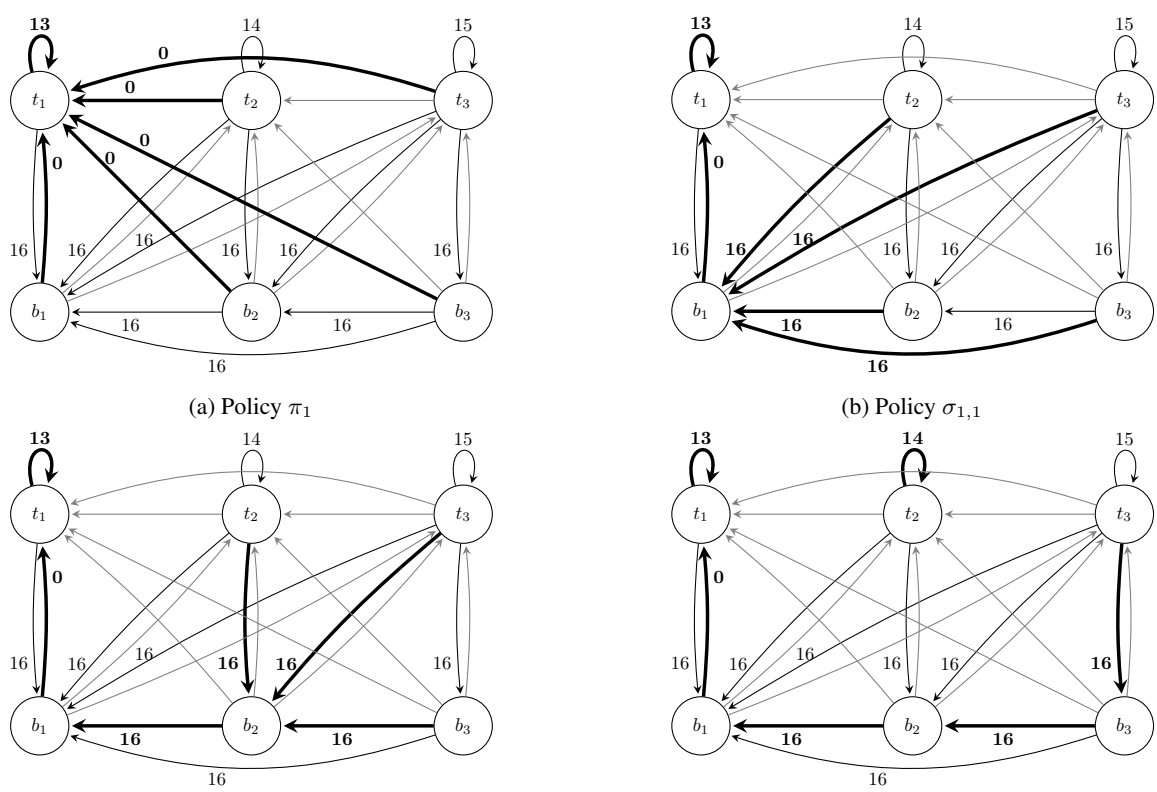
Ali Asadi[1] **Krishnendu Chatterjee**[1] **Jakob de Raaij**[2]

[1]Institute of Science and Technology Austria (ISTA)
[2]Harvard University

A SEQUENCE OF POLICIES

In this section, we illustrate the sequence of policies appearing in Howard's policy iteration on the DMDP P_3 and for the general P_n.

A.1 THE POLICIES FOR OUR RUNNING EXAMPLE



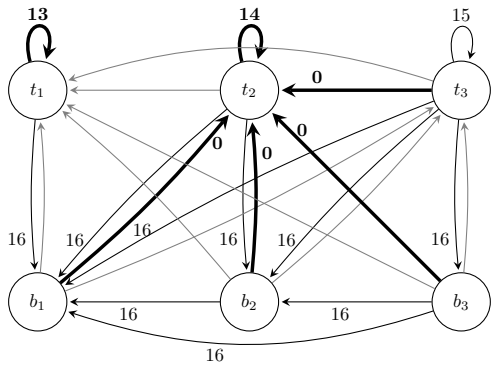
(a) Policy π_1

(b) Policy $\sigma_{1,1}$

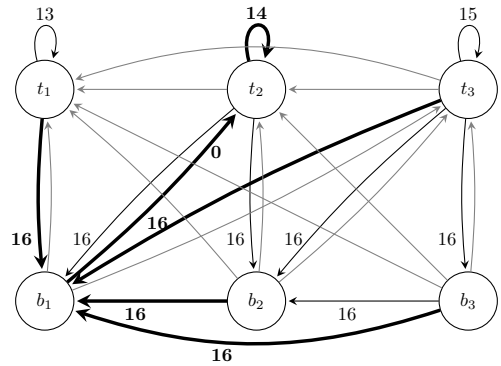
(c) Policy $\sigma_{1,2}$

(d) Policy τ_1

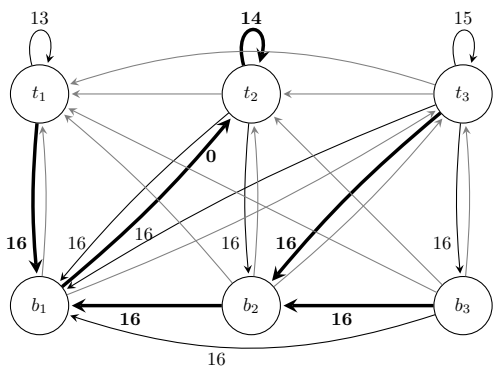
Figure 3: Part I: the sequence of policies appearing in Howard's policy iteration over our running example. Thick lines correspond to policy choices. Unlabeled (gray) edges have weight 0.



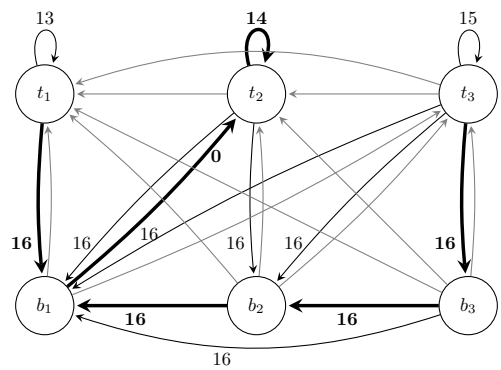
(e) Policy π_2



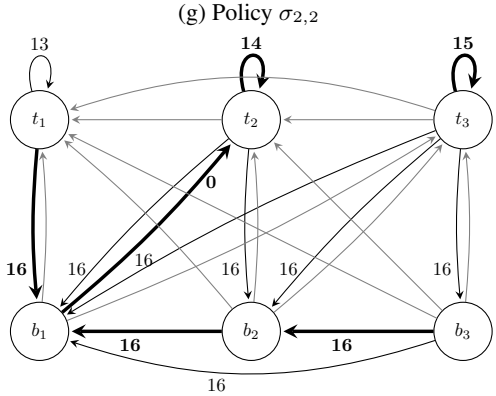
(f) Policy $\sigma_{2,1}$



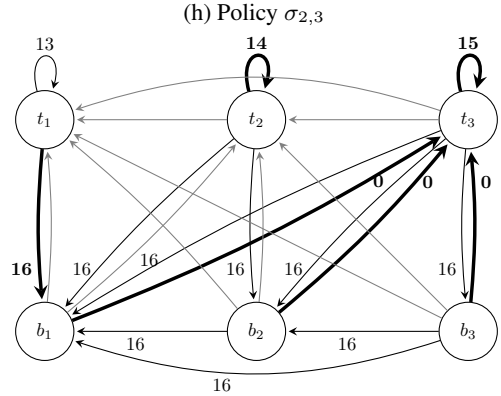
(g) Policy $\sigma_{2,2}$



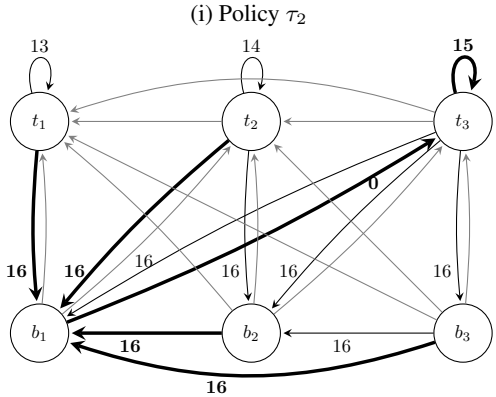
(h) Policy $\sigma_{2,3}$



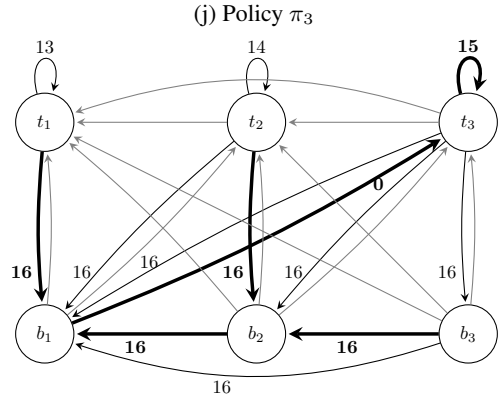
(i) Policy τ_2



(j) Policy π_3



(k) Policy $\sigma_{3,1}$

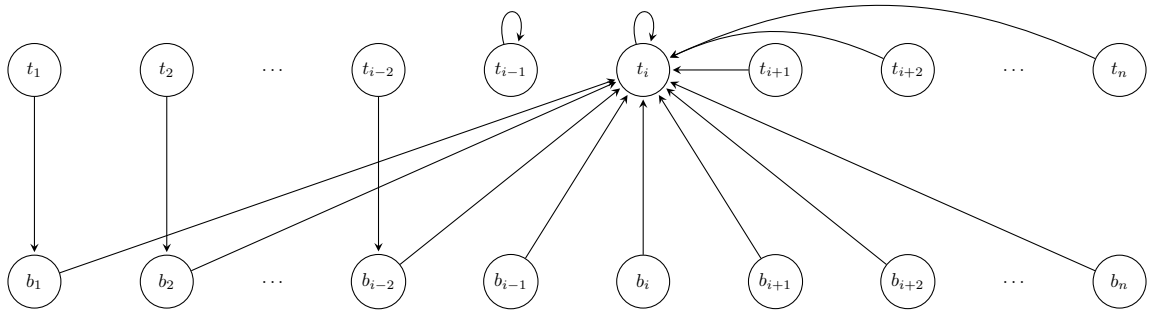


(l) Policy $\sigma_{3,2}$

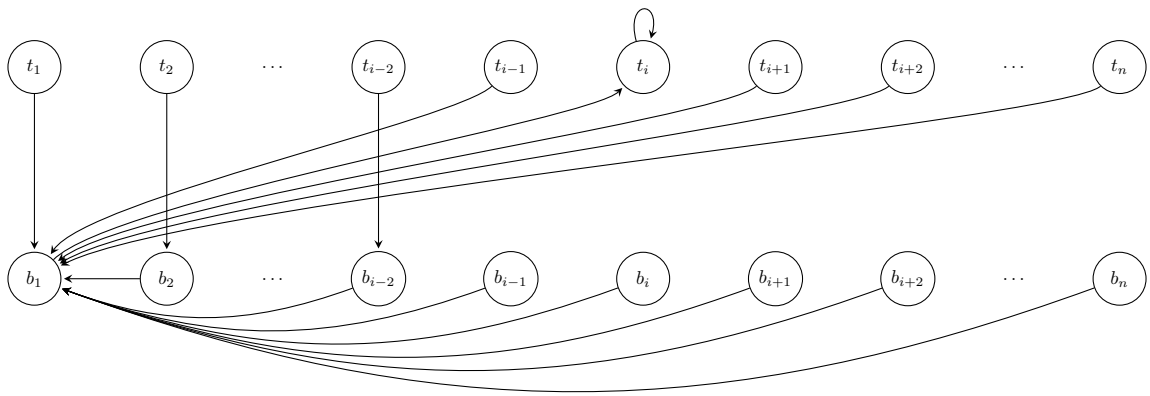
Figure 3: Part II: the sequence of policies appearing in Howard's policy iteration over our running example. Thick lines correspond to policy choices. Unlabeled (gray) edges have weight 0.

B THE POLICIES IN THE GENERAL CASE

We illustrate the sequence of policies that appear from policy π_i to policy π_{i+1}. To keep the display clear, we omit edge weights and edges not in the policy from the figures and show only the edges in the current policy.



(a) Policy π_i



(b) Policy $\sigma_{i,1}$.

Figure 4: Part I: policies appearing in Howard's policy iteration on P_n. Only edges in the policy are shown; edge weights are omitted.

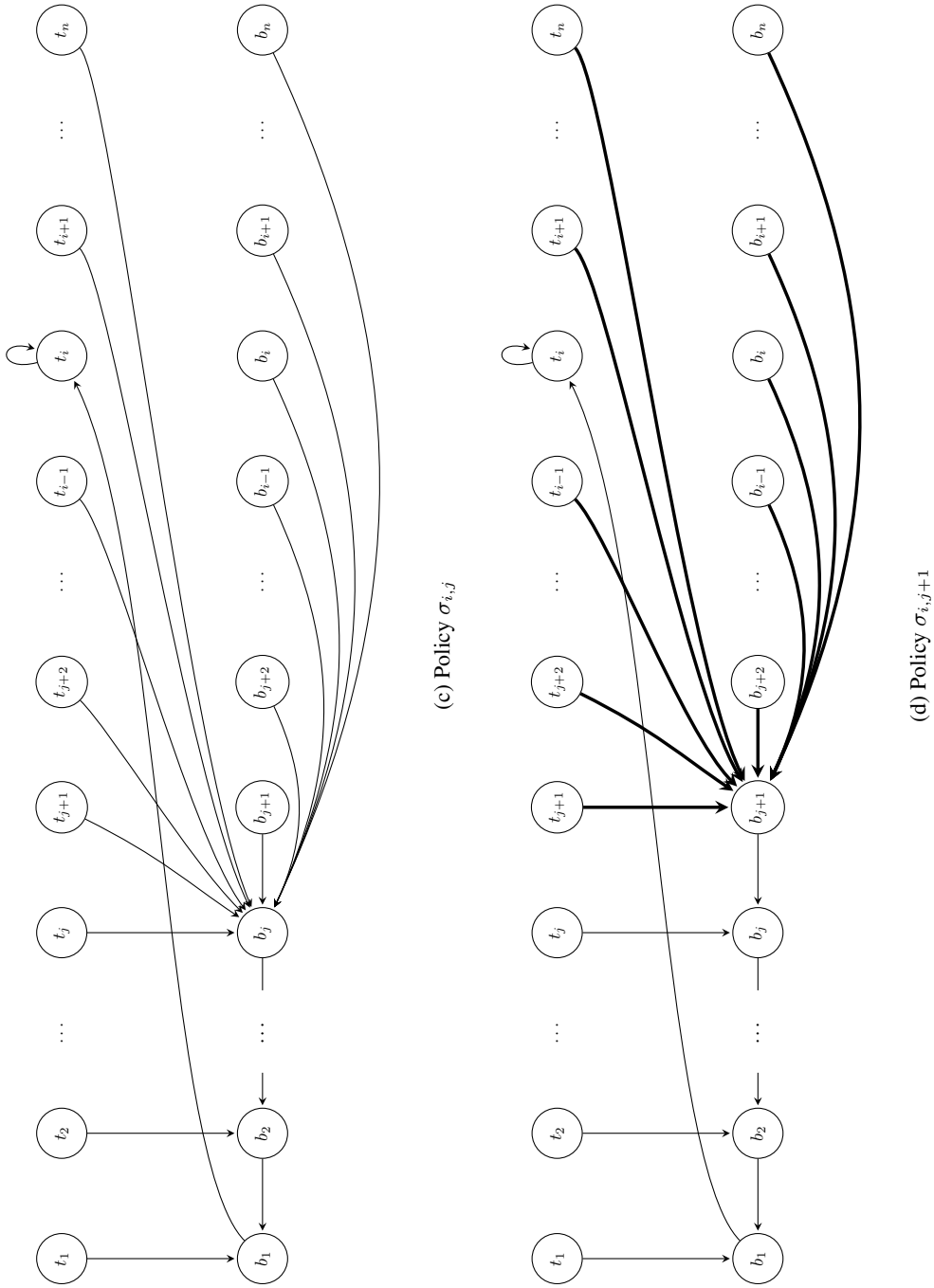
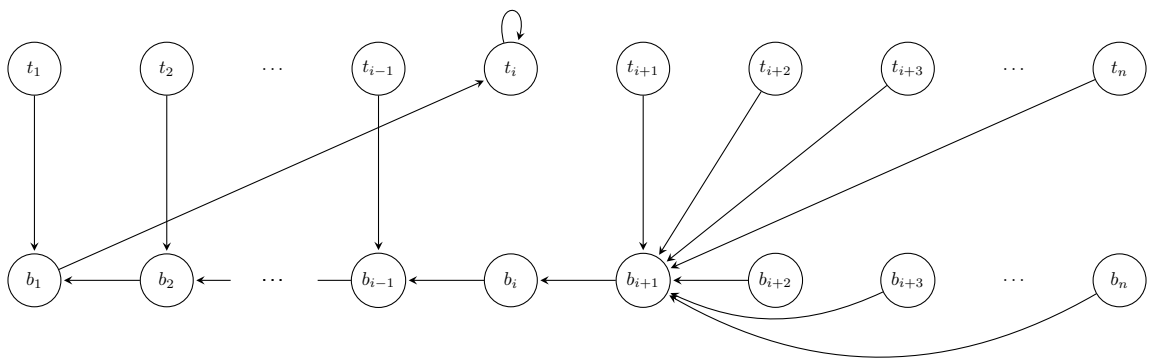
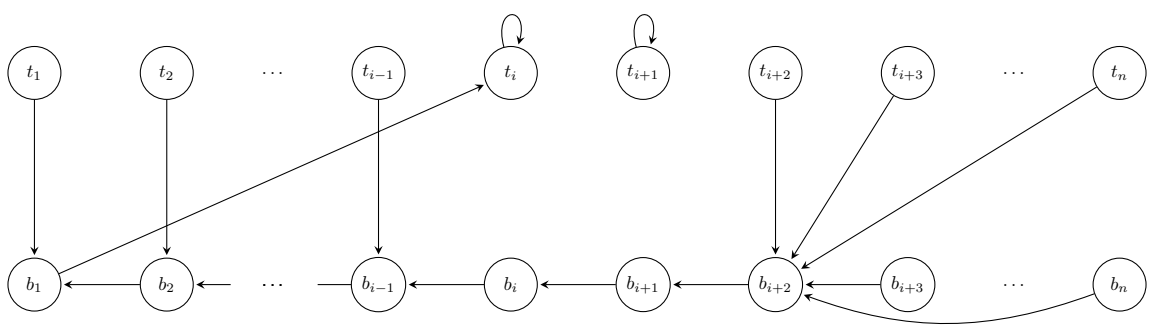


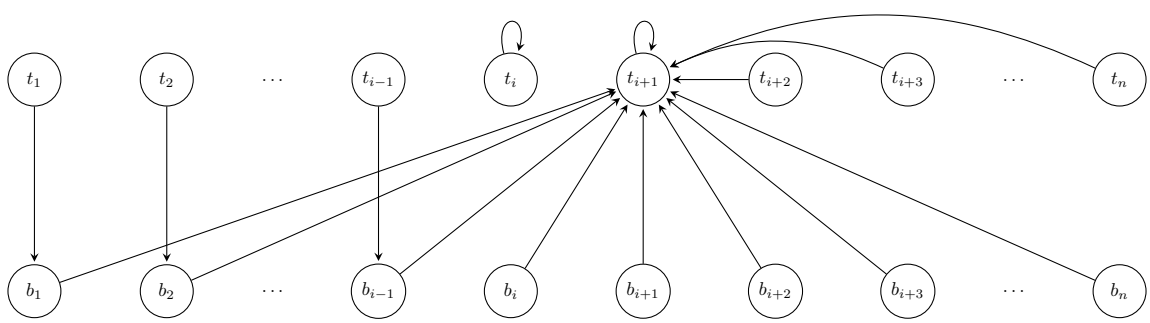
Figure 4: Part II: Comparison of policies $\sigma_{i,j}$ and $\sigma_{i,j+1}$, with $j > 1$, appearing in Howard's policy iteration on P_n. Only edges in the policy are shown; edge weights are omitted. The edges in $\sigma_{i,j+1}$ that differ from $\sigma_{i,j}$ are bold.



(e) Policy $\sigma_{i,i+1}$



(f) Policy τ_i



(g) Policy π_{i+1}

Figure 4: Part III: policies appearing in Howard's policy iteration on P_n. Only edges in the policy are shown; edge weights are omitted.