Categorizer Automata for Discounted-Sum Payoffs

Nathalie Bertrand[1], **Pranav Ghorpade**[2], **Senthil Rajasekaran**[3], **Sasha Rubin**[2], **Moshe Vardi**[4]

[1]Univ Rennes, Inria, CNRS, IRISA, France
[2]University of Sydney, Australia
[3]Université Libre de Bruxelles, Belgium
[4] Rice University, USA

Abstract

Categorizing continuous data into discrete bins is a fundamental operation in artificial intelligence. We introduce the *categorizer automaton*, a deterministic automaton that reads an infinite sequence of rewards and identifies which of finitely many bins contains its discounted sum. Categorizer automata generalize comparator automata, the special case of two bins, which have already proven useful in quantitative synthesis. Our main technical contribution is the construction of a categorizer automaton whose state space is linear in the number of bins, rather than exponential as obtained by a cross-product of comparator automata. We then apply categorizer automata to Markov decision processes, where they allow one to synthesize policies that maximize the expected utility of a discounted-sum payoff for utility functions that may be *discontinuous*. For piecewise-constant utility functions, the resulting algorithm is exact and runs in pseudo-polynomial time. For piecewise-Lipschitz utility functions, a class that includes any utility with bounded slope between finitely many jumps, it again runs in pseudo-polynomial time and yields an ε-optimal policy. We also show that the synthesis problem considered is PSPACE-hard already for piecewise-constant utilities.

1 Introduction

The fields of verification and sequential decision-making often involve assigning a numerical payoff to an infinite-length execution of a system. One of the most popular ways to reason about such executions is through the discounted-sum payoff (de Alfaro, Henzinger, and Majumdar 2003; Puterman 1994), which aggregates an infinite sequence of rewards along an execution through a discounted sum. For a discount factor $d > 1$, the discounted-sum payoff of an infinite reward sequence $r_0 r_1 \ldots$ is $\mathsf{DS}_d(r_0 r_1 \ldots) := \sum_{i=0}^{\infty} d^{-i} r_i$.

Representing numerical properties of discounted-sum payoffs with finite-state automata allows synthesis and verification problems involving such properties to be handled using efficient automata-theoretic techniques. A notable example is the development of the *comparator automaton* (Bansal, Chaudhuri, and Vardi 2018, 2022). A comparator automaton reads an infinite reward sequence w and accepts it iff $\mathsf{DS}_d(w) \bowtie t$, for a fixed rational threshold t and a comparison relation $\bowtie$ such as $>$ or $\leq$. Comparator automata have since been used for synthesis for *satisficing* (synthesize a "good enough" policy rather than an optimal one (Simon 1956)) objectives in two-player discounted-sum games (Bansal, Chatterjee, and Vardi 2021), synthesis that combines discounted-sum payoffs with temporal specifications (Bansal et al. 2022), and Nash-equilibrium realizability under discounted-sum payoffs in deterministic multi-agent systems (Rajasekaran, Bansal, and Vardi 2023).

In many settings, however, comparing the payoff with a single threshold does not provide enough information. For example, a decision maker may assign different utilities to losses, moderate gains, and large gains, or may wish to simultaneously track the payoff up to a prescribed accuracy.

In this paper, we introduce the *categorizer automaton*, a generalization of comparator automaton. Instead of reasoning about a single threshold like the comparator automaton, the categorizer automaton simultaneously considers several thresholds that divide the real line $\mathbb{R}$ into a series of disjoint intervals $I_1, \ldots, I_n$. The problem then becomes *categorizing* an infinite reward sequence by the interval that its discounted sum lies in, as opposed to *comparing* its discounted sum to a single threshold. That is, a categorizer automaton $\mathcal{A}$ reads an infinite reward sequence and identifies the unique interval containing its discounted sum. It does this based on which states repeat infinitely often along the run, a property that is captured by the automata-theoretic Büchi condition.

The comparator automaton is the special case of $n = 2$ intervals. A single boundary point splits the real line in two, and the automaton computes one bit about the payoff: whether $\mathsf{DS}_d(w)$ lies above or below the threshold. The use of finer partitions reveals more information about the payoff. In particular, because the discounted-sum payoff is bounded when the rewards are bounded, for every precision $\varepsilon > 0$ the bounded range of possible discounted sums can be divided into finitely many intervals of length at most ε (with the rest of $\mathbb{R}$ covered by two unbounded intervals) so that the resulting categorizer automaton identifies an interval of width at most ε containing $\mathsf{DS}_d(w)$.

A natural way to build a categorizer automaton is to take one comparator automaton per boundary point and construct their cross-product. This works, but the state space grows exponentially with the number n of intervals. This is prohibitively large when n is part of the input, as in the approx-

imation regime above, where reaching precision ε forces a number of intervals that grows as ε shrinks.

Our main technical contribution is a construction of the categorizer automaton that has a state space that is linear in the number n of intervals. As in the comparator automata setting, we assume that rewards are bounded integers, the boundary points are rational, and the discount factor d is an integer. Under these assumptions, for every finite partition of $\mathbb{R}$ into intervals, our construction yields a categorizer automaton whose state space is linear in n and polynomial in the numeric value of the largest reward and the largest denominator among the boundary points when written in lowest terms. The main idea behind this construction is that the per-boundary comparisons are highly dependent on one another, rather than independent as a cross-product construction via comparator automata would suggest. Exploiting this dependence is what allows one to avoid the exponential blowup.

Applications A categorizer automaton can be composed with an arbitrary finite-state model whose executions generate bounded reward sequences, including transition systems, games, Markov decision processes (MDPs), and stochastic games. In the resulting composition, the interval containing the discounted-sum payoff is represented by a Büchi condition. Thus, via categorizer automata, objectives depending on the exact payoff interval, an approximate payoff value, or both, can be reduced to automata-theoretic objectives.

We apply categorizer automata to MDPs, where they allow one to synthesize policies that maximize the expected utility of a discounted-sum payoff under utility functions that may be *discontinuous*. Utility functions provide a standard way to express how a decision maker values different outcomes, allowing the model to reflect attitudes toward risk rather than only the payoff (Mas-Colell et al. 1995). Abrupt changes in utility (discontinuities) arise naturally when meeting a target or crossing a safety threshold. Using categorizer automata together with existing automata-theoretic techniques (Courcoubetis and Yannakakis 1998; Etessami et al. 2008), we obtain, for integer discount factors, an exact algorithm for piecewise-constant utilities with finitely many rational discontinuities. The algorithm computes the optimal expected utility and synthesizes a finite-memory policy attaining it. For utilities that are Lipschitz continuous on each of finitely many intervals but may be discontinuous at their boundaries, we build on the piecewise-constant result to compute the optimal value up to an additive error of ε and synthesize an ε-optimal policy. Both algorithms run in pseudo-polynomial time: their running times are polynomial in certain numerical parameter values, but not in the lengths of their binary encodings. This dependence is unlikely to be avoidable, as we also show that the synthesis problem is PSPACE-hard even for piecewise-constant utilities.

We remark that the previous applications of comparator automata discussed above have focused on non-probabilistic models. Our application to MDPs extends the literature to stochastic settings.

Related Work The classical discounted-sum payoff objective maximizes the expectation $\mathbb{E}^{\theta}_{\mathcal{M}}[\mathsf{DS}_d]$ over policies θ in an MDP $\mathcal{M}$. This corresponds to the linear utility $u(x) = x$ and can be solved using the Bellman equations (Puterman 1994). In general, for a nonlinear utility u, the objective $\mathbb{E}^{\theta}_{\mathcal{M}}[u(\mathsf{DS}_d)]$ no longer admits Bellman-style equations on the original state space of the MDP (Kadota, Kurano, and Yasuda 1998). Existing algorithmic approaches typically exploit special structure in the utility, as for exponential utility (Bäuerle and Rieder 2014), or augment the state space with accumulated reward and apply discretizations or finite-horizon approximations (Wu and Xu 2023). Such approximations naturally apply to continuous utilities: truncating the discounted sum changes the payoff by a vanishing amount, and uniform continuity over the bounded payoff range then bounds the resulting change in the utility.

This argument fails for discontinuous utilities. Near a discontinuity, an arbitrarily small change in the payoff can produce a large change in utility. The simplest example is a threshold utility, defined by $u(x) = 0$ for $x < v$ and $u(x) = 1$ for $x \geq v$, whose expected value is precisely the probability that the discounted-sum payoff is at least v. Threshold utilities have been studied by White (1993) and Wu and Lin (1999), who characterize Bellman-style equations on an extended, possibly *infinite*, state space and investigate the conditions under which optimal policies exist. From an algorithmic perspective, they have been studied in the finite-horizon setting (Xu and Mannor 2011). For the infinite-horizon setting, Randour, Raskin, and Sankur (2015) study percentile queries through an ε-gap formulation, which permits instances sufficiently close to the payoff threshold to remain unresolved. Thus, prior algorithms either restrict the horizon or relax the threshold comparison, whereas, for integer discount factors, our algorithm solves the exact problem.

2 Categorizer Automata

This section introduces *categorizer automata* for discounted-sum payoffs (Definition 1), characterizes when categorizer automata exist (Proposition 1) and for the cases when they exist, establishes a size bound (Theorem 1).

2.1 Preliminaries

Words Given a finite *alphabet* Σ of symbols, a *word* (resp. *finite word*) w is an infinite (resp. finite) sequence of symbols. We denote the length of a finite word as $|w|$, and refer to the unique finite word of length 0 as ε. For a word (resp. finite word) w and index $1 \leq i$ (resp. $1 \leq i \leq |w|$) we denote: $w[i]$ as the i-th symbol of w, $w[1..i]$ as the length i prefix of w and $w[i..]$ as the suffix starting at position i. For a finite word w and a finite (resp. infinite) word w', we denote $w \cdot w'$ as the finite (resp. infinite) word formed by concatenating w and w'. We write Σ^{ω} (resp. Σ^{*}) for the set of all words (resp. finite words).

Transition System A *(deterministic) transition system* is a tuple $\mathcal{T} = (Q, \Sigma, q_{\text{init}}, \delta)$, where Q is a finite set of *states*, Σ is a finite alphabet, $q_{\text{init}} \in Q$ is the *initial state*, and $\delta : Q \times \Sigma \to Q$ is a *transition function*. The *run* of $\mathcal{T}$ on a word $w \in \Sigma^{\omega}$ is the sequence of states $\rho = \rho[1]\rho[2]\cdots$ with $\rho[1] = q_{\text{init}}$ and $\rho[k{+}1] = \delta(\rho[k], w[k])$ for all $k \geq 1$. We denote $\mathit{Inf}(\rho)$ for the set of states that occur infinitely often in ρ. Given transition systems $\mathcal{T}_i = (Q_i, \Sigma, q^i_{\text{init}}, \delta_i)$,

for $1 \leq i \leq n$, over the same alphabet Σ, their *cross-product* is the transition system whose states are tuples $(q_1, \ldots, q_n) \in Q_1 \times \cdots \times Q_n$, whose initial state is $(q_{\text{init}}^1, \ldots, q_{\text{init}}^n)$, and whose transition on a symbol σ maps $(q_1, \ldots, q_n)$ to $(\delta_1(q_1, \sigma), \ldots, \delta_n(q_n, \sigma))$.

Büchi Acceptance A *Büchi acceptance condition* is defined by a set $F \subseteq Q$ of states. A run ρ *satisfies* the Büchi acceptance condition F if $Inf(\rho) \cap F \neq \emptyset$, i.e. if at least one state of F occurs infinitely often in the run ρ. A *(deterministic) Büchi automaton* is a pair $\mathcal{A} = (\mathcal{T}, F)$ of a transition system and a Büchi acceptance. Its *language*, called the language recognized by $\mathcal{A}$, is the set of words $w \in \Sigma^\omega$ whose run on $\mathcal{T}$ satisfies F. A language (set of words) is *ω-regular* if it is recognized by some (possibly nondeterministic) Büchi automaton. Deterministic Büchi automata are effectively closed under intersection: given deterministic Büchi automata with states Q_1, Q_2 recognizing languages L_1, L_2, one can construct a deterministic Büchi automaton with $2|Q_1||Q_2|$ states recognizing $L_1 \cap L_2$ (Baier and Katoen 2008).

2.2 Discounted-sum and Categorizer Automaton

Discounted-Sum Given an integer *alphabet bound* $\mu \geq 1$, we denote Σ_μ as the finite alphabet consisting of all the integers from $-\mu$ to $+\mu$. For discount factor $d > 1$, the discounted sum of a word $w \in \Sigma_\mu^\omega$ (resp. $w \in \Sigma_\mu^*$) is $\mathsf{DS}_d(w) = \sum_{i=1}^{\infty} w[i]/d^{i-1}$ (resp. $\mathsf{DS}_d(w) = \sum_{i=1}^{|w|} w[i]/d^{i-1}$).

Binning of $\mathbb{R}$ A *binning* of $\mathbb{R}$ is a finite partition $B = \{I_1, \ldots, I_n\}$ of $\mathbb{R}$ into $n \geq 2$ nonempty, disjoint intervals, indexed so that every element of I_i is smaller than every element of I_{i+1}. The *endpoints* of B are the finite infimum and supremum of its intervals: the singleton $\{t\}$ has the one endpoint t, whereas $[t, t')$, $[t, t']$, (t, t'), and $(t, t']$ each have the two endpoints t and t'. The outer intervals I_1 and I_n are unbounded and so have one endpoint each. Writing $t_1 < \cdots < t_m$ for the endpoints of B in increasing order, we have $m \leq n - 1$, and each I_i is either the singleton $\{t_j\}$ or an interval with endpoints t_j and t_{j+1}, for some $0 \leq j \leq m$, where $t_0 = -\infty$ and $t_{m+1} = +\infty$. A binning is *rational* if all its endpoints are rational.

Definition 1 (Categorizer automaton)**.** *Let $\mu \geq 1$ be an integer alphabet bound, let $d > 1$ be a rational discount factor, and let $B = \{I_1, \ldots, I_n\}$ be a rational binning of $\mathbb{R}$. A* categorizer automaton *for (μ, d, B) is a tuple $\mathcal{A} = (\mathcal{T}, F_1, \ldots, F_n)$, where $\mathcal{T} = (Q, \Sigma_\mu, q_{\text{init}}, \delta)$ is a deterministic transition system and each $F_i \subseteq Q$ is a Büchi acceptance condition, such that, for every word $w \in \Sigma_\mu^\omega$ and every $1 \leq i \leq n$, it holds that $\mathsf{DS}_d(w) \in I_i$ if and only if the run of $\mathcal{T}$ on w satisfies F_i.*

In other words, a categorizer automaton for (μ, d, B) is one transition system equipped with n Büchi conditions, one per interval of B such that for each i, the language of the Büchi automaton $(\mathcal{T}, F_i)$ is $\{w \in \Sigma_\mu^\omega \mid \mathsf{DS}_d(w) \in I_i\}$.

Relation to Comparator Automata Categorizer automata generalize comparator automata (Bansal, Chaudhuri, and Vardi 2022). Given an integer alphabet bound μ, a rational discount factor $d > 1$, a rational threshold t, and a relation $\bowtie \in \{<, >, \leq, \geq, =, \neq\}$, a *comparator automaton* is a deterministic Büchi automaton that recognizes $\{w \in \Sigma_\mu^\omega \mid \mathsf{DS}_d(w) \bowtie t\}$. Bansal, Chaudhuri, and Vardi (2022) show that, for every relation $\bowtie$, such automata exist for every rational threshold t if and only if d is an integer.

We now use this result to characterize when categorizer automata exist for every rational binning. To see that the integrality of d is necessary, suppose that d is not an integer. The preceding result gives a rational threshold t for which no comparator automaton recognizes $\{w \in \Sigma_\mu^\omega \mid \mathsf{DS}_d(w) < t\}$. A categorizer automaton $(\mathcal{T}, F_1, F_2)$ for the rational binning $B_t = \{(-\infty, t), [t, +\infty)\}$ would make $(\mathcal{T}, F_1)$ precisely such a comparator automaton, yielding a contradiction.

Conversely, suppose that d is an integer, and fix a rational binning $B = \{I_1, \ldots, I_n\}$. We construct a categorizer automaton for (μ, d, B) using comparator automata. For each i, first construct a deterministic Büchi automaton $\mathcal{A}_i = (\mathcal{T}_i, G_i)$ that recognizes the language $L_i = \{w \in \Sigma_\mu^\omega \mid \mathsf{DS}_d(w) \in I_i\}$. Such an automaton can be constructed from comparator automata: depending on the form of I_i, the language L_i is either a comparator language or the intersection of two comparator languages. Taking the cross-product of $\mathcal{T}_1, \ldots, \mathcal{T}_n$ and letting F_i consist of the product states whose ith components belong to G_i yields a categorizer automaton for (μ, d, B). Hence, we obtain the following proposition.

Proposition 1 (Existence of categorizer automata)**.** *Let $\mu \geq 1$ be an integer, and let $d > 1$ be rational. Categorizer automata for (μ, d, B) exist for every rational binning B if and only if d is an integer.*

For non-integer d, the situation is more complex: it is an open problem to characterize the individual rational thresholds that admit comparator automata. For integer d, although the cross-product argument above establishes the existence of a categorizer automaton, it produces one whose size is exponential in the number of bins. In the next section we present a construction that avoids this blowup.

2.3 Construction That Avoids Exponential Blowup

In this section, we present a direct construction of a categorizer automaton whose size is linear in the number of bins.

Throughout, q_B denotes the largest denominator, in lowest terms, of endpoints of B.

Theorem 1. *Let $\mu \geq 1$ be an integer alphabet bound, let $d > 1$ be an integer discount factor, and let $B = \{I_1, \ldots, I_n\}$ be a rational binning. There exists a categorizer automaton for (μ, d, B) with $\mathcal{O}(n\mu q_B(1+\log_d(\mu q_B)))$ many states and it can be constructed in $\mathcal{O}(n\mu q_B(1 + \log_d(\mu q_B)))$ time.*

The bound is linear in the number of bins and polynomial in the numerical values of μ and q_B (thus, it is pseudo-polynomial in μ and q_B).

The remainder of this section proves Theorem 1 by an explicit construction. To that end, fix μ, d, and B as in Theorem 1. Let $t_1 < \cdots < t_m$ be the endpoints of B, set $t_0 = -\infty$ and $t_{m+1} = +\infty$, and write each finite endpoint in lowest terms as $t_j = p_j/q_j$ with $q_j > 0$ so that $q_B = \max_{1 \leq j \leq m} q_j$. Define $D = \mathsf{DS}_d(\mu\mu\cdots) = (\mu d)/(d-1)$, so that $\mathsf{DS}_d(w) \in [-D, D]$ for every $w \in \Sigma_\mu^\omega$.

High-Level Idea After every finite word u, the states of the categorizer automaton track a *gap value* for each endpoint. Intuitively, the gap value of an endpoint records the discounted sum that a continuation of u must have for the complete word to have a discounted sum exactly equal to that endpoint. Since every continuation has a discounted sum in $[-D, D]$, a gap value outside this interval determines on which side of the endpoint every continuation must lie. If the gap value lies below $-D$ (resp. above D), then, for every continuation w, the discounted sum of $u \cdot w$ lies above (resp. below) the endpoint. Otherwise, different continuations can still place the discounted sum below, at, or above the endpoint. Tracking how these gap values evolve therefore allows the automaton to locate the discounted sum relative to every endpoint and, consequently, to locate the bin that contains it.

Storing the tuple of all gap values explicitly would produce a state space exponential in the number of endpoints. The key observation behind our construction is that these values are not independent: all gap values can be recovered from the gap value of one endpoint and the length $|u|$. Their pairwise separations also grow exponentially with $|u|$. Consequently, after only logarithmically many symbols, at most one endpoint has a gap value in $[-D, D]$. From that point onward, its gap value and index suffice to determine the position of the discounted sum relative to every endpoint. The construction therefore encodes all gap values compactly by using one gap value, its endpoint index, and, during the short initial phase, the length $|u|$. This encoding gives a construction whose size is linear in the number of bins.

Gap Values Gap values are a standard tool for tracking a discounted sum relative to a single threshold (Boker and Henzinger 2014; Bansal, Chaudhuri, and Vardi 2022). For a finite word $u \in \Sigma_\mu^*$ and an endpoint t_j, the *gap value* of t_j after u is defined by the recurrence

$$g_j(\varepsilon) = t_j, \quad g_j(u \cdot a) = d \cdot \big(g_j(u) - a\big) \quad \text{for } a \in \Sigma_\mu. \tag{1}$$

An induction on $|u|$ gives the closed form

$$g_j(u) = d^{\,|u|}\big(t_j - \mathsf{DS}_d(u)\big). \tag{2}$$

Since $\mathsf{DS}_d(u \cdot w) = \mathsf{DS}_d(u) + d^{-|u|}\mathsf{DS}_d(w)$ for every $w \in \Sigma_\mu^\omega$, the gap value is the discounted sum a continuation of u must have in order to reach t_j: for $\bowtie \in \{<, =, >\}$ we have $\mathsf{DS}_d(u \cdot w) \bowtie t_j$ iff $\mathsf{DS}_d(w) \bowtie g_j(u)$. In particular, if $g_j(u) < -D$ (resp. $g_j(u) > D$) then $\mathsf{DS}_d(u \cdot w) > t_j$ (resp. $< t_j$) for every continuation w; we then say t_j is *resolved to* $\bot$ (resp. *to* $\top$) after u. Otherwise, i.e. if $g_j(u) \in [-D, D]$, we say t_j is *active* after u.

The following two propositions follow from the definition of gap values and are proved in the supplementary material.

Proposition 2. *Let $1 \le j \le m$ and let $w \in \Sigma_\mu^\omega$. Then $\mathsf{DS}_d(w) > t_j$ iff t_j is resolved to $\bot$ after some prefix of w; $\mathsf{DS}_d(w) < t_j$ iff t_j is resolved to $\top$ after some prefix of w; and $\mathsf{DS}_d(w) = t_j$ iff t_j is active after every prefix of w.*

For indices $1 \le j \le m$ let $G_j = \{x/q_j \mid x \in \mathbb{Z}\} \cap [-D, D]$. In the following proposition, P1 bounds the gaps of active endpoints, and P2 justifies the use of the term "resolved".

Proposition 3. *Let $u \in \Sigma_\mu^*$ and let $1 \le j \le m$. Then:*

P1 if t_j is active after u then $g_j(u) \in G_j$;

P2 if t_j is resolved to $\bot$ (resp. $\top$) after u then t_j is resolved to $\bot$ (resp. $\top$) after $u \cdot a$, for all $a \in \Sigma_\mu$.

We now describe how these facts can be used to construct a categorizer automata.

The Shared Structure of Gap Values By Prop. 2 and Prop. 3, it suffices for the transition system of the categorizer automaton for (μ, d, B) to track, after every finite word, the *status* of every endpoint: its gap value (from the finite set G_j) if active, and the value $\bot, \top$ it resolved to otherwise. However, storing the status of every endpoint explicitly requires $\prod_j (|G_j| + 2)$ states, which is still exponential in n. The next lemma shows that the statuses of m endpoints are not independent, which gives the ingredients for a transition system to track all of them simultaneously without incurring an exponential blowup.

Lemma 1. *Let $u \in \Sigma_\mu^*$ and let $K_0 = \big\lfloor \log_d(2Dq_B{}^2) \big\rfloor$. Then:*

P1 $g_i(u) - g_j(u) = d^{\,|u|}(t_i - t_j)$ for all $1 \le i, j \le m$;

P2 if $|u| > K_0$ then at most one endpoint is active after u;

P3 there exist ℓ, r with $0 \le \ell \le r \le m$ such that the endpoints $t_1, \dots, t_\ell$ are resolved to $\bot$ after u, the endpoints $t_{\ell+1}, \dots, t_r$ are active after u, and the endpoints $t_{r+1}, \dots, t_m$ are resolved to $\top$ after u.

Proof. P1 follows from Eq. 2. For P2, let t_i, t_j be the endpoints active after u i.e. $g_i(u), g_j(u) \in [-D, D]$. It follows that $|g_i(u) - g_j(u)| \le 2D$. Moreover, note that $|t_i - t_j| = |p_i q_j - p_j q_i|/q_i q_j \ge 1/q_i q_j \ge 1/{q_B}^2$. Substituting these two inequalities in P1 of the lemma, we get $d^{\,|u|}/{q_B}^2 \le 2D$. Rearranging and taking the logarithm gives $|u| \le K_0$. For P3, since $t_1 < \cdots < t_m$, by Eq. 2, $g_1(u) < g_2(u) < \cdots < g_m(u)$, hence if t_i is resolved to $\bot$ i.e. $g_i(u) < -D$ then t_j for all $j < i$ is also resolved to $\bot$. Similarly for $\top$. □

The Construction Formally, the categorizer automaton for (μ, d, B) is $\mathcal{A} = (\mathcal{T}, F_1, \dots, F_n)$. We encode the status of every endpoint compactly, using a representation according to the number of endpoints active after u. Informally, after reading a finite word u, the state of transition system $\mathcal{T}$ records the status of every endpoint, using an encoding that depends on how many endpoints remain active after u. Accordingly, the state of $\mathcal{T}$ takes one of three forms: **(a)** if two or more endpoints are active, the state is a triple $(j, g_j(u), k)$, where j is the index of smallest active endpoint and $k = |u|$. This triple encodes the status of every endpoint: by P1 of Lemma 1, the gap value of every t_i can be recovered, which in turn determines the resolved endpoints and their status; **(b)** If exactly one endpoint is active, the state is a pair $(j, g_j(u))$, where j is the index of that endpoint. By P3 of Lemma 1, $t_1, \dots, t_{j-1}$ are resolved to $\bot$ and $t_{j+1}, \dots, t_m$ to $\top$; **(c)** If no endpoint is active, the state is an integer ℓ, where $t_1, \dots, t_\ell$ are resolved to $\bot$ and $t_{\ell+1}, \dots, t_m$ to $\top$.

Transition System Formally, $\mathcal{T} = (Q, \Sigma_\mu, q_{\text{init}}, \delta)$, where $Q = Q_a \uplus Q_b \uplus Q_c$ is the union of the states of forms (a), (b), and (c), respectively, with

$$Q_a = \{\, (j, g, k) \mid 1 \le j \le m,\ g \in G_j,\ 0 \le k \le K_0 \,\},$$
$$Q_b = \{\, (j, g) \mid 1 \le j \le m,\ g \in G_j \,\}, \quad Q_c = \{0, \ldots, m\}.$$

We now describe the initial state q_{init} and transition function δ (the formal definition is given in the supplementary material). The initial state $q_{\text{init}} \in Q$ is determined by the status of the initial gap values, which by (1) coincide with the endpoints. On input $a \in \Sigma_\mu$, the transition under δ can be viewed as follows. First, decode the current state into the status of every endpoint. Second, update these statuses: resolved endpoints are left unchanged (P2 of Prop. 3), whereas active endpoints are updated using their new gap values, computed as $g'_j = d(g_j - a)$, as in (1). Finally, re-encode the resulting status as a state of Q, choosing the form according to the number of endpoints that remain active.

Acceptance Conditions Along every run ρ of $\mathcal{T}$, resolved endpoints stay resolved (P2 of Prop. 3) and eventually all but at most one of the endpoints become resolved. Hence, for a run ρ of $\mathcal{T}$, there exist $x \ge 0$ such that either (i) for all $y > x$ we have $\rho[y] = (j, g)$ for some $1 \le j \le m$ and $g \in G_j$, or (ii) for all $y > x$ we have $\rho[y] = j$ for some $0 \le j \le m$. By Prop. 2, it follows that the run ρ of $\mathcal{T}$ on $w \in \Sigma_\mu^\omega$ visits $\{(j, g) \mid g \in G_j\} \subseteq Q_b$ infinitely often iff $\mathsf{DS}_d(w) = t_j$, and visits the state $j \in Q_c$ infinitely often iff $\mathsf{DS}_d(w) \in (t_j, t_{j+1})$. Thus

$$F_i = \{(j, g) \in Q_b \mid t_j \in I_i\} \cup \{j \in Q_c \mid (t_j, t_{j+1}) \subseteq I_i\},$$

ensures that the run of $\mathcal{T}$ on w satisfies F_i iff $\mathsf{DS}_d(w) \in I_i$.

Size We bound $|Q| = |Q_a| + |Q_b| + |Q_c|$. Since $|G_j| = 2\lfloor D\, q_j \rfloor + 1 \le 2\lfloor D\, q_B \rfloor + 1$ for every j, and $m \le n - 1$:

$$|Q_a| \le n\,(2\lfloor D\, q_B \rfloor + 1)\,(K_0 + 1),$$
$$|Q_b| \le n\,(2\lfloor D\, q_B \rfloor + 1), \text{ and } |Q_c| \le n.$$

Thus $|Q| = \mathcal{O}(nDq_B(1 + K_0))$. Since $d \ge 2$, $D = (\mu d)/(d - 1) \le 2\mu$ and hence $D = \mathcal{O}(\mu)$. Moreover, $K_0 = \mathcal{O}(1 + \log_d(\mu\, q_B))$. Substituting these bounds gives $|Q| = \mathcal{O}(n\mu q_B(1 + \log_d(\mu q_B)))$. This proves Theorem 1: the above bound gives the size while the correctness follows from the discussion above.

3 Application: Policy Synthesis for Maximizing Expected Utility in MDPs

In this section we develop the application of categorizer automata to the problem of maximizing expected utility in Markov decision process (MDP) with discounted-sum payoffs. We provide pseudo-polynomial time algorithms and a PSPACE-hard lower bound for the synthesis problem under a broad class of utility functions.

3.1 Preliminaries

We follow the standard terminology of (Puterman 1994; Baier and Katoen 2008).

MDP A (discounted reward) *Markov decision process* (MDP) is a tuple $\mathcal{M} = (S, Act, s_{\text{init}}, P, r, d)$, where S is a finite set of states, Act is a finite set of actions, $s_{\text{init}} \in S$ is the initial state, $P : S \times Act \times S \to [0, 1] \cap \mathbb{Q}$ is a transition function, $r : S \times Act \to \mathbb{Z}$ is a reward function, and $d > 1$ is an integer discount factor. For every state s and action a we require $\sum_{s' \in S} P(s, a, s') \in \{0, 1\}$. When this sum equals 1, we say that a is *enabled* at s, and assume that at least one action is enabled at every state. We denote $\mu_{\mathcal{M}}$ as the *reward bound* $\max_{s \in S,\, a \in Act} |r(s, a)|$ of MDP $\mathcal{M}$. Let $D_{\mathcal{M}} = \mu_{\mathcal{M}} d/(d - 1)$. We write $|\mathcal{M}|$ for the size of the representation of $\mathcal{M}$: states, actions, and nonzero transitions are counted explicitly, while transition probabilities, rewards, and the discount factor are encoded in binary.

Plays and Policies A *play* is an infinite sequence $\pi = s_0\, a_0\, s_1\, a_1 \cdots$ with $s_0 = s_{\text{init}}$ and $P(s_k, a_k, s_{k+1}) > 0$ for every $k \ge 0$. A *history* is a finite prefix of a play ending in a state. Writing $H_{\mathcal{M}}$ for the set of all histories, a *policy* is a function $\theta : H_{\mathcal{M}} \times Act \to [0, 1]$ such that $\sum_{a \in Act} \theta(h, a) = 1$ for every history h, and $\theta(h, a) = 0$ whenever a is not enabled at the last state of history h. A policy is finite-memory if it can be implemented by a finite-state machine (Baier and Katoen 2008, Def. 10.97). We write $\Theta_{\mathcal{M}}$ for the set of all policies. A policy θ induces the standard probability measure $\Pr^\theta_{\mathcal{M}}$ on the set of plays, equipped with the σ-algebra generated by the cylinder sets of histories.

Discounted-Sum Payoff and Expected Utility Every play π determines the *reward word* $r(s_0, a_0)\, r(s_1, a_1) \cdots \in \Sigma_{\mu_{\mathcal{M}}}^\omega$. We denote $\mathsf{DS}_d(\pi)$ as the discounted sum of its reward word and refer to it as the *discounted-sum payoff of* π. The map $\pi \mapsto \mathsf{DS}_d(\pi)$ is measurable, so DS_d is a random variable under $\Pr^\theta_{\mathcal{M}}$ for every policy θ. A utility function $u : \mathbb{R} \to \mathbb{R}$ assigns a utility value to each discounted-sum payoff. We make the standard technical assumptions that u is Borel measurable and bounded on $[-D_{\mathcal{M}}, D_{\mathcal{M}}]$. Since every discounted-sum payoff generated by $\mathcal{M}$ lies in $[-D_{\mathcal{M}}, D_{\mathcal{M}}]$, $u(\mathsf{DS}_d)$ is a bounded random variable under $\Pr^\theta_{\mathcal{M}}$ for every policy θ. We denote $V^\theta_u(\mathcal{M})$ as the expectation $\mathbb{E}^\theta_{\mathcal{M}}[u(\mathsf{DS}_d)]$, and refer to it as the *expected utility* of θ. We denote $V^*_u(\mathcal{M})$ as the supremum of $V^\theta_u(\mathcal{M})$, over all policies $\theta \in \Theta_{\mathcal{M}}$. A policy θ is called *optimal* if $V^\theta_u(\mathcal{M}) = V^*_u(\mathcal{M})$, and *$\varepsilon$-optimal* if $V^\theta_u(\mathcal{M}) \ge V^*_u(\mathcal{M}) - \varepsilon$.

Büchi Events Let $Z \subseteq S$ be a set of states. We denote $\mathsf{Büchi}(Z)$ to be the event consisting of the plays that visit Z infinitely often.

3.2 Problem Statement and Overview

In this subsection, we formalize the synthesis problem, state the results, and give high level ideas used to establish them.

Piecewise-Lipschitz functions Given a rational binning $B = \{I_1, \ldots, I_n\}$, a function $u \colon \mathbb{R} \to \mathbb{R}$ is *piecewise-Lipschitz* on B with Lipschitz constant $L \ge 0$ if $|u(x) - u(y)| \le L\,|x - y|$ for every i and all $x, y \in I_i$. In other words, within each interval, a small change in the payoff cannot cause a disproportionately large change in its utility.

However, since no such restriction is imposed across different intervals, u may be discontinuous at their endpoints. The special case $L = 0$ is the *piecewise-constant* one, in which u takes a single value $u_i \in \mathbb{Q}$ on each I_i.

Synthesis Problem Instance A synthesis problem instance consists of an MDP $\mathcal{M}$, a rational binning $B = \{I_1, \ldots, I_n\}$, and a utility u that is piecewise-Lipschitz on B with a given constant $L \geq 0$.

For $L = 0$ the utility is piecewise-constant and we assume it is given explicitly, as the list of its values $u_1, \ldots, u_n \in \mathbb{Q}$ on the intervals of B. For $L > 0$ the problem instance additionally specifies a rational precision $\varepsilon > 0$, and the utility is presented by an *evaluation oracle* $\hat{u}$: on rational inputs x and $\eta > 0$ it returns a rational $\widehat{u}(x, \eta)$ with $|\widehat{u}(x, \eta) - u(x)| \leq \eta$, in time and with output length polynomial in the bit-length of x and in the numerical value $1/\eta$. Here and for the rest of the section, the bit-length of a rational number is the sum of the bit-lengths of its numerator and denominator when written in lowest terms.

For $L = 0$, the task is to compute $V_u^*(\mathcal{M})$ exactly together with an optimal policy. For $L > 0$, the task is to compute an ε-accurate value and an ε-optimal policy.

Recall that $\mu_{\mathcal{M}}$ is the reward bound of MDP $\mathcal{M}$ and q_B is the largest denominator, in lowest terms, of endpoints of B.

Theorem 2 (Piecewise-constant utility)**.** *For every synthesis problem instance with $L = 0$, the optimal expected utility $V_u^*(\mathcal{M})$ is rational. Moreover, $V_u^*(\mathcal{M})$ and an optimal finite-memory policy attaining it, can be computed in time polynomial in $|\mathcal{M}|$, the number n of bins, and the numerical values $\mu_{\mathcal{M}}$ and q_B.*

Theorem 3 (Piecewise-Lipschitz utility)**.** *For every synthesis problem instance with $L > 0$, an ε-optimal finite-memory policy and a rational number $\widehat{V}$ satisfying $|\widehat{V} - V_u^*(\mathcal{M})| \leq \varepsilon$ can be computed in time polynomial in $|\mathcal{M}|$, the number n of bins, and the numerical values $\mu_{\mathcal{M}}$, q_B, $\lceil L \rceil$, and $\lceil 1/\varepsilon \rceil$.*

For both theorems, the running time additionally depends polynomially on the maximum bit-length of the bin endpoints and, as applicable, the values $u_1, \ldots, u_n$, L, and ε.

High-Level Idea We now give a high-level overview of the proofs of the two theorems. The full details are presented in the following subsections. Let S and d be the states and discount factor of MDP $\mathcal{M}$. For a piecewise-constant utility u, the expected utility for a policy θ decomposes as

$$V_u^\theta(\mathcal{M}) = \sum_{i=1}^{n} u_i \Pr_{\mathcal{M}}^\theta[\mathsf{DS}_d \in I_i]. \tag{3}$$

Taking the product of $\mathcal{M}$ with the categorizer automaton $\mathcal{A} = (\mathcal{T}, F_1, \ldots, F_n)$ for $(\mu_{\mathcal{M}}, d, B)$ turns each event $\mathsf{DS}_d \in I_i$ into a Büchi event $\mathsf{Büchi}(S \times F_i)$. The problem therefore becomes the optimization of a weighted sum of Büchi probabilities in a finite MDP, which can be solved using the techniques of Courcoubetis and Yannakakis (1998).

For a piecewise-Lipschitz utility, we refine the given binning and use the evaluation oracle to construct a piecewise-constant utility u' satisfying $|u(x) - u'(x)| \leq \varepsilon/2$ for $x \in [-D_{\mathcal{M}}, D_{\mathcal{M}}]$. An optimal value and policy for u' are then ε-optimal for u.

Lower Bound The algorithms of Theorems 2 and 3 are pseudo-polynomial: their running times depend on the numerical values of $\mu_{\mathcal{M}}$ and q_B, rather than only on their bit-lengths. We complement these upper bounds with a hardness result that already holds for three bins and $q_B = 1$.

The QSubsetSum problem asks, given natural numbers $k_1, \ldots, k_N, T$, with N even, whether $\exists x_1 \in \{0,1\} \forall x_2 \in \{0,1\} \cdots \exists x_{N-1} \in \{0,1\} \forall x_N \in \{0,1\} : \sum_{i=1}^{N} x_i k_i = T$. The problem is PSPACE-complete (Travers 2006, Lem. 4) and, as observed by Haase and Kiefer (2015, §5), can be represented by a layered MDP with one level per number: the agent chooses whether to take *reward* k_i at odd level i, while a fair coin makes the choice at even level i. The agent has a policy under which the total reward equals T almost surely exactly when the QSubsetSum instance is positive.

Lemma 2. *Deciding whether $V_u^*(\mathcal{M}) \geq 1$ for a piecewise-constant utility u is PSPACE-hard, already for discount factor $d = 2$ and a binning B with three bins and integer endpoints, i.e. $q_B = 1$.*

Proof sketch. Following the discount-balancing construction of Randour, Raskin, and Sankur (2015), in the layered MDP above set $d = 2$ and multiply each reward at level i by d^{i-1}. The discounted-sum payoff of any play is then exactly the total reward in the original MDP. Finally, take $B = \{(-\infty, T), \{T\}, (T, +\infty)\}$ and assign utility 1 to $\{T\}$ and utility 0 to the other two bins. Thus, $V_u^*(\mathcal{M}) = 1$ exactly when the QSubsetSum instance is positive. Full details are provided in the supplementary material. □

We remark that this strongly suggests that there is no algorithm constructing categorizer automata for (μ, d, B) that runs in polynomial time in the bit-length of alphabet bound μ.

3.3 Proof of Theorem 2

In this subsection we prove Thm. 2. Fix an instance consisting of an MDP $\mathcal{M} = (S, Act, s_{\text{init}}, P, r, d)$, a rational binning $B = \{I_1, \ldots, I_n\}$, and a piecewise-constant utility u on B with value u_i on bin I_i.

Let $\mathcal{A} = (\mathcal{T}, F_1, \ldots, F_n)$ be the categorizer automaton for $(\mu_{\mathcal{M}}, d, B)$, with $\mathcal{T} = (Q, \Sigma_{\mu_{\mathcal{M}}}, q_{\text{init}}, \delta)$.

Product with Categorizer Automaton The product $\mathcal{M} \otimes \mathcal{A}$ is an MDP that records the current states of both the MDP $\mathcal{M}$ and the categorizer automaton $\mathcal{A}$. It has state space $S \times Q$, action set Act, initial state $(s_{\text{init}}, q_{\text{init}})$, and transition probability $P^{\otimes}((s,q), a, (s', q')) = P(s, a, s')$ if $q' = \delta(q, r(s,a))$, and 0 otherwise.

Because the categorizer automaton $\mathcal{A}$ is deterministic, every play $\pi = s_0 a_0 s_1 a_1 \cdots$ of $\mathcal{M}$ has a unique corresponding play $\pi^{\otimes} = (s_0, q_0) a_0 (s_1, q_1) a_1 \ldots$ of $\mathcal{M} \otimes \mathcal{A}$, where $q_0 = q_{\text{init}}$ and $q_{k+1} = \delta(q_k, r(s_k, a_k))$ for every $k \geq 0$. Conversely, the projection of every play of $\mathcal{M} \otimes \mathcal{A}$ on its S-component yields a unique play of $\mathcal{M}$.

The same correspondence holds between finite histories of $\mathcal{M}$ and $\mathcal{M} \otimes \mathcal{A}$ and extends to their policies: given a policy θ on $\mathcal{M}$, define the policy $\theta^{\otimes}$ on $\mathcal{M} \otimes \mathcal{A}$ by $\theta^{\otimes}(h^{\otimes}, a) = \theta(h, a)$ for every pair of corresponding histories h and $h^{\otimes}$. Conversely, every policy on the product determines a policy

on $\mathcal{M}$. Corresponding histories use the same actions and transition probabilities, and hence corresponding measurable events have the same probability.

Reduction By construction, a play π of $\mathcal{M}$ satisfies $\mathsf{DS}_d(\pi) \in I_i$ iff the corresponding play $\pi^\otimes$ of $\mathcal{M} \otimes \mathcal{A}$ visits $S \times F_i$ infinitely often. With (3) this gives, for corresponding θ and $\theta^\otimes$, the expected utility $V_u^\theta(\mathcal{M})$ is equal to the weighted sum $\sum_{i=1}^n u_i \Pr^{\theta^\otimes}_{\mathcal{M}\otimes\mathcal{A}}[\mathsf{Büchi}(S \times F_i)]$ of Büchi probabilities. When all u_i are nonnegative, maximizing the right-hand side can be done using the following straightforward consequence of (Courcoubetis and Yannakakis 1998).

Proposition 4. *For a finite MDP $\mathcal{N}$ with state space Z, sets $Z_1, \ldots, Z_k \subseteq Z$, and weights $v_1, \ldots, v_k \in \mathbb{Q}_{\geq 0}$, the optimal value $\sup_{\theta\in\Theta_\mathcal{N}} \sum_{i=1}^k v_i \cdot \Pr^\theta_\mathcal{N}[\mathsf{Büchi}(Z_i)]$ is rational. Moreover, this value and a finite-memory policy attaining it can be computed in time polynomial in $|\mathcal{N}|$, k and the maximum bit-length of the weights.*

Indeed, in the corresponding theorem in (Courcoubetis and Yannakakis 1998, Thm. 5.1, § 7) the events $\mathsf{Büchi}(Z_i)$ are given by automata that must first be composed with the MDP $\mathcal{N}$, and the running time is polynomial in the size of that product. Here, each event is already a Büchi condition on the states of $\mathcal{N}$, so the composition step is not needed and the rest of their proof applies.

If some utility values are negative, we shift all values by a common constant before applying Prop. 4. Let $c = \max\{0, -\min_i u_i\}$ and define $\bar{u}(x) = u(x) + c$. Then $\bar{u}$ is piecewise-constant on B, with value $\bar{u}_i = u_i + c \geq 0$ on I_i. By taking expectations, for every policy θ, $V_{\bar{u}}^\theta(\mathcal{M}) = V_u^\theta(\mathcal{M}) + c$. Consequently, u and $\bar{u}$ have the same optimal policies, and $V_{\bar{u}}^*(\mathcal{M}) = V_u^*(\mathcal{M}) + c$.

Applying Prop. 4 to $\mathcal{M} \otimes \mathcal{A}$, with Büchi sets $S \times F_1, \ldots, S \times F_n$ and nonnegative rational weights $\bar{u}_1, \ldots, \bar{u}_n$, computes $V_{\bar{u}}^*(\mathcal{M})$ and an optimal finite-memory policy $\theta^\otimes$ on the product. Subtracting c from the computed value gives $V_u^*(\mathcal{M})$. Transferring the policy $\theta^\otimes$ to θ of $\mathcal{M}$ yields an optimal finite-memory policy for u: its memory maintains the current state of $\mathcal{A}$ together with the memory used by $\theta^\otimes$. The size of $\mathcal{M} \otimes \mathcal{A}$ is polynomial in $|\mathcal{M}|$ and $|\mathcal{A}|$. Thm. 1 and Prop. 4 therefore give the complexity bound of Thm. 2.

3.4 Proof of Theorem 3

In this subsection, we prove Thm. 3. Fix an instance consisting of an MDP $\mathcal{M} = (S, Act, s_{\text{init}}, P, r, d)$, a rational binning $B = \{I_1, \ldots, I_n\}$, a rational precision $\varepsilon > 0$, and a piecewise-Lipschitz utility u on B with Lipschitz constant $L > 0$, represented by an evaluation oracle $\widehat{u}$.

Piecewise-Constant Approximation Let $\hat{L} = \lceil L \rceil$, $E = \lceil 1/\varepsilon \rceil$, $\tau = 1/(4\hat{L}E)$, and $W = \lceil D_\mathcal{M}/\tau \rceil + 1$. Thus, $L \leq \hat{L}$, $1/E \leq \varepsilon$, and $[-D_\mathcal{M}, D_\mathcal{M}] \subseteq [-W\tau, W\tau]$.

Construct a binning $B' = \{J_1, \ldots, J_r\}$, using the endpoints of B and the grid points $k\tau$, for $k \in \{-W, \ldots, W\}$, such that every bin intersecting $[-D_\mathcal{M}, D_\mathcal{M}]$ has length at most τ and every bin J_i is contained in a bin of B.

For every bin J_i intersecting $[-D_\mathcal{M}, D_\mathcal{M}]$, choose a rational point $z_i \in J_i \cap [-D_\mathcal{M}, D_\mathcal{M}]$ whose bit-length is polynomial in the bit-lengths of the endpoints of J_i. Query the evaluation oracle at z_i with precision $\eta = 1/(4E)$, and let $u'_i = \widehat{u}(z_i, \eta)$. Define $u'(x) = u'_i$ for every $x \in J_i$. On bins disjoint from $[-D_\mathcal{M}, D_\mathcal{M}]$, define u' to be 0.

The resulting utility u' is piecewise-constant on B' and satisfies $|u(x) - u'(x)| \leq \varepsilon/2$ for every $x \in [-D_\mathcal{M}, D_\mathcal{M}]$. Indeed, let J_i be the bin containing x. Since $x, z_i \in J_i$, J_i has length at most τ, and J_i is contained in a bin of B, the Lipschitz condition gives $|u(x) - u(z_i)| \leq L\tau$. The oracle guarantee gives $|u(z_i) - u'_i| \leq \eta$. Therefore, $|u(x) - u'(x)| \leq |u(x) - u(z_i)| + |u(z_i) - u'_i| \leq L\tau + \eta \leq 1/(2E) \leq \varepsilon/2$.

Reduction and Complexity Taking expectations, for every policy θ of $\mathcal{M}$, $V_{u'}^\theta(\mathcal{M}) - \varepsilon/2 \leq V_u^\theta(\mathcal{M}) \leq V_{u'}^\theta(\mathcal{M}) + \varepsilon/2$. Thus, the optimal value $V_{u'}^*(\mathcal{M})$ satisfies $|V_u^*(\mathcal{M}) - V_{u'}^*(\mathcal{M})| \leq \varepsilon/2$, and an optimal policy θ^* for u' is ε-optimal for u (since $V_u^*(\mathcal{M}) \leq V_{u'}^{\theta^*}(\mathcal{M}) + \varepsilon/2 \leq V_u^{\theta^*}(\mathcal{M}) + \varepsilon$). Thm. 2 computes $V_{u'}^*(\mathcal{M})$ and such a policy θ^*.

Since $d \geq 2$, we have $D_\mathcal{M} \leq 2\mu_\mathcal{M}$, and hence $W = \mathcal{O}(\mu_\mathcal{M}\hat{L}E)$. Therefore, the number r of bins of B' is on the order of $n + \mathcal{O}(\mu_\mathcal{M}\hat{L}E)$. Moreover, since every new endpoint is of the form $k/(4\hat{L}E)$, the largest denominator $q_{B'}$ is at most $\max\{q_B, 4\hat{L}E\}$.

Constructing u' requires r oracle calls, each with precision $\eta = 1/(4E)$. By the choice of the query points and the oracle assumption, the total running time of these calls and the bit-lengths of their outputs are polynomial in r, the bit-lengths of the endpoints of B', and the numerical value E. Together with the bounds on r and $q_{B'}$ above, Thm. 2 gives the running-time bound claimed in Thm. 3.

4 Conclusion

We introduced categorizer automata and showed how to construct them in time and space linear (rather than exponential) in the number of bins. We then applied this construction to obtain algorithms for synthesizing policies that maximize the expected utility of the payoff for a broad class of (possibly) discontinuous utilities. We view this as one instance of a broader use of categorizer automata. A natural next step is to synthesize policies in MDPs that optimize *soft* quantitative preferences, expressed through discounted-sum payoffs, subject to *hard* qualitative requirements, expressed by LTL specifications. These two objectives are typically handled using different algorithmic techniques, see (Puterman 1994) and (Baier and Katoen 2008). A categorizer automaton bridges the mismatch by translating the quantitative payoff into ω-regular conditions.

Finally, our construction of categorizer automata, like the comparator automata before it, requires the discount factor to be an integer. One potential route to non-integer discount factors is an approximate categorizer automaton, in the spirit of Randour, Raskin, and Sankur (2015), that is required to identify the correct bin only when the payoff is at a distance greater than ε from every endpoint. This relaxation introduces only a vanishing utility error when the utility is continuous at the endpoints. At a discontinuity, however, it cannot recover the guarantees of Theorems 2 and 3, and new ideas are needed.

References

Baier, C.; and Katoen, J. 2008. *Principles of model checking*. MIT Press.

Bansal, S.; Chatterjee, K.; and Vardi, M. Y. 2021. On Satisficing in Quantitative Games. In Groote, J. F.; and Larsen, K. G., eds., *Tools and Algorithms for the Construction and Analysis of Systems TACAS 2021*, Lecture Notes in Computer Science, 20–37. Springer.

Bansal, S.; Chaudhuri, S.; and Vardi, M. Y. 2018. Comparator Automata in Quantitative Verification. In *FoSSaCS*, volume 10803 of *Lecture Notes in Computer Science*, 420–437. Springer.

Bansal, S.; Chaudhuri, S.; and Vardi, M. Y. 2022. Comparator automata in quantitative verification. *Logical Methods in Computer Science*, 18.

Bansal, S.; Kavraki, L.; Vardi, M.; and Wells, A. 2022. Synthesis from Satisficing and Temporal Goals. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36: 9679–9686.

Boker, U.; and Henzinger, T. A. 2014. Exact and Approximate Determinization of Discounted-Sum Automata. *Log. Methods Comput. Sci.*, 10(1).

Bäuerle, N.; and Rieder, U. 2014. More Risk-Sensitive Markov Decision Processes. *Mathematics of Operations Research*, 39(1): 105–120.

Courcoubetis, C.; and Yannakakis, M. 1998. Markov decision processes and regular events. *IEEE Trans. Autom. Control.*, 43(10): 1399–1418.

de Alfaro, L.; Henzinger, T. A.; and Majumdar, R. 2003. Discounting the Future in Systems Theory. In *ICALP*, volume 2719 of *Lecture Notes in Computer Science*, 1022–1037. Springer.

Etessami, K.; Kwiatkowska, M.; Vardi, M. Y.; and Yannakakis, M. 2008. Multi-Objective Model Checking of Markov Decision Processes. *Logical Methods in Computer Science*, Volume 4, Issue 4: 8.

Haase, C.; and Kiefer, S. 2015. The Odds of Staying on Budget. In *ICALP (2)*, volume 9135 of *Lecture Notes in Computer Science*, 234–246. Springer.

Kadota, Y.; Kurano, M.; and Yasuda, M. 1998. On the General Utility of Discounted Markov Decision Processes. *International Transactions in Operational Research*, 5(1): 27–34.

Mas-Colell, A.; Whinston, M. D.; Green, J. R.; et al. 1995. *Microeconomic theory*, volume 1. Oxford university press New York.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley.

Rajasekaran, S.; Bansal, S.; and Vardi, M. Y. 2023. Multi-Agent Systems with Quantitative Satisficing Goals. In *IJCAI*, 280–288. ijcai.org.

Randour, M.; Raskin, J.-F.; and Sankur, O. 2015. Percentile Queries in Multi-dimensional Markov Decision Processes. In Kroening, D.; and Păsăreanu, C. S., eds., *Computer Aided Verification*, 123–139. Cham: Springer International Publishing. ISBN 978-3-319-21690-4.

Simon, H. A. 1956. Rational choice and the structure of the environment. *Psychological Review*, 63(2): 129–138.

Travers, S. D. 2006. The complexity of membership problems for circuits over sets of integers. *Theor. Comput. Sci.*, 369(1-3): 211–229.

White, D. J. 1993. Minimizing a Threshold Probability in Discounted Markov Decision Processes. *Journal of Mathematical Analysis and Applications*, 173(2): 634–646.

Wu, C.; and Lin, Y. 1999. Minimizing Risk Models in Markov Decision Processes with Policies Depending on Target Values. *Journal of Mathematical Analysis and Applications*, 231(1): 47–67.

Wu, Z.; and Xu, R. 2023. Risk-sensitive Markov Decision Process and Learning under General Utility Functions. *CoRR*, abs/2311.13589.

Xu, H.; and Mannor, S. 2011. Probabilistic goal Markov decision processes. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, 2046–2052. Barcelona, Catalonia, Spain: AAAI Press. ISBN 978-1-57735-515-1.

A Supplementary Material

A.1 Categorizer Automata

Proof of Proposition 2. Fix j and w, and abbreviate $g^k = g_j(w[1..k])$ and $\sigma^k = \mathsf{DS}_d(w[k{+}1..]) \in [-D, D]$. Since $\mathsf{DS}_d(w) = \mathsf{DS}_d(w[1..k]) + d^{-k}\sigma^k$, the closed form (2) gives

$$t_j - \mathsf{DS}_d(w) \;=\; d^{-k}(g^k - \sigma^k) \qquad \text{for every } k \geq 0. \quad (4)$$

If $g^k < -D$ for some k, then $g^k - \sigma^k < 0$ because $\sigma^k \geq -D$, so $\mathsf{DS}_d(w) > t_j$ by (4); symmetrically, $g^k > D$ for some k implies $\mathsf{DS}_d(w) < t_j$. If $g^k \in [-D, D]$ for all k, then $|g^k - \sigma^k| \leq 2D$, so $|t_j - \mathsf{DS}_d(w)| \leq 2D\, d^{-k}$ for all k, and the right-hand side tends to 0 as k tends to infinity because $d > 1$; hence $\mathsf{DS}_d(w) = t_j$.

The three hypotheses above say that t_j is resolved to $\bot$ after some prefix of w, resolved to $\top$ after some prefix of w, and active after every prefix of w, respectively. They are exhaustive, since $g^k \notin [-D, D]$ means $g^k < -D$ or $g^k > D$, and pairwise disjoint: the third excludes the other two by definition, and the first two exclude each other because their conclusions $\mathsf{DS}_d(w) > t_j$ and $\mathsf{DS}_d(w) < t_j$ do. Since the three conclusions are likewise exhaustive and pairwise disjoint, each implication is in fact an equivalence. □

Proof of Proposition 3. P1. We claim that $g_j(u) \in \frac{1}{q_j}\mathbb{Z}$ for every $u \in \Sigma_\mu^*$. Indeed, by (2),

$$g_j(u) \;=\; \frac{d^{|u|}p_j}{q_j} - \sum_{i=1}^{|u|} u[i]\, d^{|u|-i+1},$$

where the sum is an integer because d is an integer and $|u| - i + 1 \geq 1$ for $1 \leq i \leq |u|$. If t_j is active after u then moreover $g_j(u) \in [-D, D]$, so $g_j(u) \in G_j$.

P2. Since $D - \mu = \mu/(d-1)$, the bound D is the gap update at the integer bound μ:

$$d\,(D - \mu) = D \qquad \text{and} \qquad d\,(-D + \mu) = -D. \quad (5)$$

Let $a \in \Sigma_\mu$, so $-\mu \le a \le \mu$. If t_j is resolved to $\bot$ after u, i.e. $g_j(u) < -D$, then $g_j(u) - a < -D + \mu$, and multiplying by $d > 0$ and using (5) gives

$$g_j(u \cdot a) \;=\; d\,(g_j(u) - a) \;<\; d\,(-D + \mu) \;=\; -D,$$

so t_j is resolved to $\bot$ after $u \cdot a$. Symmetrically, if $g_j(u) > D$ then $g_j(u) - a > D - \mu$ and $g_j(u \cdot a) > d\,(D - \mu) = D$, so t_j remains resolved to $\top$. □

Formal Definition of q_{init} Recall that by (1) we have $g_i(\varepsilon) = t_i$. Let $A_{\text{init}} = \{i \mid t_i \in [-D, D]\}$ and $\ell_{\text{init}} = \max\{i \mid t_i < -D\}$, where we use the convention $\max \emptyset = 0$. Accordingly,

$$q_{\text{init}} = \begin{cases} (j, t_j, 0),\ j = \min A_{\text{init}} & \text{if } |A_{\text{init}}| \ge 2, \\ (j, t_j) & \text{if } A_{\text{init}} = \{j\}, \\ \ell_{\text{init}} & \text{if } A_{\text{init}} = \emptyset. \end{cases}$$

Formal Definition of δ Let $s \in Q$ be a state and let $r \in \Sigma_\mu$ be an input symbol. The transition $\delta(s, r)$ is defined as follows:

- $\mathbf{s \in Q_a}$ Let $s = (j, g, k)$. We call s *consistent* if $g - d^k t_j \in \mathbb{Z}$. If s is not consistent, set $\delta(s, r) = s$.

  Every state in Q_a reachable from q_{init} is consistent: if it is reached after a word u of length k, then by (2),

  $$g - d^k t_j = -d^k \mathsf{DS}_d(u) \in \mathbb{Z}.$$

  Hence, the self-loops added above do not affect any run from q_{init}.

  Suppose now that s is consistent. First reconstruct $\widehat{g}_h = g + d^k(t_h - t_j)$ for $1 \le h \le m$, as in P1 of Lemma 1, and then apply update (1) to obtain $\widehat{g}'_h = d(\widehat{g}_h - r)$. Let

  $$A' = \{i \mid \widehat{g}'_i \in [-D, D]\}, \quad \ell' = \max\{i \mid \widehat{g}'_i < -D\}.$$

  The successor $s' = \delta(s, r)$ is determined by $|A'|$:
  - if $|A'| \ge 2$, then $s' = (j', \widehat{g}'_{j'}, k{+}1)$ with $j' = \min A'$;
  - if $A' = \{h\}$, then $s' = (h, \widehat{g}'_h)$;
  - if $A' = \emptyset$, then $s' = \ell'$.
- $\mathbf{s \in Q_b}$ Let $s = (j, g)$, let $g' = d\,(g - r)$ as in (1), and set

  $$\delta(s, r) = \begin{cases} (j, g') & \text{if } g' \in [-D, D], \\ j & \text{if } g' < -D, \\ j - 1 & \text{if } g' > D. \end{cases}$$
- $\mathbf{s \in Q_c}$ $\delta(s, r) = s$.

A.2 Lower Bound

Proof of Lemma 2. We give the details of the reduction from QSubsetSum described in the proof sketch. Fix an instance $k_1, \ldots, k_N, T$, where N is even. We construct the MDP $\mathcal{M}$ shown in Figure 1 and set its discount factor to $d = 2$. For every odd i, the MDP has one state c_i, while for every even i, it has two states c_i^0 and c_i^1. It also has an absorbing state s_{sink}, and its initial state is c_1. Thus, $\mathcal{M}$ has $3N/2 + 1$ states.

At an odd-level state c_i, the actions take and skip are enabled. Under either action, the next state is c_{i+1}^0 or c_{i+1}^1, each with probability $1/2$. The rewards of these actions are

$$r(c_i, \mathsf{take}) = k_i d^{i-1} \qquad \text{and} \qquad r(c_i, \mathsf{skip}) = 0.$$

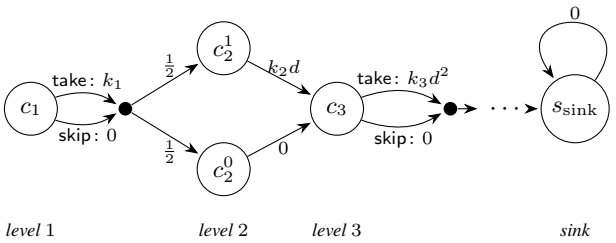

Figure 1: The layered MDP $\mathcal{M}$ used in the lower bound. At odd levels, edge labels give the action and its reward. At even levels and at s_{sink}, only one action is enabled, so only its reward is shown. Filled dots denote fair probabilistic branching. The repeated levels after level 3 are omitted.

At an even-level state c_i^b, where $b \in \{0, 1\}$, exactly one action is enabled, and its reward is $b\,k_i d^{i-1}$. For $i < N$, this action leads with probability one to c_{i+1}; from level N, it leads to s_{sink}. The only action enabled at s_{sink} has reward 0 and returns to s_{sink}.

For a play π, let $x_i(\pi) \in \{0, 1\}$ indicate whether k_i is taken. At an odd level, this is determined by whether the policy chooses take or skip. At an even level, it is determined by whether the play enters c_i^1 or c_i^0. The reward associated with level i is collected at step $i - 1$. Therefore,

$$\mathsf{DS}_d(\pi) = \sum_{i=1}^{N} \frac{x_i(\pi) k_i d^{i-1}}{d^{i-1}} = \sum_{i=1}^{N} x_i(\pi) k_i.$$

Thus, the scaling by d^{i-1} exactly cancels the discounting. The rewards may be exponentially large in value, but multiplying k_i by 2^{i-1} increases its binary encoding length by only $i - 1$. Hence, $\mathcal{M}$ can be constructed in time polynomial in the size of the QSubsetSum instance.

Consider the three-bin binning $B = \{(-\infty, T), \{T\}, (T, +\infty)\}$ [1] and the piecewise-constant utility u that is 1 on $\{T\}$ and 0 on the other two bins. All endpoints are integers, so $q_B = 1$. Viewing each x_i as a random variable on plays, for every policy θ we have

$$V_u^\theta(\mathcal{M}) = \Pr_{\mathcal{M}}^{\theta}\left[\sum_{i=1}^{N} x_i k_i = T\right].$$

Since u takes values in $\{0, 1\}$, we have $V_u^*(\mathcal{M}) \le 1$. Moreover, only the first N steps affect the payoff, so an optimal policy exists. It follows that $V_u^*(\mathcal{M}) \ge 1$ exactly when some policy makes the sum equal to T almost surely. By (Haase and Kiefer 2015) such a policy exists exactly when the QSubsetSum instance is positive.

The reduction is polynomial and uses the fixed discount factor $d = 2$, three bins, and integer endpoints. The claimed PSPACE-hardness follows. □

[1] Binning $B = \{(-\infty, T - 1], (T - 1, T + 1), [T + 1, +\infty)\}$ also works as the discounted-sum payoff is always an integer.