Journal of Automated Reasoning (2020) 64:1483–1522
https://doi.org/10.1007/s10817-020-09574-9

Multi-cost Bounded Tradeoff Analysis in MDP

Arnd Hartmanns[1] · **Sebastian Junges**[2] · **Joost-Pieter Katoen**[1,2] · **Tim Quatmann**[2]

Received: 23 June 2020 / Accepted: 2 July 2020 / Published online: 28 July 2020

Abstract
We provide a memory-efficient algorithm for multi-objective model checking problems on Markov decision processes (MDPs) with multiple cost structures. The key problem at hand is to check whether there exists a scheduler for a given MDP such that all objectives over cost vectors are fulfilled. We cover multi-objective reachability and expected cost objectives, and combinations thereof. We further transfer approaches for computing quantiles over single cost bounds to the multi-cost case and highlight the ensuing challenges. An empirical evaluation shows the scalability of our new approach both in terms of memory consumption and runtime. We discuss the need for more detailed visual presentations of results beyond Pareto curves and present a first visualisation approach that exploits all the available information from the algorithm to support decision makers.

Keywords Markov decision process · Multi-objective verification · Pareto-optimal strategies · Cost-bounded reachability · Expected rewards · Probabilistic model checking

1 Introduction

Markov decision processes [46] (MDPs) with *rewards* or *costs* are a popular model to describe planning problems under uncertainty. Planning algorithms aim to find strategies which perform well (or even optimally) for a given objective. These algorithms typically assume that

The authors are listed in alphabetical order. This work was supported by DFG RTG 2236 "UnRAVeL" and NWO VENI Grant 639.021.754.

✉ Arnd Hartmanns
a.hartmanns@utwente.nl

Sebastian Junges
sebastian.junges@cs.rwth-aachen.de

Joost-Pieter Katoen
j.p.katoen@utwente.nl; katoen@cs.rwth-aachen.de

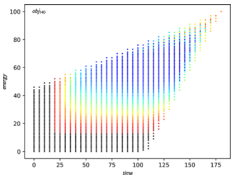
Tim Quatmann
tim.quatmann@cs.rwth-aachen.de

[1] University of Twente, Enschede, The Netherlands

[2] RWTH Aachen, Aachen, Germany

task	time	energy consumption	scientific value	success prob.
1	high	{low, medium}	medium	1/2
2	low	{medium, high}	medium	3/5
3	low	low	low	4/5
4	high	high	high	1/10

(a) Different possible tasks for a Mars rover



(b) Tradeoffs in costs

Fig. 1 Science on Mars: planning under several resource constraints

a goal is reached eventually (with probability one) and optimise the *expected* reward or cost to reach that goal [46,53]. This assumption however is unrealistic in many scenarios, e.g. due to insufficient resources or the possibility of attempted actions failing. Furthermore, the resulting optimal schedulers often admit single runs which perform far below the user's expectation. Such deviations to the expected value are unsuitable in many scenarios with high stakes. Examples range from deliveries reaching an airport after the plane's departure to more serious scenarios in e.g. wildfire management [56]. In particular, many practical scenarios call for minimising the probability to run out of resources before reaching the goal: while it is *beneficial* for a plane to reach its destination with low *expected* fuel consumption, it is *essential* to reach its destination with the *fixed* available amount of fuel.

Schedulers that optimise solely for the probability to reach a goal are mostly very expensive. Even in the presence of just a single cost structure, decision makers have to trade the success probability against the costs. This tradeoff makes many planning problems inherently multi-objective [12,17]. In particular, safety properties cannot be averaged out by good performance [22]. Planning scenarios in various application areas [51] have different resource constraints. Typical examples are energy consumption and time [11], or optimal expected revenue and time [42] in robotics, and monetary cost and available capacity in logistics [17].

Example 1 Consider a simplified (discretised) version of the Mars rover task scheduling problem [11]. We want to plan a variety of experiments for a day on Mars. The experiments vary in their success probability, time, energy consumption, and scientific value upon success. The time, energy consumption, and scientific value are uncertain and modelled by probability distributions, cf. Figure 1a. Each day, the rover can perform multiple experiments until it runs out of time or out of energy. The objective is to achieve a *minimum* of daily scientific progress while keeping the risk of exceeding the time or energy limits low. As the rover is expected to work for a longer period, we *prefer* a high *expected* scientific value.

This article focuses on (i) multi-objective multi-cost bounded reachability queries as well as (ii) multi-cost quantiles on MDPs. We take as input an MDP with multiple cost structures (e.g. energy, utility, and time).

The bounded reachability problem is specified as multiple objectives of the form "maximise/minimise the probability to reach a state in G_i such that the cumulative cost for the i-th cost structure is below/above a cost limit b_i". This multi-objective variant of cost-bounded reachability in MDPs is PSPACE-hard [49]. The focus of this article is on the practical side: we aim at finding a practically efficient algorithm to obtain (an approximation of) the Pareto-optimal points. To accomplish this, we adapt and generalise recent approaches for the single-objective case [28,37] towards the multi-objective setting. The basic idea of [28,37] is to *implicitly* unfold the MDP along cost epochs, and exploit the regularities of the epoch MDP. PRISM [39] and the MODEST TOOLSET [31] have been updated with such methods for the single-objective case and significantly outperform the traditional explicit unfolding

approach of [1,44]. This article presents an algorithm that lifts this principle to multiple cost objectives and determines approximation errors when using value iteration. We also sketch extensions to expected accumulated cost objectives.

The problem of computing quantiles [2,37,52,57] is essentially the inversion of the bounded reachability problem: a quantile query has the form "what are the cumulative cost limits b_i such that the maximum/minimum probability to reach a state in G_i with the accumulated cost for the i-th cost structure being below/above b_i is less/greater than a fixed probability threshold p." Such an inversion is natural: Instead of asking how likely it is to arrive at the planned destination with the pre-specified amount of fuel, we now ask how much fuel to take such that we arrive at the planned destination in 99.9% of the cases. A key difference to multi-cost bounded reachability as described earlier is that we do not know a priori how far to unfold the MDP. The main algorithm for quantiles iteratively extends the unfolding, reusing the ideas developed for an efficient implementation for multi-cost bounded reachability. The algorithm thereby explores a frontier of the set of cost limits for which the probability threshold holds. To ensure that the representation of the frontier is finite, already in the single-bounded case, some preprocessing is necessary, see e.g. [2]. We generalise these preprocessing steps to the multi-bounded case, and show that this is not always straightforward.

Our new approach has been implemented in the probabilistic model checker STORM [21]. We evaluate its performance, compared to the traditional unfolding approach, on a number of MDP examples as well as on discrete-time Markov chains as a special case of MDPs. We find that the new approach provides not only the expected memory advantage, but is usually faster, too, especially for high cost limits.

In addition, we equip our algorithm with means to visualise (inspired by the recent techniques in [43]) the tradeoffs between various objectives that go beyond Pareto curves. We believe that this is key to obtain better insights into multi-objective decision making. An example is given in Fig. 1b: it depicts the probability (indicated by the colours) to satisfy an objective based on the remaining energy (y-axis) and time (x-axis). Our visualisations provide a way to inspect all of the data that our algorithm implicitly computes anyway. The key challenge here is to reduce the dimensionality of the available data to make the available information easy to grasp without obscuring important dependencies. As such, our visualisations are a first proposal, and come with a call to visualisation experts for improved methods.

Related Work The analysis of single-objective (cost-bounded) reachability in MDPs is an active area of research in both the AI and the formal methods communities, and referred to in e.g. [3,18,38,59]. Various model checking approaches for single objectives exist. In [35], the topology of the unfolded MDP is exploited to speed up the value iteration. In [28], three different model checking approaches are explored and compared. A survey for heuristic approaches is given in [53]. A Q-learning based approach is described in [13]. An extension of this problem to the partially observable setting was considered in [14], and to probabilistic timed automata in [28]. Quantile queries with a single cost bound have been studied in [57]. Multiple cost bounds where all but one cost limits are fixed a priori have been considered in [2]: the idea is to explicitly unfold the model with respect to the given cost bounds, effectively transforming the query to a single-dimensional one. [37] presents a symbolic implementation of these approaches. The method of [4] computes optimal expected values under e.g. the *condition* that the goal is reached, and is thus applicable in settings where a goal is not *necessarily* reached. A similar problem is considered in [55]. For multi-objective analysis, the model checking community typically focuses on probabilities and expected costs as in the seminal works [15,23]. Implementations are typically based on a value iteration approach as

in [25], and have been extended to stochastic games [16], Markov automata [47], and interval MDPs [30]. Other considered cases include e.g. multi-objective mean-payoff objectives [8], objectives over instantaneous costs [10], and parity objectives [7]. Multi-objective problems for MDPs with an unknown cost-function are considered in [36]. Surveys on multi-objective decision making in AI and machine learning can be found in [51] and [58], respectively.

This article is an extended version of a previous conference paper [32]. We provide more details on the core algorithms, extended proofs, an expanded explanation of our visualisations, and additional models in the experimental evaluation. We added Sect. 5, which presents methods for computing multi-cost quantiles, for which we also provide an experimental evaluation in Sect. 7.

Structure of the Paper After the preliminaries (in Sect. 2), we first recap the existing unfolding technique that the new approach conceptually builds upon (in Sect. 3). Then, we present (in Sect. 4) our approach to computing the Pareto curve under multiple cost bounds. We use similar techniques to compute multi-cost quantiles (outlined in Sect. 5). Finally, we show proposals for visualisations of the available data (in Sect. 6), and empirically evaluate the proposed algorithms based on their implementation in STORM (in Sect. 7).

2 Preliminaries

We first introduce notation used throughout this article, then define the model of Markov decision processes, its semantics, and the multi-objective cost-bounded properties that we are interested in.

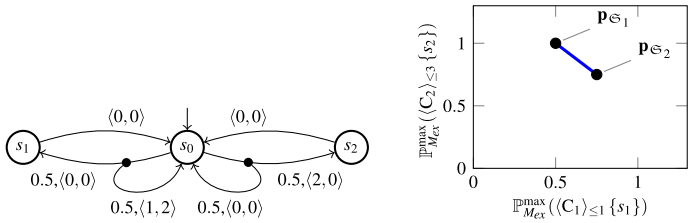
2.1 Mathematical Notation

The i-th component of a tuple $\mathbf{t} = \langle v_1, \ldots, v_n \rangle$ is $\mathbf{t}[i] \stackrel{\text{def}}{=} v_i$. Given a set Ω, we write 2^Ω for its powerset. A (discrete) *probability distribution* over Ω is a function $\mu \colon \Omega \to [0, 1]$ such that $\text{support}(\mu) \stackrel{\text{def}}{=} \{ \omega \in \Omega \mid \mu(\omega) > 0 \}$ is countable and $\sum_{\omega \in \text{support}(\mu)} \mu(\omega) = 1$. $\text{Dist}(\Omega)$ is the set of all probability distributions over Ω. $\mathscr{D}(s)$ is the *Dirac distribution* for s, defined by $\mathscr{D}(s)(s) = 1$. We use the Iverson bracket notation $[cond]$ for Boolean expressions *cond*: $[cond] = 1$ if *cond* is *true* and $[cond] = 0$ otherwise.

2.2 Markov Decision Processes

Markov decision processes (MDPs) combine nondeterministic choices, capturing e.g. user input, scheduler decisions, or unknown and possibly adversarial environments, with probabilistic behaviour as in discrete-time Markov chains. They are the fundamental model for decision-making under uncertainty. We use MDPs in which the branches of transitions are annotated with (multiple) integer *costs* (also called *rewards*), allowing properties to observe quantities such as the passage of discrete time, energy usage, or monetary costs of decision outcomes.

Definition 1 A *Markov decision process* (MDP) with m cost structures is a triple $M = \langle S, T, s_{init} \rangle$ where S is a finite set of states, $T \colon S \to 2^{\text{Dist}(\mathbb{N}^m \times S)}$ is the transition function, and $s_{init} \in S$ is the initial state. For all $s \in S$, we require that $T(s)$ is finite and non-empty. M is a *discrete-time Markov chain* (DTMC) if $\forall s \in S \colon |T(s)| = 1$.



(a) M_{ex} **(b)** $Pareto(M_{ex}, \Phi)$

Fig. 2 Example MDP and Pareto curve

We write $s \to_T \mu$ for $\mu \in T(s)$ and call it a *transition*. We write $s \xrightarrow{\mathbf{c}}_T s'$ if additionally $\langle \mathbf{c}, s' \rangle \in \text{support}(\mu)$ and call $\langle \mathbf{c}, s' \rangle$ a *branch* with cost vector $\mathbf{c}$. If T is clear from the context, we just write $\to$ in place of $\to_T$. Graphically, we represent transitions by lines to a node from which branches labelled with their probability and costs lead to successor states. We may omit the node and probability for transitions into Dirac distributions.

Example 2 Figure 2a shows an example MDP M_{ex}. From the initial state s_0, the choice of going towards s_1 or s_2 is nondeterministic. Either way, the probability to stay in s_0 is 0.5, otherwise we move to s_1 (or s_2). M_{ex} has two cost structures: Failing to move to s_1 has a cost of 1 for the first, and 2 for the second structure. Moving to s_2 yields cost 2 for the first and no cost for the second structure.

Using MDPs directly to build complex models is cumbersome. In practice, high-level formalisms like PRISM's [39] guarded command language or the high-level modelling language MODEST [29] are used to specify MDPs. Aside from a parallel composition operator, they extend MDPs with variables over finite domains that can be used in expressions to e.g. enable or disable transitions. Their semantics is an MDP whose states are the valuations of the variables. This allows to compactly describe very large MDPs.

2.3 Paths and Schedulers

For the remainder of this article, we fix an MDP $M = \langle S, T, s_{init} \rangle$. Its semantics is captured by the notion of *paths*. A path in M represents the infinite concrete resolution of both nondeterministic and probabilistic choices.

Definition 2 A *path* in M is an infinite sequence

$$\pi = s_0 \, \mu_0 \, \mathbf{c}_0 \, s_1 \, \mu_1 \, \mathbf{c}_1 \ldots$$

where $s_i \in S$, $s_i \to \mu_i$ and $\langle \mathbf{c}_i, s_{i+1} \rangle \in \text{support}(\mu_i)$ for all $i \in \mathbb{N}$. A *finite path*

$$\pi_{\text{fin}} = s_0 \, \mu_0 \, \mathbf{c}_0 \, s_1 \, \mu_1 \, \mathbf{c}_1 \, s_2 \ldots \mu_{n-1} \, \mathbf{c}_{n-1} \, s_n$$

is a finite prefix of a path with $\text{last}(\pi_{\text{fin}}) \stackrel{\text{def}}{=} s_n \in S$. Let $\text{cost}_i(\pi_{\text{fin}}) \stackrel{\text{def}}{=} \sum_{j=0}^{n-1} \mathbf{c}_j[i]$. $\text{Paths}_{\text{fin}}(M)$ is the set of all finite paths and $\text{Paths}(M)$ the set of all (infinite) paths starting in s_{init}.

An end component is a subset of the states and transitions such that it is possible (by choosing only transitions in the subset) to remain within the subset of states forever (with probability 1).

Definition 3 An *end component* (EC) of M is given by $T' \colon S' \to 2^{\text{Dist}(\mathbb{N}^m \times S)}$ for a non-empty $S' \subseteq S$ such that

– for all $s \in S'$, $T'(s) \subseteq T(s)$ and $s \xrightarrow{\mathbf{c}}_{T'} s'$ implies $s' \in S'$, and
– for all $s, s' \in S'$ there is a finite path in M from s to s' only using transitions in T'.

A scheduler (or *adversary*, *policy*, or *strategy*) resolves the nondeterministic choices.

Definition 4 A function $\mathfrak{S} \colon \mathrm{Paths}_{\mathrm{fin}}(M) \to \mathrm{Dist}(\mathrm{Dist}(\mathbb{N}^m \times S))$ is a *scheduler* for M if

$$\forall \pi_{\mathrm{fin}} \in \mathrm{Paths}_{\mathrm{fin}}(M) \colon \mu \in \mathrm{support}(\mathfrak{S}(\pi_{\mathrm{fin}})) \Rightarrow \mathrm{last}(\pi_{\mathrm{fin}}) \to_T \mu .$$

The set of all schedulers of M is $\mathrm{Sched}(M)$. We call a scheduler $\mathfrak{S} \in \mathrm{Sched}(M)$ *deterministic* if $|\mathrm{support}(\mathfrak{S}(\pi_{\mathrm{fin}}))| = 1$ and *memoryless* if $\mathrm{last}(\pi_{\mathrm{fin}}) = \mathrm{last}(\pi'_{\mathrm{fin}})$ implies $\mathfrak{S}(\pi_{\mathrm{fin}}) = \mathfrak{S}(\pi'_{\mathrm{fin}})$ for all finite paths π_{fin} and π'_{fin}. For simplicity, we also write deterministic memoryless schedulers as functions $\mathfrak{S} \colon S \to \mathrm{Dist}(\mathbb{N}^m \times S)$.

Via the standard cylinder set construction a scheduler $\mathfrak{S}$ induces a probability measure $\mathbb{P}^{\mathfrak{S}}_M$ on measurable sets of paths starting from s_{init}. More details can be found in e.g. [26] for the case of deterministic schedulers and [46, Section 2.1.6] for the general case. We define the *extremal* values $\mathbb{P}^{\max}_M(\Pi) = \sup_{\mathfrak{S} \in \mathrm{Sched}(M)} \mathbb{P}^{\mathfrak{S}}_M(\Pi)$ and $\mathbb{P}^{\min}_M(\Pi) = \inf_{\mathfrak{S} \in \mathrm{Sched}(M)} \mathbb{P}^{\mathfrak{S}}_M(\Pi)$ for measurable $\Pi \subseteq \mathrm{Paths}(M)$. For clarity, we focus on probabilities in this article, but note that expected accumulated costs can be defined analogously (see e.g. [26]) and our methods apply to them with only minor changes.

2.4 Cost-Bounded Reachability

Recall that the branches of an MDP are annotated with tuples of costs. In our notations we use C_j to refer to the j-th cost structure, i.e. the costs obtained by taking the j-th component of each tuple. We are interested in the probabilities of sets of paths that reach certain goal states while respecting a conjunction of multiple cost bounds.

Definition 5 A *cost bound* is given by $\langle \mathrm{C}_j \rangle_{\sim b}\, G$ where C_j with $j \in \{1, \ldots, m\}$ identifies a cost structure, $\sim \in \{<, \leq, >, \geq\}$, $b \in \mathbb{N}$ is a cost limit, and $G \subseteq S$ is a set of goal states. A *cost-bounded reachability formula* is a conjunction $\bigwedge_{i=1}^{n \in \mathbb{N}} (\langle \mathrm{C}_{j_i} \rangle_{\sim_i b_i}\, G_i)$ of cost bounds. It characterises the measurable set of paths Π where, for every i, every $\pi \in \Pi$ has a prefix π^i_{fin} with $\mathrm{last}(\pi^i_{\mathrm{fin}}) \in G_i$ and $\mathrm{cost}_{j_i}(\pi^i_{\mathrm{fin}}) \sim_i b_i$.

We call a cost-bounded reachability formula $\varphi = \bigwedge_{i=1}^{n \in \mathbb{N}} (\langle \mathrm{C}_{j_i} \rangle_{\sim_i b_i}\, G_i)$ *single*-cost bounded if $n = 1$ and *multi*-cost bounded in the general case. A (single-objective) multi-cost bounded reachability query asks for the maximal (minimal) probability to satisfy a conjunction of cost bounds, i.e. for $\mathbb{P}^{opt}_M(\varphi)$ where $opt \in \{\max, \min\}$ and φ is a cost-bounded reachability formula. Unbounded and step-bounded reachability are special cases of cost-bounded reachability. A single-objective query may contain multiple bounds, but asks for a *single* scheduler that optimises the probability of satisfying them all.

Example 3 The single-objective multi-cost bounded query $\mathbb{P}^{\max}_M(\langle \mathrm{C}_1 \rangle_{\leq 1}\, \{s_1\} \wedge \langle \mathrm{C}_2 \rangle_{\leq 2}\, \{s_2\})$ for M_{ex} of Fig. 2a asks for the maximal probability to reach s_1 with at most cost 1 for the first cost structure and s_2 with at most cost 2 for the second cost structure. This probability is 0.5, e.g. attained by the scheduler that tries to move to s_1 once and to s_2 afterwards.

Given multiple objectives (i.e. multiple reachability queries) at once, a scheduler that optimises for one objective might be suboptimal for the other objectives. We thus consider *multi-objective tradeoffs* (or simply *tradeoffs*), i.e. sets of single-objective queries written as

$$\Phi = \mathit{multi}\big(\mathbb{P}^{opt_1}_M(\varphi_1), \ldots, \mathbb{P}^{opt_\ell}_M(\varphi_\ell)\big).$$

The cost-bounded reachability formulas φ_k occurring in Φ are called *objectives*. For tradeoffs, we are interested in the *Pareto curve* $Pareto(M, \Phi)$ which consists of all achievable probability vectors $\mathbf{p}_{\mathfrak{S}} = \langle \mathbb{P}^{\mathfrak{S}}_M(\varphi_1), \ldots, \mathbb{P}^{\mathfrak{S}}_M(\varphi_\ell)\rangle$ for $\mathfrak{S} \in \text{Sched}(M)$ that are not *dominated* by another achievable vector $\mathbf{p}_{\mathfrak{S}'}$. More precisely, $\mathbf{p}_{\mathfrak{S}} \in Pareto(M, \Phi)$ if and only if for all $\mathfrak{S}' \in \text{Sched}(M)$ either $\mathbf{p}_{\mathfrak{S}} = \mathbf{p}_{\mathfrak{S}'}$ or for some $i \in \{1, \ldots, \ell\}$ we have $\mathbf{p}_{\mathfrak{S}}[i] \sqsupset \mathbf{p}_{\mathfrak{S}'}[i]$ with

$$\sqsupset = \begin{cases} > & \text{if } opt_i = \max \\ < & \text{if } opt_i = \min. \end{cases}$$

Example 4 We consider $\Phi = multi\big(\mathbb{P}^{\max}_{M_{ex}}(\langle \mathrm{C}_1\rangle_{\leq 1}\, \{s_1\}), \mathbb{P}^{\max}_{M_{ex}}(\langle \mathrm{C}_2\rangle_{\leq 3}\, \{s_2\})\big)$ for M_{ex} of Fig. 2a. Let $\mathfrak{S}_j$ be the scheduler that tries to move to s_1 for at most j attempts and afterwards almost surely moves to s_2. The induced probability vectors $\mathbf{p}_{\mathfrak{S}_1} = \langle 0.5, 1\rangle$ and $\mathbf{p}_{\mathfrak{S}_2} = \langle 0.75, 0.75\rangle$ both lie on the Pareto curve since no $\mathfrak{S} \in \text{Sched}(M_{ex})$ induces (strictly) larger probabilities $\mathbf{p}_{\mathfrak{S}}$. By also considering schedulers that randomise between the choices of $\mathfrak{S}_1$ and $\mathfrak{S}_2$ we obtain $Pareto(M_{ex}, \Phi) = \{w \cdot \mathbf{p}_{\mathfrak{S}_1} + (1-w) \cdot \mathbf{p}_{\mathfrak{S}_2} \mid w \in [0, 1]\}$. Graphically, the Pareto curve corresponds to the line between $\mathbf{p}_{\mathfrak{S}_1}$ and $\mathbf{p}_{\mathfrak{S}_2}$ as shown in Fig. 2b.

For clarity of presentation in the following sections, and unless otherwise noted, we restrict to tradeoffs Φ where every cost structure occurs exactly once, i.e. the number m of cost structures of M matches the number of cost bounds occurring in Φ. Furthermore, we require that none of the sets of goal states contains the initial state. Both assumptions are without loss of generality since any formula can be made to satisfy this restriction by copying cost structures as needed and adding a new initial state with a zero-cost transition to the old initial state. We will also introduce all ideas with the upper-bounded maximum case first, assuming a tradeoff

$$\Phi = multi\big(\mathbb{P}^{\max}_M(\varphi_1), \ldots, \mathbb{P}^{\max}_M(\varphi_\ell)\big)$$

with ℓ cost-bounded reachability formulas (cf. Definition 5)

$$\varphi_k = \bigwedge_{i=n_{k-1}}^{n_k-1} (\langle \mathrm{C}_i\rangle_{\leq b_i}\, G_i),\ 0 = n_0 < \cdots < n_\ell = m.$$

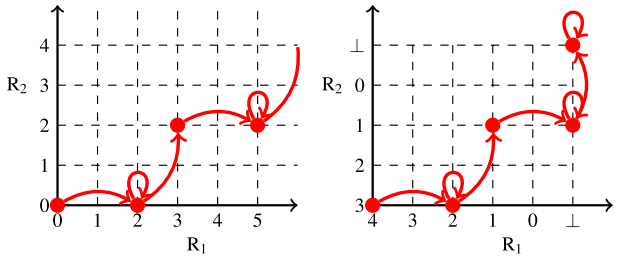
We discuss other bound types in Sect. 4.4, including combinations of lower and upper bounds.

3 The Unfolding Approach Revisited

The classic approach to compute cost-bounded properties is to unfold the accumulated cost into the state space [1]. Our new approach is more memory-efficient than unfolding, but fundamentally rests on the same notions and arguments for correctness. We thus present the unfolding approach in this section first.

3.1 Epochs and Goal Satisfaction

The central concept in our approach is that of *cost epochs*. The idea is that, to compute a Pareto curve, we analyse all reachable cost epochs one by one, in a specific order. Let us start with the most naïve way to track costs: For each path, we can plot the accumulated cost in all dimensions along this path in a cost grid. A coordinate $\langle c_1, \ldots, c_m\rangle$ in the cost grid reflects that the amount of collected cost in dimension i is c_i.



(a) The naïve approach **(b)** Cost epochs

Fig. 3 An illustration of epochs

Example 5 Consider path $\pi = (s_0\ \langle 2,0\rangle\ s_2\ \langle 0,0\rangle\ s_0\ \langle 1,2\rangle)^\omega$ through M_{ex} of Fig. 2a. We plot the collected costs in Fig. 3a. Starting from $\langle 0,0\rangle$, the first transition yields cost 2 for the first cost structure: we jump to coordinate $\langle 2,0\rangle$. The next transition, back to s_0, has no cost, so we stay at $\langle 2,0\rangle$. The failed attempt to move to s_1 incurs costs $\langle 1,2\rangle$, jumping to coordinate $\langle 3,2\rangle$. This series of updates repeats ad infinitum.

For an infinite path, infinitely many points in the cost grid may be reached. These points are therefore not a suitable notion to use in model checking. However, a tradeoff specifies limits for the costs, e.g. for

$$\Phi_{ex} = multi\big(\mathbb{P}^{\max}_{M_{ex}}(\langle \mathrm{C}_1\rangle_{\leq 4}\ \{s_1\}),\ \mathbb{P}^{\max}_{M_{ex}}(\langle \mathrm{C}_2\rangle_{\leq 3}\ \{s_2\})\big)$$

we get cost limits 4 and 3. Once the limit for a cost is reached, accumulating further costs in this dimension does not impact the satisfaction of the corresponding formula. It thus suffices to keep track of the *remaining* costs before reaching the cost limit of each bound. This leads to a finite grid of cost epochs.

Definition 6 An m-dimensional *cost epoch* $\mathbf{e}$ is a tuple in $\mathbf{E}_m \stackrel{\text{def}}{=} (\mathbb{N} \cup \{\bot\})^m$. For $\mathbf{e} \in \mathbf{E}_m$, $\mathbf{c} \in \mathbb{N}^m$, the *successor epoch* $succ(\mathbf{e}, \mathbf{c}) \in \mathbf{E}_m$ is point-wise defined by

$$succ(\mathbf{e}, \mathbf{c})[i] = \begin{cases} \mathbf{e}[i] - \mathbf{c}[i] & \text{if } \mathbf{e}[i] \geq \mathbf{c}[i] \\ \bot & \text{otherwise.} \end{cases}$$

Example 6 Reconsider path π of Example 5 and Φ_{ex} as above. We illustrate the finite epoch grid in Fig. 3b. We start in cost epoch $\langle 4,3\rangle$. The first transition incurs cost $\langle 2,0\rangle$, and subsequently the remaining cost before reaching the bound is $\langle 2,3\rangle$. These updates continue analogously to Example 5. From $\langle 1,1\rangle$ taking cost $\langle 2,0\rangle$ means that we violate the bound in the first dimension. We indicate this violation with $\bot$, and move to $\langle \bot, 1\rangle$. Later, taking cost $\langle 1,2\rangle$ does not change that we already violated the first bound: the first entry remains $\bot$, but as we now also violate the second bound, we move to $\langle \bot, \bot\rangle$. We then remain in this cost epoch forever.

Recall that we consider upper bounds. Consequently, if the entry for a bound is $\bot$, it cannot be satisfied any more: too much cost has already been incurred. To check whether an objective $\varphi_k = \bigwedge_{i=n_{k-1}}^{n_k-1} (\langle \mathrm{C}_i\rangle_{\leq b_i}\ G_i)$ is satisfied, we need to memorise whether each individual bound already holds, that is, whether we have reached a state in G_i before exceeding the cost limit.

Definition 7 A *goal satisfaction* $\mathbf{g} \in \mathbf{G}_m \stackrel{\text{def}}{=} \{0,1\}^m$ represents the cost structure indices i for which bound $\langle \mathrm{C}_i\rangle_{\leq b_i}\ G_i$ already holds, i.e. G_i was reached before exceeding the cost

limit b_i. For $\mathbf{g} \in \mathbf{G}_m$, $\mathbf{e} \in \mathbf{E}_m$ and $s \in S$, let $succ(\mathbf{g}, s, \mathbf{e}) \in \mathbf{G}_m$ define the update upon reaching s:

$$succ(\mathbf{g}, s, \mathbf{e})[i] \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } s \in G_i \wedge \mathbf{e}[i] \neq \bot \\ \mathbf{g}[i] & \text{otherwise.} \end{cases}$$

Example 7 Reconsider path π of Example 5 and Φ_{ex} as previously. As $s_0 \notin G_i$, we start with $\mathbf{g} = \langle 0, 0 \rangle$. We visit s_1 and update $\mathbf{g}$ to $\langle 1, 0 \rangle$ since $s_1 \in G_1$ and $s_1 \notin G_2$. We then visit s_0 and $\mathbf{g}$ remains $\langle 1, 0 \rangle$. After that, upon visiting s_2, $\mathbf{g}$ is updated to $\langle 1, 1 \rangle$.

3.2 The Unfolding Approach

We can now compute $Pareto(M, \Phi)$ by reducing Φ to a multi-objective *unbounded* reachability problem on the *unfolded* MDP M_{unf}. The states of M_{unf} are the Cartesian product of the original MDP's states, the epochs, and the goal satisfactions, thereby effectively generalising the construction from [1].

Definition 8 The *unfolding* for an MDP M as in Definition 1 and upper-bounded maximum tradeoff Φ is the MDP

$$M_{unf} = \langle S', T', s'_{init} \rangle$$

with $S' = S \times \mathbf{E}_m \times \mathbf{G}_m$, $s'_{init} = \langle s_{init}, \langle b_1, \ldots, b_m \rangle, \mathbf{0} \rangle$, no cost structures, and

$$T'(\langle s, \mathbf{e}, \mathbf{g} \rangle) \stackrel{\text{def}}{=} \{\, unf(\mu) \in \mathrm{Dist}(S') \mid \mu \in T(s) \,\}$$

where

$$unf(\mu)\left(\langle s', \mathbf{e}', \mathbf{g}' \rangle\right) \stackrel{\text{def}}{=} \mu(\langle \mathbf{c}, s' \rangle) \cdot [\mathbf{e}' = succ(\mathbf{e}, \mathbf{c})] \cdot [\mathbf{g}' = succ(\mathbf{g}, s', \mathbf{e}')].$$

Transitions and probabilities are thus as before, but if a branch is equipped with costs, we update the cost epoch entry in the state; likewise, if a state is a goal state, we update the corresponding goal satisfaction entry. As costs are now encoded in the state space, it suffices to consider the *unbounded* tradeoff

$$\Phi' = multi\big(\mathbb{P}^{\max}_{M_{unf}}(\varphi'_1), \ldots, \mathbb{P}^{\max}_{M_{unf}}(\varphi'_\ell)\big)$$

with

$$\varphi'_k = \langle\cdot\rangle_{\geq 0}\, G'_k,\ G'_k = \left\{ \langle s, \mathbf{e}, \mathbf{g} \rangle \mid \bigwedge_{i=n_{k-1}}^{n_k - 1} \mathbf{g}[i] = 1 \right\}.$$

Example 8 Consider M_{ex} of Fig. 2a and Φ_{ex} as previously. Figure 4 contains a fragment of the unfolding.

Lemma 1 *There is a bijection* $\xi\colon \mathrm{Sched}(M) \to \mathrm{Sched}(M_{unf})$ *with* $\mathbb{P}^{\mathfrak{S}}_M(\varphi_k) = \mathbb{P}^{\xi(\mathfrak{S})}_{M_{unf}}(\varphi'_k)$ *for all* $\mathfrak{S} \in \mathrm{Sched}(M)$ *and* $k \in \{1, \ldots, \ell\}$. *Consequently, we have that* $Pareto(M, \Phi) = Pareto(M_{unf}, \Phi')$.

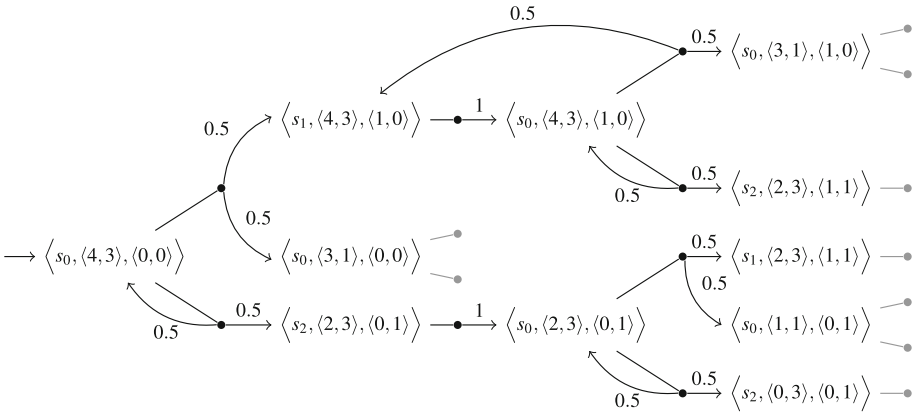


Fig. 4 Initial fragment of the unfolding. Successors of actions in gray are omitted

3.3 Multi-objective Model Checking on the Unfolding

$Pareto(M_{unf}, \Phi')$ can be computed with existing multi-objective model checking algorithms for unbounded reachability. We build on the approach of [25]: we iteratively choose weight vectors $\mathbf{w} = \langle w_1, \ldots, w_\ell \rangle \in [0,1]^\ell \setminus \{\mathbf{0}\}$ and compute points

$$\mathbf{p_w} = \langle \mathbb{P}^{\mathfrak{S}}_{M_{unf}}(\varphi'_1), \ldots, \mathbb{P}^{\mathfrak{S}}_{M_{unf}}(\varphi'_\ell) \rangle \text{ with } \mathfrak{S} \in \arg\max_{\mathfrak{S}'} \left(\sum_{k=1}^{\ell} w_k \cdot \mathbb{P}^{\mathfrak{S}'}_{M_{unf}}(\varphi'_k) \right). \tag{1}$$

The Pareto curve $Pareto(M_{unf}, \Phi')$ is convex, has finitely many vertices, and contains the point $\mathbf{p_w}$ for each weight vector $\mathbf{w}$. Moreover, $\mathbf{q} \cdot \mathbf{w} > \mathbf{p_w} \cdot \mathbf{w}$ implies $\mathbf{q} \notin Pareto(M_{unf}, \Phi')$. These observations enable us to approximate the Pareto curve with arbitrary precision by enumerating its vertices $\mathbf{p_w}$ in a smart order. At any point, the algorithm maintains two convex areas which under- and overapproximate the area under the Pareto curve. Further details are given in [25], including a method to compute a bound on the error at any stage.

To reduce the computation of $\mathbf{p_w}$ to standard MDP model checking, [25] characterises $\mathbf{p_w}$ via weighted expected costs: we construct M^+_{unf} from M_{unf}. States and transitions are as in M_{unf}, but M^+_{unf} is additionally equipped with ℓ cost structures used to calculate the probability of each of the ℓ objectives. This is achieved by setting the value of the k-th cost structure on each branch to 1 if and only if the objective φ'_k is satisfied in the target state of the branch but was *not* satisfied in the transition's source state. More precisely, the cost of a branch $\langle s, \mathbf{e}, \mathbf{g} \rangle \xrightarrow{\mathbf{c}} \langle s', \mathbf{e}', \mathbf{g}' \rangle$ in M^+_{unf} is set to $\mathbf{c} = satObj_\Phi(\mathbf{g}, \mathbf{g}')$, where function

$$satObj_\Phi : \mathbf{G}_m \times \mathbf{G}_m \to \{0,1\}^\ell$$

is point-wise defined by

$$satObj_\Phi(\mathbf{g}, \mathbf{g}')[k] = \begin{cases} 1 & \text{if } \exists\, i : \mathbf{g}[i] = 0 \text{ and } \forall\, i : \mathbf{g}'[i] = 1 \text{ where } i \in \{n_{k-1}, \ldots, n_k - 1\} \\ 0 & \text{otherwise.} \end{cases}$$

On a path π through M^+_{unf}, we collect exactly cost 1 for cost structure k if and only if π satisfies objective φ_k.

Definition 9 For $\mathfrak{S} \in \text{Sched}(M^+_{unf})$ and $\mathbf{w} \in [0,1]^\ell$, the *weighted expected cost* is

$$\mathbb{E}^{\mathfrak{S}}_{M^+_{unf}}(\mathbf{w}) = \sum_{k=1}^{\ell} \mathbf{w}[k] \cdot \int_{\pi \in \text{Paths}(M)} \text{cost}_k(\pi) \mathrm{d}\mathbb{P}^{\mathfrak{S}}_{M^+_{unf}}(\pi).$$

The weighted expected cost is the expected value of the weighted sum of the costs accumulated on paths in M^+_{unf}. In the definition, we consider a Lebesgue integral instead of a sum since $\text{Paths}(M)$ is generally uncountable. The *maximal* weighted expected cost for M^+_{unf} and $\mathbf{w}$ is given by $\mathbb{E}^{\max}_{M^+_{unf}}(\mathbf{w}) = \max_{\mathfrak{S}} \mathbb{E}^{\mathfrak{S}}_{M^+_{unf}}(\mathbf{w})$. There is always a deterministic, memoryless scheduler $\mathfrak{S}$ that attains the maximal expected cost [46].

The following characterisation of $\mathbf{p}_{\mathbf{w}}$ is equivalent to Eq. 1:

$$\begin{aligned} \mathbf{p}_{\mathbf{w}} = \langle \mathbb{E}^{\mathfrak{S}}_{M^+_{unf}}(\mathbf{1}_1), \ldots, \mathbb{E}^{\mathfrak{S}}_{M^+_{unf}}(\mathbf{1}_\ell) \rangle \quad \text{where} \quad \mathfrak{S} \in \arg\max_{\mathfrak{S}'} \mathbb{E}^{\mathfrak{S}'}_{M^+_{unf}}(\mathbf{w}), \text{ and} \\ \mathbf{1}_k \in \{0,1\}^\ell \text{ defined by } \mathbf{1}_k[j] = 1 \text{ iff } j = k. \end{aligned} \tag{2}$$

Standard MDP model checking algorithms [46] can be applied to compute an optimal (deterministic and memoryless) scheduler $\mathfrak{S}$ and the induced costs $\mathbb{E}^{\mathfrak{S}}_{M^+_{unf}}(\mathbf{1}_k)$.

4 Multi-cost Multi-objective Sequential Value Iteration

An unfolding-based approach as discussed in Sect. 3.2 does not scale well in terms of memory consumption: If the original MDP has n states, then the unfolding has on the order of $n \cdot \prod_{i=1}^{m}(b_i+2)$ states. This blow-up makes an a priori unfolding infeasible for larger cost limits b_i over multiple bounds. The bottleneck lies in computing the points $\mathbf{p}_{\mathbf{w}}$ as in equations 1 and 2. In this section, we show how to compute these probability vectors efficiently, i.e. given a weight vector $\mathbf{w} = \langle w_1, \ldots, w_\ell \rangle \in [0,1]^\ell \setminus \{\mathbf{0}\}$, compute

$$\mathbf{p}_{\mathbf{w}} = \langle \mathbb{P}^{\mathfrak{S}}_M(\varphi_1), \ldots, \mathbb{P}^{\mathfrak{S}}_M(\varphi_\ell) \rangle \text{ with } \mathfrak{S} \in \arg\max_{\mathfrak{S}'} \left(\sum_{k=1}^{\ell} w_i \cdot \mathbb{P}^{\mathfrak{S}'}_M(\varphi_k) \right) \tag{3}$$

without creating the unfolding. The two characterisations of $\mathbf{p}_{\mathbf{w}}$ given in equations 1 and 3 are equivalent due to Lemma 1.

The efficient analysis of single-objective queries $\Phi_1 = \mathbb{P}^{\max}_M(\langle \mathrm{C} \rangle_{\leq b}\, G)$ with a single bound has recently been addressed [28,37]. The key idea is based on *dynamic programming*. The unfolding M_{unf} is decomposed into $b+2$ *epoch models* MDPs $M^b, \ldots, M^0, M^\perp$ such that the epoch model MDPs correspond to the cost epochs. Each epoch model MDP is a copy of M with only slight adaptations (detailed later). The crucial observation is that, since costs are non-negative, reachability probabilities in copies corresponding to epoch i only depend on the copies $\{ M^j \mid j \leq i \vee j = \perp \}$. It is thus possible to analyse $M^\perp, \ldots, M^b$ *sequentially* instead of considering all copies at once. In particular, it is not necessary to construct the complete unfolding.

We lift this idea to multi-objective tradeoffs with multiple cost bounds: we aim to build an MDP for each epoch $\mathbf{e} \in \mathbf{E}_m$ that can be analysed via standard model checking techniques using the weighted expected cost encoding of objective probabilities. Notably, in the single cost bound case with a single objective, it is easy to determine whether the one property is satisfied: either reaching a goal state for the first time or exceeding the cost bound immediately suffices to determine whether the property is satisfied. Thus, while $M^\perp$ is just one sink state in

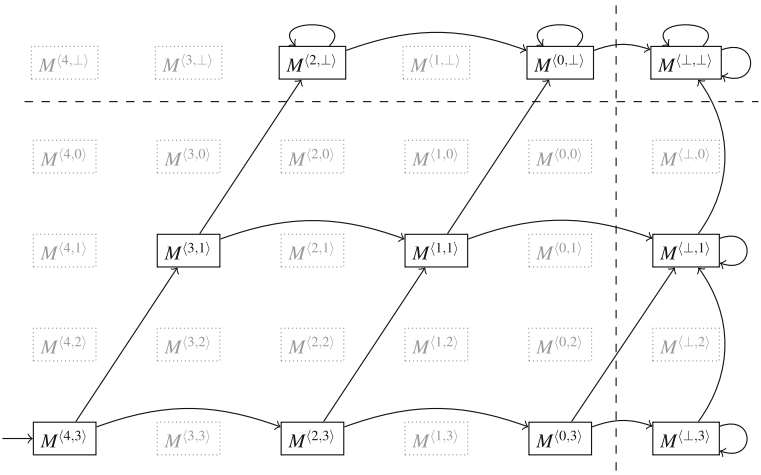


Fig. 5 Epoch models reachable from $M^{\langle 4,3\rangle}$ in a grid

the single cost bound case, its structure is more involved in the presence of multiple objectives and multiple cost bounds.

4.1 An Epoch Model Approach without Unfolding

We first formalise epoch models for multiple bounds. As noted, the overall epoch structure is the same as in the unfolding approach.

Example 9 We illustrate the structure of the epoch models in Fig. 5. For our running example MDP M_{ex} of Fig. 2a with bounds 4 and 3, we obtain $(4+2)\cdot(3+2) = 30$ epoch models. The epoch models can be partitioned into 4 partitions (indicated by the dashed lines), with all epoch models inside a partition having the same MDP structure. The overall graph of the epoch models is acyclic (up to self-loops). From the maximum costs in M_{ex}, we a priori know that e.g. epoch model $M^{\langle 2,1\rangle}$ can only be reached from epochs $M^{\langle i,j\rangle}$ with $i \leq 2$, $j \leq 1$. In our illustration, we only show the transitions between the epoch models that are forward-reachable from $M^{\langle 4,3\rangle}$; observe that in this example, these are significantly fewer than what the backward-reachability argument based on the maximum costs gives, which are again only a fraction of all possible epochs.

Before we give a formal definition of the epoch model in Definition 10, we give an intuitive description. The state space of an individual epoch model for epoch $\mathbf{e}$ consists of up to one copy of each original state for each of the 2^m goal satisfaction vectors $\mathbf{g} \in \mathbf{G}_m$. Additional sink states $\langle s_\perp, \mathbf{g}\rangle$ encode the target for a jump to *any* other cost epoch $\mathbf{e}' \neq \mathbf{e}$. Similar to the unfolding M^{+}_{unf}, we use the function $satObj_\Phi : \mathbf{G}_m \times \mathbf{G}_m \to \{0,1\}^\ell$ to assign cost 1 for objectives that change from not (yet) satisfied to satisfied, based on the information in the two goal satisfaction vectors. More precisely, we put cost 1 in entry $1 \leq k \leq m$ if and only if a reachability property φ_k is satisfied according to the target goal satisfaction vector and not in the previous goal satisfaction vector. For the transitions' branches, we distinguish two cases:

1. If the successor epoch $\mathbf{e}' = succ(\mathbf{e}, \mathbf{c})$ with respect to the *original* cost $\mathbf{c} \in \mathbb{N}^m$ of M is the same as the current epoch $\mathbf{e}$, we jump to the successor state as before, and update the goal

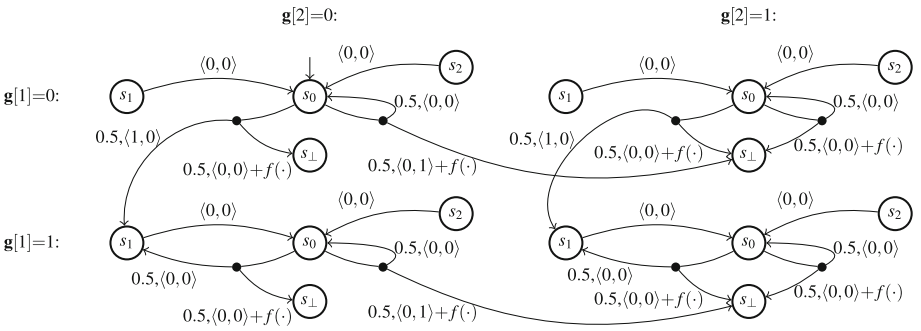


Fig. 6 One epoch model of M_{ex}

satisfaction. We collect the *new* costs for the *objectives* if the updated goal satisfaction newly satisfies an objective given by $satObj_\Phi$, i.e. if it is now satisfied by the new goal satisfaction and the old goal satisfaction did not satisfy that objective.

2. If the successor epoch $\mathbf{e}' = succ(\mathbf{e}, \mathbf{c})$ is different from the current epoch $\mathbf{e}$, the transitions' branch is redirected to the sink state $\langle s_\perp, \mathbf{g}' \rangle$ with the corresponding goal state satisfaction vector. Notice that this might require to merge some branches, hence we have to sum over all branches.

The collected costs contain the part of the goal satisfaction as in item 1, but also the results obtained by analysing the successor epoch $\mathbf{e}'$. The latter is incorporated by a function $f : \mathbf{G}_m \times \text{Dist}(\mathbb{N}^m \times S) \to [0, 1]^\ell$ such that the k-th entry of the vector $f(\mathbf{g}, \mu)$ reflects the probability to newly satisfy the k-th objective after leaving the current epoch via distribution μ.

Definition 10 The *epoch model* of an MDP M as in Definition 1 for $\mathbf{e} \in \mathbf{E}_m$ and a function $f : \mathbf{G}_m \times \text{Dist}(\mathbb{N}^m \times S) \to [0, 1]^\ell$ is the MDP $M_f^{\mathbf{e}} = \langle S^{\mathbf{e}}, T_f^{\mathbf{e}}, \langle s_{init}, \mathbf{0} \rangle \rangle$ with ℓ cost structures defined by

$$S^{\mathbf{e}} \stackrel{\text{def}}{=} (S \uplus s_\perp) \times \mathbf{G}_m, \quad T_f^{\mathbf{e}}(\langle s_\perp, \mathbf{g} \rangle) = \{ \mathscr{D}(\langle \mathbf{0}, \langle s_\perp, \mathbf{g} \rangle \rangle) \},$$

and for every $\tilde{s} = \langle s, \mathbf{g} \rangle \in S^{\mathbf{e}}$ and $\mu \in T(s)$, there is a $\nu \in T_f^{\mathbf{e}}(\tilde{s})$ such that

1. $\nu(\langle satObj_\Phi(\mathbf{g}, \mathbf{g}'), \langle s', \mathbf{g}' \rangle \rangle) = \mu(\mathbf{c}, s') \cdot [succ(\mathbf{e}, \mathbf{c}) = \mathbf{e}] \cdot [succ(\mathbf{g}, s', \mathbf{e}) = \mathbf{g}']$
2. $\nu(\langle satObj_\Phi(\mathbf{g}, \mathbf{g}') + f(\mathbf{g}, \mu), \langle s_\perp, \mathbf{g}' \rangle \rangle) = \sum_{\langle \mathbf{c}, s' \rangle} \mu(\mathbf{c}, s') \cdot [succ(\mathbf{e}, \mathbf{c}) = \mathbf{e}' \neq \mathbf{e}] \cdot [succ(\mathbf{g}, s', \mathbf{e}') = \mathbf{g}']$.

In contrast to Definition 1, the MDP $M_f^{\mathbf{e}}$ may consider cost vectors that consist of non-natural numbers—as reflected by the image of f. The two items in the definition reflect the two cases described before. For item 2, the sum $satObj_\Phi(\mathbf{g}, \mathbf{g}') + f(\mathbf{g}, \mu)$ reflects the two cases where an objective is satisfied in the current step (upon taking a branch that leaves the epoch) or only afterwards. In particular, our algorithm constructs f in a way that $satObj_\Phi(\mathbf{g}, \mathbf{g}')[k] = 1$ implies $f(\mathbf{g}, \mu)[k] = 0$.

Example 10 Figure 6 shows an epoch model $M_f^{\mathbf{e}}$ of the MDP M_{ex} in Fig. 2a with respect to tradeoff Φ as in Example 4 and any epoch $\mathbf{e} \in \mathbf{E}_m$ in the partition where $\mathbf{e}[1] \neq \perp$ and $\mathbf{e}[2] \neq \perp$.

As already mentioned before, the structure of $M_f^{\mathbf{e}}$ differs only slightly between epochs. In particular consider epochs $\mathbf{e}$ and $\mathbf{e}'$ with $\mathbf{e}[i] = \bot$ if and only if $\mathbf{e}'[i] = \bot$. To construct epoch model $M_{f'}^{\mathbf{e}'}$ from $M_f^{\mathbf{e}}$, only transitions to the bottom states $\langle s_\bot, \mathbf{g}\rangle$ need to be adapted, by adapting f accordingly.

Consider the unfolding M_{unf}^+ with ℓ cost structures as in Sect. 3.3. Intuitively, the states of $M_f^{\mathbf{e}}$ reflect the states of M_{unf}^+ with cost epoch $\mathbf{e}$. We use the function f to propagate values for the remaining states of M_{unf}^+. This is formalised by the following lemma. We use the notation $\mathbb{E}^{\mathfrak{S}}_{M_{unf}^+}(\mathbf{w})[\langle s, \mathbf{e}, \mathbf{g}\rangle]$ for the weighted expected costs for M_{unf}^+ when changing the initial state to $\langle s, \mathbf{e}, \mathbf{g}\rangle$.

Lemma 2 *Let $M = \langle S, T, s_{init}\rangle$ be an MDP with unfolding $M_{unf}^+ = \langle S', T', s'_{init}\rangle$ as above. Further, let $M_f^{\mathbf{e}} = \langle S^{\mathbf{e}}, T_f^{\mathbf{e}}, \langle s_{init}, \mathbf{0}\rangle\rangle$ be an epoch model of M for epoch $\mathbf{e} \in \mathbf{E}_m$, and f given by*

$$f(\mathbf{g}, \mu)[k] = \frac{1}{\mu_{\text{exit}}} \sum_{\langle \mathbf{c}, s'\rangle} \mu(\mathbf{c}, s') \cdot [succ(\mathbf{e}, \mathbf{c}) = \mathbf{e}' \neq \mathbf{e}] \cdot \mathbb{E}^{\max}_{M_{unf}^+}(\mathbf{1}_k)[\langle s', \mathbf{e}', succ(\mathbf{g}, s', \mathbf{e}')\rangle]$$

if $\mu_{\text{exit}} = \sum_{\langle \mathbf{c}, s\rangle} \mu(\mathbf{c}, s) \cdot [succ(\mathbf{e}, \mathbf{c}) \neq \mathbf{e}] > 0$ and $f(\mathbf{g}, \mu)[k] = 0$ otherwise. For every weight vector $\mathbf{w} \in [0, 1]^\ell$ and state $\langle s, \mathbf{g}\rangle$ of $M_f^{\mathbf{e}}$ with $s \neq s_\bot$ we have

$$\mathbb{E}^{\max}_{M_{unf}^+}(\mathbf{w})[\langle s, \mathbf{e}, \mathbf{g}\rangle] = \mathbb{E}^{\max}_{M_f^{\mathbf{e}}}(\mathbf{w})[\langle s, \mathbf{g}\rangle].$$

Proof We apply the characterisation of (weighted) expected rewards as the smallest solution of a Bellman equation system [25,46]. For M_{unf}^+, assume variables $\mathbf{x}[\langle s, \hat{\mathbf{e}}, \mathbf{g}\rangle] \in \mathbb{R}_{\geq 0}$ for every $\langle s, \hat{\mathbf{e}}, \mathbf{g}\rangle \in S'$. The smallest solution of the equation system

$$\forall \langle s, \hat{\mathbf{e}}, \mathbf{g}\rangle \in S' : \quad \mathbf{x}[\langle s, \hat{\mathbf{e}}, \mathbf{g}\rangle] = \max_{\mu \in T'(\langle s, \hat{\mathbf{e}}, \mathbf{g}\rangle)} \left(\sum_{\langle \mathbf{c}, \hat{s}\rangle} \mu(\langle \mathbf{c}, \hat{s}\rangle) \cdot \left(\mathbf{w} \cdot \mathbf{c} + \mathbf{x}[\hat{s}]\right)\right) \tag{4}$$

satisfies $\mathbf{x}[\langle s, \hat{\mathbf{e}}, \mathbf{g}\rangle] = \mathbb{E}^{\max}_{M_{unf}^+}(\mathbf{w})[\langle s, \hat{\mathbf{e}}, \mathbf{g}\rangle]$. Similarly, for $M_f^{\mathbf{e}}$, the smallest solution of

$$\forall \langle s, \mathbf{g}\rangle \in S^{\mathbf{e}} : \quad \mathbf{y}^{\mathbf{e}}[\langle s, \mathbf{g}\rangle] = \max_{\nu \in T_f^{\mathbf{e}}(\langle s, \mathbf{g}\rangle)} \left(\sum_{\langle \mathbf{c}, \tilde{s}\rangle} \nu(\langle \mathbf{c}, \tilde{s}\rangle) \cdot \left(\mathbf{w} \cdot \mathbf{c} + \mathbf{y}^{\mathbf{e}}[\tilde{s}]\right)\right) \tag{5}$$

satisfies $\mathbf{y}^{\mathbf{e}}[\langle s, \mathbf{g}\rangle] = \mathbb{E}^{\max}_{M_f^{\mathbf{e}}}(\mathbf{w})[\langle s, \mathbf{g}\rangle]$. We prove the lemma by showing the following claim: If $\mathbf{x}[\langle s, \hat{\mathbf{e}}, \mathbf{g}\rangle]$ for $\langle s, \hat{\mathbf{e}}, \mathbf{g}\rangle \in S'$ is the smallest solution for Eq. 4, the smallest solution for Eq. 5 is given by $\mathbf{y}^{\mathbf{e}}[\langle s, \mathbf{g}\rangle] = [s \neq s_\bot] \cdot \mathbf{x}[\langle s, \mathbf{e}, \mathbf{g}\rangle]$ for $\langle s, \mathbf{g}\rangle \in S^{\mathbf{e}}$.

Let $\mathbf{x}[\langle s, \hat{\mathbf{e}}, \mathbf{g}\rangle]$ be the smallest solution for Eq. 4. Since no cost can be reached from $s_\bot$ in $M_f^{\mathbf{e}}$, we can show that $\mathbf{y}^{\mathbf{e}}[\langle s_\bot, \mathbf{g}\rangle] = 0$ has to hold. Now let $\langle s, \mathbf{g}\rangle \in S^{\mathbf{e}}$ with $s \neq s_\bot$. To improve readability, we use $\mathbf{e}'$ as short for $succ(\mathbf{e}, \mathbf{c})$ and $\mathbf{g}'$ as short for $succ(\mathbf{g}, s', \mathbf{e}')$.

$$\begin{aligned}
\mathbf{y}^{\mathbf{e}}[\langle s, \mathbf{g}\rangle] &= [s \neq s_\bot] \cdot \mathbf{x}[\langle s, \mathbf{e}, \mathbf{g}\rangle] = \mathbf{x}[\langle s, \mathbf{e}, \mathbf{g}\rangle] \\
&= \max_{\mu \in T'(\langle s, \mathbf{e}, \mathbf{g}\rangle)} \left(\sum_{\langle \mathbf{c}, \hat{s}\rangle} \mu(\langle \mathbf{c}, \hat{s}\rangle) \cdot \left(\mathbf{w} \cdot \mathbf{c} + \mathbf{x}[\hat{s}]\right)\right) \\
&= \max_{\mu \in T(s)} \left(\sum_{\langle \mathbf{c}, s'\rangle} \mu(\langle \mathbf{c}, s'\rangle) \cdot \left(\mathbf{w} \cdot satObj_\Phi(\mathbf{g}, \mathbf{g}') + \mathbf{x}[\langle s', \mathbf{e}', \mathbf{g}'\rangle]\right)\right)
\end{aligned}$$

$$
\begin{aligned}
&= \max_{\mu \in T(s)} \Bigg(\sum_{\langle \mathbf{c}, s' \rangle} [\mathbf{e} = \mathbf{e}'] \cdot \mu(\langle \mathbf{c}, s' \rangle) \cdot \big(\mathbf{w} \cdot \mathit{satObj}_\Phi(\mathbf{g}, \mathbf{g}') + \mathbf{x}[\langle s', \mathbf{e}', \mathbf{g}' \rangle] \big) \\
&\quad + \sum_{\langle \mathbf{c}, s' \rangle} [\mathbf{e} \neq \mathbf{e}'] \cdot \mu(\langle \mathbf{c}, s' \rangle) \cdot \big(\mathbf{w} \cdot \mathit{satObj}_\Phi(\mathbf{g}, \mathbf{g}') + \mathbb{E}^{\max}_{M^+_{unf}}(\mathbf{w})[\langle s, \mathbf{e}', \mathbf{g} \rangle] \big) \Bigg) \\
&= \max_{\mu \in T(s)} \Bigg(\sum_{\langle \mathbf{c}, s' \rangle} [\mathbf{e} = \mathbf{e}'] \cdot \mu(\langle \mathbf{c}, s' \rangle) \cdot \big(\mathbf{w} \cdot \mathit{satObj}_\Phi(\mathbf{g}, \mathbf{g}') + \mathbf{x}[\langle s', \mathbf{e}', \mathbf{g}' \rangle] \big) \\
&\quad + \sum_{\langle \mathbf{c}, s' \rangle} [\mathbf{e} \neq \mathbf{e}'] \cdot \mu(\langle \mathbf{c}, s' \rangle) \cdot \sum_{\bar{\mathbf{g}} \in \mathbf{G}_m} [\mathbf{g}' = \bar{\mathbf{g}}] \cdot \big(\mathbf{w} \cdot \mathit{satObj}_\Phi(\mathbf{g}, \bar{\mathbf{g}}) \big) \\
&\quad + \sum_{\langle \mathbf{c}, s' \rangle} [\mathbf{e} \neq \mathbf{e}'] \cdot \mu(\langle \mathbf{c}, s' \rangle) \cdot \big(\mathbf{w} \cdot f(\mathbf{g}, \mu) \big) \cdot \sum_{\bar{\mathbf{g}} \in \mathbf{G}_m} [\mathbf{g}' = \bar{\mathbf{g}}] \Bigg) \\
&= \max_{\mu \in T(s)} \Bigg(\sum_{\langle \mathbf{c}, s' \rangle} [\mathbf{e} = \mathbf{e}'] \cdot \mu(\langle \mathbf{c}, s' \rangle) \cdot \big(\mathbf{w} \cdot \mathit{satObj}_\Phi(\mathbf{g}, \mathbf{g}') + \mathbf{x}[\langle s', \mathbf{e}', \mathbf{g}' \rangle] \big) \\
&\quad \sum_{\bar{\mathbf{g}} \in \mathbf{G}_m} \mathbf{w} \cdot \big(\mathit{satObj}_\Phi(\mathbf{g}, \bar{\mathbf{g}}) + f(\mathbf{g}, \mu) \big) \cdot \sum_{\langle \mathbf{c}, s' \rangle} [\mathbf{e} \neq \mathbf{e}'] \cdot \mu(\langle \mathbf{c}, s' \rangle) \cdot [\mathbf{g}' = \bar{\mathbf{g}}] \Bigg) \\
&= \max_{\nu \in T^{\mathbf{e}}_f(s)} \Bigg(\sum_{\langle \mathbf{c}, \langle s', \mathbf{g}' \rangle \rangle} [s' \neq s_\perp] \cdot \nu(\langle \mathbf{c}, \langle s', \mathbf{g}' \rangle \rangle) \cdot \big(\mathbf{w} \cdot \mathbf{c} + \mathbf{x}[\langle s', \mathbf{e}, \mathbf{g}' \rangle] \big) \\
&\quad + \sum_{\langle \mathbf{c}, \langle s_\perp, \mathbf{g}' \rangle \rangle} \nu(\mathbf{c}, \langle s_\perp, \mathbf{g}' \rangle) \cdot \mathbf{w} \cdot \mathbf{c} \Bigg) \\
&= \max_{\nu \in T^{\mathbf{e}}_f(s)} \Bigg(\sum_{\langle \mathbf{c}, \langle s', \mathbf{g}' \rangle \rangle} \nu(\langle \mathbf{c}, \langle s', \mathbf{g}' \rangle \rangle) \cdot \big(\mathbf{w} \cdot \mathbf{c} + [s' \neq s_\perp] \cdot \mathbf{x}[\langle s', \mathbf{e}, \mathbf{g}' \rangle] \big) \Bigg) \\
&= \max_{\nu \in T^{\mathbf{e}}_f(s)} \Bigg(\sum_{\langle \mathbf{c}, \langle s', \mathbf{g}' \rangle \rangle} \nu(\langle \mathbf{c}, \langle s', \mathbf{g}' \rangle \rangle) \cdot \big(\mathbf{w} \cdot \mathbf{c} + \mathbf{y}^{\mathbf{e}}[\langle s', \mathbf{g}' \rangle] \big) \Bigg).
\end{aligned}
$$

We conclude that $\mathbf{y}^{\mathbf{e}}[\langle s, \mathbf{g} \rangle] = [s \neq s_\perp] \cdot \mathbf{x}[\langle s, \mathbf{e}, \mathbf{g} \rangle]$ is indeed a solution for Eq. 5. If there is a smaller solution $\hat{\mathbf{y}}^{\mathbf{e}}[\langle s, \mathbf{g} \rangle] < \mathbf{y}^{\mathbf{e}}[\langle s, \mathbf{g} \rangle]$, the equalities above can be used to construct a smaller solution for Eq. 4, violating our assumption for $\mathbf{x}[\langle s, \mathbf{e}, \mathbf{g} \rangle]$. □

To analyse an epoch model $M^{\mathbf{e}}_f$, any successor epoch $\mathbf{e}'$ of $\mathbf{e}$ needs to have been analysed before. Since costs are non-negative, we can ensure this by analysing the epochs in a specific order: In the case of a single cost bound, this order is uniquely given by $\perp, 0, 1, \ldots, b$.

Definition 11 Let $\preceq \subseteq \mathbf{E}_m \times \mathbf{E}_m$ be the partial order with

$$\mathbf{e}' \preceq \mathbf{e} \quad \text{iff} \quad \forall i : \mathbf{e}'[i] \leq \mathbf{e}[i] \vee \mathbf{e}'[i] = \perp.$$

A *proper epoch sequence* is a sequence of epochs $\mathfrak{E} = \mathbf{e}_1 \ldots, \mathbf{e}_n$ such that (i) $\mathbf{e}_1 \leqslant \mathbf{e}_2 \leqslant \ldots \leqslant \mathbf{e}_n$ for some linearisation $\leqslant$ of $\preceq$ and (ii) if $\mathbf{e}$ occurs in $\mathfrak{E}$ and $\mathbf{e}' \preceq \mathbf{e}$, then also $\mathbf{e}'$ occurs in $\mathfrak{E}$.

For multiple cost bounds any proper epoch sequence can be considered. This definition coincides with the topological sort of the graph in Fig. 6. To improve performance, we group the epoch models with a common MDP structure.

Input : MDP $M = \langle S, T, s_{init} \rangle$, tradeoff $\Phi = multi\big(\mathbb{P}^{\max}_M(\varphi_1), \ldots, \mathbb{P}^{\max}_M(\varphi_\ell)\big)$ with cost limits $b_1, \ldots, b_m$, weight vector $\mathbf{w} \in [0,1]^\ell$ and proper epoch sequence $\mathfrak{E}$ ending with $\text{last}(\mathfrak{E}) = \langle b_1, \ldots, b_m \rangle$
Output : Point $\mathbf{p_w} \in \mathbb{R}^\ell$ satisfying Equation 3

1 **foreach** $\mathbf{e} \in \mathfrak{E}$ in ascending order **do**
2 **foreach** $\mathbf{g} \in \mathbf{G}_m$, $\mu \in \{\nu \mid \exists s\colon \nu \in T(s)\}$ **do**
3 $\mathbf{z} \leftarrow \mathbf{0}$; $\mu_{\text{exit}} \leftarrow 0$
4 **foreach** $\langle \mathbf{c}, s' \rangle \in \text{support}(\mu)$ **do**
5 $\mathbf{e}' \leftarrow succ(\mathbf{e}, \mathbf{c})$; $\mathbf{g}' \leftarrow succ(\mathbf{g}, s', \mathbf{e}')$
6 **if** $\mathbf{e}' \neq \mathbf{e}$ **then**
7 $\mathbf{z} \leftarrow \mathbf{z} + \mu(\mathbf{c}, s') \cdot x^{\mathbf{e}'}[\langle s', \mathbf{g}' \rangle]$
8 $\mu_{\text{exit}} \leftarrow \mu_{\text{exit}} + \mu(\mathbf{c}, s')$
9 **if** $\mu_{\text{exit}} > 0$ **then**
10 $f(\mathbf{g}, \mu) \leftarrow \mathbf{z}/\mu_{\text{exit}}$
11 **else**
12 $f(\mathbf{g}, \mu) \leftarrow \mathbf{0}$
13 build epoch model $M^{\mathbf{e}}_f = \langle S^{\mathbf{e}}, T^{\mathbf{e}}_f, s^{\mathbf{e}}_{init} \rangle$
14 $\mathfrak{S} \leftarrow \arg\max_{\mathfrak{S}'} \mathbb{E}^{\mathfrak{S}'}_{M^{\mathbf{e}}_f}(\mathbf{w})$
15 **foreach** $k \in \{1, \ldots, \ell\}$, $\tilde{s} \in S^{\mathbf{e}}$ **do**
16 $x^{\mathbf{e}}[\tilde{s}][k] \leftarrow \mathbb{E}^{\mathfrak{S}}_{M^{\mathbf{e}}_f}(\mathbf{1}_k)[\tilde{s}]$
17 **return** $x^{\text{last}(\mathfrak{E})}[s^{\text{last}(\mathfrak{E})}_{init}]$

Algorithm 1: Sequential multi-cost bounded analysis

Example 11 For the epoch models depicted in Fig. 5, a possible proper epoch sequence is

$$\mathfrak{E} = \langle \bot, \bot \rangle, \langle 0, \bot \rangle, \langle 2, \bot \rangle, \langle \bot, 1 \rangle, \langle \bot, 3 \rangle, \langle 1, 1 \rangle, \langle 0, 3 \rangle, \langle 3, 1 \rangle, \langle 2, 3 \rangle, \langle 4, 3 \rangle.$$

We compute the points $\mathbf{p_w}$ by analysing the different epoch models (i.e. the coordinates of Fig. 3b) sequentially, using a dynamic programming-based approach. The main procedure is outlined in Algorithm 1. The costs of the model for the current epoch $\mathbf{e}$ are computed in lines 2-12. These costs comprise the results from previously analysed epochs $\mathbf{e}'$ (line 7). In lines 13-16, the current epoch model $M^{\mathbf{e}}_f$ is built and analysed: We compute weighted expected costs on $M^{\mathbf{e}}_f$ where $\mathbb{E}^{\mathfrak{S}}_{M^{\mathbf{e}}_f}(\mathbf{w})[s]$ denotes the expected costs for $M^{\mathbf{e}}_f$ when changing the initial state to s. In line 14, a (deterministic and memoryless) scheduler $\mathfrak{S}$ that induces the maximal weighted expected costs (i.e. $\mathbb{E}^{\mathfrak{S}}_{M^{\mathbf{e}}_f}(\mathbf{w})[s] = \max_{\mathfrak{S}'} \mathbb{E}^{\mathfrak{S}'}_{M^{\mathbf{e}}_f}(\mathbf{w})[s]$ for all states s) is computed. In line 16, we then compute the expected costs induced by $\mathfrak{S}$ for the individual objectives. Forejt et al. [25] describe how this computation can be implemented with a value iteration-based procedure. Alternatively, we can apply policy iteration or linear programming [46] for this purpose.

Theorem 1 *The output of Algorithm 1 satisfies Eq. 3.*

Proof We have to show:

$$x^{\text{last}(\mathfrak{E})}[s^{\text{last}(\mathfrak{E})}_{init}] = \langle \mathbb{P}^{\mathfrak{S}}_M(\varphi_1), \ldots, \mathbb{P}^{\mathfrak{S}}_M(\varphi_\ell) \rangle \text{ with } \mathfrak{S} \in \arg\max_{\mathfrak{S}'} \left(\sum_{k=1}^{\ell} w_i \cdot \mathbb{P}^{\mathfrak{S}'}_M(\varphi_k) \right)$$

We prove the following statement for each epoch $\mathbf{e}$:

$$x^{\mathbf{e}}[\langle s, \mathbf{g}\rangle] = \langle \mathbb{P}_M^{\mathfrak{S}}(\varphi_1'), \ldots, \mathbb{P}_M^{\mathfrak{S}}(\varphi_\ell')\rangle \text{ with } \mathfrak{S} \in \arg\max_{\mathfrak{S}'} \left(\sum_{k=1}^{\ell} w_i \cdot \mathbb{P}_M^{\mathfrak{S}'}(\varphi_k') \right)$$

where

$$\varphi_k' = \bigwedge_{i=n_{k-1}}^{n_k-1} (\langle \mathrm{C}_i \rangle_{\leq \mathbf{e}[i]}\, G_i) \text{ using } \varphi_k = \bigwedge_{i=n_{k-1}}^{n_k-1} (\langle \mathrm{C}_i \rangle_{\leq b_i}\, G_i)$$

i.e. φ_k' is obtained from φ_k by adapting the cost limits based on the current epoch. For $\mathbf{e}[i] = \bot$ we assume that the cost bound $\langle \mathrm{C}_i \rangle_{\leq \bot}\, G_i$ is not satisfied by any path.

Thus, the algorithm correctly computes the bounded reachability for *all* states and *all* epochs. This statement is now proven by induction over any proper epoch sequence. For the induction base, the algorithm correctly computes the epoch $\langle \bot, \ldots, \bot\rangle$. In particular, notice that there exists an optimal memoryless scheduler on the unfolding, and thus a memoryless scheduler on the epoch model. For the induction step, let $\mathbf{e}$ be the currently analysed epoch. Since $\mathfrak{E}$ is assumed to be a *proper* epoch sequence, we already computed any reachable successor epoch $\mathbf{e}'$ of $\mathbf{e}$, i.e. line 7 is only executed for epochs $\mathbf{e}'$ for which $x^{\mathbf{e}'}$ has already been computed, and by the induction hypothesis these $x^{\mathbf{e}'}[\langle s, \mathbf{g}\rangle][k]$ computed by the algorithm coincide with the probability to satisfy φ_k' from state $\langle s, \mathbf{e}', \mathbf{g}\rangle$ in the unfolding M_{unf} under a scheduler $\mathfrak{S}$ that maximises the weighted sum. Hence, the algorithm computes the function f as given in Lemma 2. Then, the algorithm computes weighted expected costs for the epoch model and writes them into $x^{\mathbf{e}}[\langle s, \mathbf{g}\rangle][k]$. By Lemma 2, these values coincide with the unfolding. □

4.2 Runtime and Memory Requirements

In the following, we discuss the complexity of our approach relative to the size of a binary encoding of the cost limits $b_1, \ldots, b_m$ occurring in a tradeoff Φ. Algorithm 1 computes expected weighted costs for $|\mathfrak{E}|$ many epoch models $M_f^{\mathbf{e}}$. Each of these computations can be done in polynomial time (in the size of $M_f^{\mathbf{e}}$) via a linear programming encoding [46]. With $|\mathfrak{E}| \leq \prod_{i=1}^{m} b_i$, we conclude that the runtime of Algorithm 1 is exponential in a binary encoding of the cost limits. For the unfolding approach, weighted expected costs have to be computed for a single MDP whose size is, again, exponential in a binary encoding of the cost limits. Although we observe similar theoretical runtime complexities for both approaches, experiments with topological value iteration [5,19] and single cost bounds [2,28] have shown the practical benefits of analysing several small sub-models instead of one large MDP. We make similar observations with our approach in Sect. 7.

Algorithm 1 stores a solution vector $x^{\mathbf{e}}[\langle s, \mathbf{g}\rangle] \in \mathbb{R}^{\ell}$ for each $\mathbf{e} \in \mathfrak{E}, s \in S$, and $\mathbf{g} \in \mathbf{G}_m$, i.e. a solution vector is stored for every state of the unfolding. However, memory consumption can be optimised by erasing solutions $x^{\mathbf{e}}[\langle s, \mathbf{g}\rangle]$ as soon as this value is not accessed by any of the remaining epoch models (for example if all predecessor epochs of $\mathbf{e}$ have been considered already). If $m = 1$ (i.e. there is only a single cost bound), such an optimisation yields an algorithm that runs in polynomial space. In the general case ($m > 1$), the memory requirements remain exponential in the size of a binary encoding of the cost limits. However, our experiments in Sect. 7 indicate substantial memory savings in practice.

4.3 Error Propagation

As presented above, the algorithm assumes that (weighted) expected costs $\mathbb{E}_M^{\mathfrak{S}}(\mathbf{w})$ are computed exactly. Practical implementations, however, are often based on numerical methods that only approximate the correct solution. The de-facto standard in MDP model checking for this purpose is *value iteration*. Methods based on value iteration do not provide any guarantee on the accuracy of the obtained result [27] for the properties considered here. Recently, interval iteration [5,27] and similar techniques [9,34,48] have been suggested to provide error bounds. These methods guarantee that the obtained result x_s is ε-precise for any predefined precision $\varepsilon > 0$, i.e. upon termination we obtain $|x[s] - \mathbb{E}_M^{\mathfrak{S}}(\mathbf{w})[s]| \leq \varepsilon$ for all states s. We describe how to adapt our approach for multi-objective multi-cost bounded reachability to work with an ε-precise method for computing the expected costs.

4.3.1 General Models

Results from topological interval iteration [5] indicate that individual epochs can be analysed with precision ε to guarantee this same precision for the overall result. The downside is that such an adaptation requires the storage of the obtained bounds for all previously analysed epochs. Therefore, we extend the following result from [28].

Lemma 3 *For the single-cost bounded variant of Algorithm 1, to compute* $\mathbb{P}_M^{\max}(\langle \mathrm{C} \rangle_{\leq b}\, G)$ *with precision* ε*, each epoch model needs to be analysed with precision* $\frac{\varepsilon}{b+1}$.

The bound is easily deduced: assume the results of previously analysed epochs (given by f) are η-precise and that $M_f^{\mathbf{e}}$ is analysed with precision δ. The total error for $M_f^{\mathbf{e}}$ can accumulate to at most $\delta + \eta$. As we analyse $b + 1$ (non-trivial) epoch models, the error thus accumulates to $(b + 1) \cdot \delta$. Setting δ to $\frac{\varepsilon}{b+1}$ guarantees the desired bound ε. We generalise this result to multi-cost bounded tradeoffs.

Theorem 2 *If the values* $x^{\mathbf{e}}[\tilde{s}][k]$ *at line 16 of Algorithm 1 are computed with precision* $\varepsilon / \sum_{i=1}^{m}(b_i + 1)$ *for some* $\varepsilon > 0$*, the output* $\mathbf{p}'_{\mathbf{w}}$ *of the algorithm satisfies* $|\mathbf{p}_{\mathbf{w}} - \mathbf{p}'_{\mathbf{w}}| \cdot \mathbf{w} \leq \varepsilon$ *where* $\mathbf{p}_{\mathbf{w}}$ *is as in Eq. 3.*

Proof As in the single-cost bounded case, the total error for $M_f^{\mathbf{e}}$ can accumulate to $\delta + \eta$ when η is the (maximal) error bound on f. The error bound on f is again recursively determined by $\delta - 1$ times the maximum number of epochs visited along paths from the successor epochs. Since a path through the MDP M visits at most $\sum_{i=1}^{m}(b_i + 1)$ non-trivial cost epochs, each incurring cost δ, the overall error can be upper-bounded by $\delta \cdot \sum_{i=1}^{m}(b_i + 1)$. □

While an approach based on Theorem 2 thus requires the analysis of epoch models with tighter error bounds than the bounds induced by [5], and therefore potentially increases the per-epoch analysis time, it still allows us to be significantly more memory-efficient.

4.3.2 Acyclic Epoch Models

The error bound in Theorem 2 is pessimistic, as it does not assume any structure in the epoch models. However, very often, the individual epoch models are in fact acyclic, in particular for cost epochs $\mathbf{e} \in \mathbb{N}^m$, i.e. $\mathbf{e}[i] \neq \bot$ for all i. Intuitively, costs usually represent quantities like time or energy usage for which the possibility to perform infinitely many interesting steps without accumulating cost would be considered a modelling error. In the timed case,

Fig. 7 Example MDP for multi-bounded single-goal queries

for example, such a model would allow Zeno behaviour, which is generally considered unrealistic and undesirable. When epoch models are acyclic, interval iteration [5,27] will converge to the exact result in a finite number of iterations. In this case, the tightening of the precision according to Theorem 2 usually has no effect on runtime. The epoch models for cost epochs $\mathbf{e} \in \mathbb{N}^m$ are acyclic for almost all models that we experiment with in Sect. 7.

4.4 Different Bound Types

Minimising Objectives Objectives $\mathbb{P}^{\min}_M(\varphi_k)$ can be handled by adapting the function $satObj_\Phi$ in Definition 10 such that it assigns cost -1 to branches that lead to the satisfaction of φ_k. To obtain the desired probabilities we then maximise negative costs and multiply the result by -1 afterwards. As interval iteration supports mixtures of positive and negative costs [5], arbitrary combinations of minimising and maximising objectives can be considered[1].

Beyond Upper Bounds Our approach also supports bounds of the form $\langle \mathrm{C}_j \rangle_{\sim b}\, G$ for $\sim \in \{<, \leq, >, \geq\}$, and we allow *combinations* of lower and upper cost bounds. Strict upper bounds $< b$ can be reformulated to non-strict upper bounds $\leq b - 1$. Likewise, we reformulate non-strict lower bounds $\geq b$ to $> b - 1$, and only consider strict lower bounds in the following.

For bound $\langle \mathrm{C}_i \rangle_{>b_i}\, G_i$ we adapt the update of goal satisfactions (Definition 7) such that

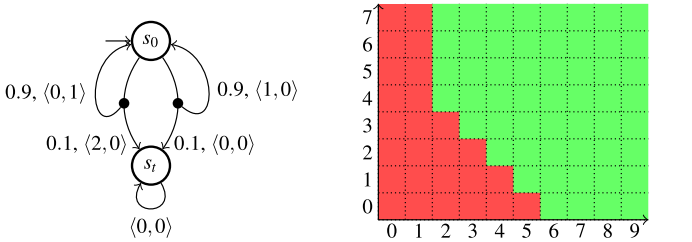
$$succ(\mathbf{g}, s, \mathbf{e})[i] = \begin{cases} 1 & \text{if } s \in G_i \wedge \mathbf{e}[i] = \bot, \\ \mathbf{g}[i] & \text{otherwise.} \end{cases}$$

Moreover, we support multi-bounded single-goal queries like $\langle \mathrm{C}_{(j_1,\ldots,j_n)} \rangle_{(\sim_1 b_1,\ldots,\sim_n b_n)}\, G$, which characterises the paths π with a prefix π_{fin} satisfying $\text{last}(\pi_{\text{fin}}) \in G$ and *all* cost bounds *simultaneously*, i.e. $\text{cost}_{j_i}(\pi_{\text{fin}}) \sim_i b_i$. Let us clarify the meaning of simultaneously with an example.

Example 12 The formula $\varphi = \langle \mathrm{C}_{(1,1)} \rangle_{(\leq 1, \geq 1)}\, G$ expresses the paths that reach G while collecting exactly one cost with respect to the first cost structure. This formula is not equivalent to $\varphi' = \langle \mathrm{C}_1 \rangle_{\leq 1}\, G \wedge \langle \mathrm{C}_1 \rangle_{\geq 1}\, G$. Consider the trivial MDP in Fig. 7 with $G = \{ s_0 \}$. The MDP (and the trivial strategy) satisfies φ' but not φ: Initially, the left-hand side of φ' is (already) satisfied, and after one more step along the unique path, also the right-hand side is satisfied, thereby satisfying the conjunction. However, there is no point where exactly cost 1 is collected, hence φ is never satisfied.

Expected Cost Objectives The algorithm supports cost-bounded expected cost objectives $\mathbb{E}^{opt}_M(\mathrm{R}_{j_1}, \langle \mathrm{C}_{j_2} \rangle_{\leq b})$ with $opt \in \{\max, \min\}$, which refer to the expected cost accumulated for cost structure j_1 within a given cost bound $\langle \mathrm{C}_{j_2} \rangle_{\leq b}$. The computation is analogous to cost-bounded reachability queries: we treat them by computing (weighted) expected costs within epoch models. Therefore, they can be used in multi-objective queries, potentially in combination with cost-bounded reachability objectives.

[1] This supersedes a restriction of the algorithm of [25].



(a) MDP M_{Qu} **(b)** $Sat(\Psi_{ex})$ (green) and $Sat(\overline{\Psi_{ex}})$ (red)

Fig. 8 Example MDP M_{Qu} and satisfying cost limits

5 Multi-cost Quantiles

The queries presented in previous sections assume that cost limits are fixed a priori and ask for the induced probabilities. We now study the opposite question: What are the cost limits required to satisfy a given probability threshold? This question thus asks for computing *quantiles* as considered in [2,37,52,57], and we lift it to multiple cost bounds. In particular, we present an efficient implementation of an algorithm to answer questions like

- How much time and energy is required to fulfil a task with at least probability 0.8?
- How many product types can be manufactured without failure with probability 0.99?
- How much energy is needed to complete how many jobs with probability 0.9?

In this section, we introduce multi-cost bounded quantile queries to formalise these questions. We then first sketch our approach to solve them, and after that provide a more extensive treatment of quantiles with only upper cost bounds and of quantiles with only lower cost bounds. Finally, we address more complex forms of quantiles in Sect. 5.5.

5.1 Quantiles in Multiple Dimensions

Definition 12 An m-dimensional quantile query for an MDP M and $m \in \mathbb{N}$ is given by $Qu\left(\mathbb{P}_M^{opt}(\varphi_?) \sim p\right)$, with $opt \in \{\min, \max\}$, $\sim \in \{<, \leq, >, \geq\}$, a fixed probability threshold $p \in [0,1]$, and a cost-bounded reachability formula $\varphi_? = \bigwedge_{i=1}^{m \in \mathbb{N}} (\langle \mathrm{C}_{j_i} \rangle_{\sim_i ?}\, G_i)$ with unspecified (i.e. a priori unknown) cost limits.

The solution of a quantile query is a set of cost limits that satisfy the probability threshold.

Definition 13 The set of satisfying cost limits for an m-dimensional quantile query $\Psi = Qu\left(\mathbb{P}_M^{opt}(\varphi_?) \sim p\right)$ is given by

$$Sat(\Psi) = \{\, \mathbf{b} \in \mathbb{N}^m \mid \mathbb{P}_M^{opt}(\varphi_{\mathbf{b}}) \sim p \,\}$$

where $\varphi_{\mathbf{b}} = \bigwedge_{i=1}^{m \in \mathbb{N}} (\langle \mathrm{C}_{j_i} \rangle_{\sim_i \mathbf{b}[i]}\, G_i)$ arises from $\varphi_?$ by inserting cost limits $\mathbf{b}$.

Example 13 Consider the MDP M_{Qu} given in Fig. 8a and the quantile query

$$\Psi_{ex} = Qu\left(\mathbb{P}_{M_{Qu}}^{max}\ \left(\langle \mathrm{C}_1 \rangle_{\leq ?}\, \{ s_t \} \wedge \langle \mathrm{C}_2 \rangle_{\leq ?}\, \{ s_t \}\right)\ > 0.5\right).$$

The (upper-right, brighter) green area in Fig. 8b indicates the set of satisfying cost limits for Ψ_{ex}, given by

$$Sat(\Psi_{ex}) = \{\, \mathbf{c} \in \mathbb{N}^2 \mid \exists\, \mathbf{b} \in \{\, \langle 2,4\rangle, \langle 3,3\rangle, \langle 4,2\rangle, \langle 5,1\rangle, \langle 6,0\rangle \,\} \colon \mathbf{b}[1] \leq \mathbf{c}[1] \wedge \mathbf{b}[2] \leq \mathbf{c}[2] \,\}.$$

Concretely, the set describes a form of closure of a set of points on the frontier. We discuss why this is the satisfying set. First, consider cost limits $\langle 1, y\rangle$ for arbitrary y. The point indicates a limit of 1 in the first dimension. In particular, the leftmost action is then never helpful in satisfying the objective as it takes cost 2 in the first dimension. Thus, we have to take the right action. When taking this action, we may return to s_0 at most once before violating the cost limit. Thus, the probability to reach the target is $0.1 + 0.9 \cdot 0.1 < 0.5$, and these cost limits violate the query. Now, consider cost limits $\langle 6, 0\rangle$. Using similar reasoning as above, only the right action is relevant. We may take the self-loop at most 6 times, which yields a probability to reach the target within the cost limit of $\sum_{i=0}^{6} 0.1 \cdot 0.9^i > 0.5$, and thus these cost limits satisfy the query. Finally, consider cost limits $\langle 2, 4\rangle$. Now, the left action helps: We can take the left action at most 4 times. If we still have not reached the target, we have $\langle 2, 0\rangle$ cost remaining, which can be spend trying the right action up to 2 times. The probability of reaching the target under this scheduler is again $\sum_{i=0}^{6} 0.1 \cdot 0.9^i > 0.5$.

For the remainder, let us fix an m-dimensional quantile query $\Psi = Qu\left(\mathbb{P}_M^{opt}(\varphi_?) \sim p\right)$ with $\varphi_? = \bigwedge_{i=1}^{m \in \mathbb{N}} (\langle \mathrm{C}_{j_i} \rangle_{\sim_i ?}\, G_i)$. We write $\overline{\Psi}$ for the complementary query $Qu\left(\mathbb{P}_M^{opt}(\varphi_?) \nsim p\right)$ where the comparison operator is inverted (e.g. $\nsim\, = \leq$ if $\sim\, = >$). Observe that $Sat(\overline{\Psi}) = \mathbb{N}^m \setminus Sat(\Psi)$.

In Example 13, the infinite set of satisfying cost limits is concisely described as the closure of the finitely many points generating its "frontier". We lift this type of representation to general quantile queries.

Definition 14 The *closure* of a set $\mathscr{B} \subseteq (\mathbb{N} \cup \{\infty\})^m$ with respect to a quantile query Ψ is given by $cl^{\Psi}(\mathscr{B}) = \{\mathbf{c} \in (\mathbb{N} \cup \{\infty\})^m \mid \exists\, \mathbf{b} \in \mathscr{B} : \mathbf{b} \trianglelefteq \mathbf{c}\}$, where

$$\mathbf{b} \trianglelefteq \mathbf{c} \text{ iff } \forall i \in \{1, \ldots, m\} : \mathbf{b}[i] = \mathbf{c}[i] \text{ or } \begin{cases} \mathbf{b}[i] \sim_i \mathbf{c}[i] & \text{if } \sim \in \{>, \geq\} \\ \mathbf{c}[i] \sim_i \mathbf{b}[i] & \text{if } \sim \in \{<, \leq\}. \end{cases}$$

Indeed, we can always characterise the set of satisfying cost limits by $\mathscr{B} \subseteq Sat(\Psi)$ with $cl^{\Psi}(\mathscr{B}) = cl^{\Psi}(Sat(\Psi))$.

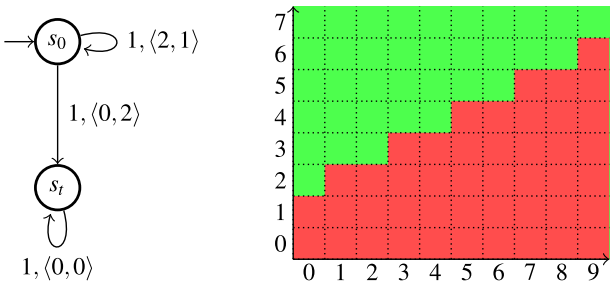
Lemma 4 $Sat(\Psi) = cl^{\Psi}(Sat(\Psi)) \cap \mathbb{N}^m$.

Proof The probability $\mathbb{P}_M^{opt}(\varphi_{\mathbf{b}})$ is monotonic in the cost limits $\mathbf{b}$. More precisely, increasing cost limit $\mathbf{b}[i]$ for $i \in \{1, \ldots, m\}$ increases the probability if $\sim_i \in \{<, \leq\}$ and decreases it otherwise. It follows that $\mathbf{b} \in Sat(\Psi) \wedge \mathbf{b} \trianglelefteq \mathbf{c}$ implies $\mathbf{c} \in Sat(\Psi)$ for $\mathbf{c} \in \mathbb{N}^m$. The lemma follows by the definition of closure. □

The smallest set whose closure is $Sat(\Psi)$ is called the *generator* of $Sat(\Psi)$.

Definition 15 The *generator* $gen^{\Psi}(\mathscr{B})$ of $\mathscr{B} \subseteq (\mathbb{N} \cup \{\infty\})^m$ is the smallest set $\mathscr{G}$ such that $cl^{\Psi}(\mathscr{G}) = cl^{\Psi}(\mathscr{B})$ and $cl^{\Psi}(\mathscr{G}') \neq cl^{\Psi}(\mathscr{B})$ for every proper subset $\mathscr{G}' \subsetneq \mathscr{G}$.

Lemma 5 *The generator* $gen^{\Psi}(\mathscr{B})$ *of any* $\mathscr{B} \subseteq (\mathbb{N} \cup \{\infty\})^m$ *is unique.*



(a) MDP M'_{Qu} **(b)** $Sat(\Psi'_{ex})$

Fig. 9 Example MDP M'_{Qu} and satisfying cost limits for Ψ_{ex}

Proof For the sake of contradiction, assume $\mathscr{B} \subseteq (\mathbb{N} \cup \{\infty\})^m$ has two generators $\mathscr{G}_1, \mathscr{G}_2$ with $\mathscr{G}_1 \neq \mathscr{G}_2$. According to Definition 15, $\mathscr{G}_1$ cannot be a subset of the generator $\mathscr{G}_2$ since $cl^{\Psi}(\mathscr{G}_1) = cl^{\Psi}(\mathscr{B}) = cl^{\Psi}(\mathscr{G}_2)$. Let $\mathbf{b}_1 \in \mathscr{G}_1 \setminus \mathscr{G}_2$. We have $\mathbf{b}_1 \in cl^{\Psi}(\mathscr{G}_2)$, thus there is $\mathbf{b}_2 \in \mathscr{G}_2$ with $\mathbf{b}_2 \trianglelefteq \mathbf{b}_1$, where $\trianglelefteq$ is as in Definition 14. Similarly, $\mathbf{b}_2 \in cl^{\Psi}(\mathscr{G}_1)$ implies $\mathbf{b}_3 \trianglelefteq \mathbf{b}_2$ for some $\mathbf{b}_3 \in \mathscr{G}_1$. Let $\mathbf{b} \in cl^{\Psi}(\mathscr{G}_1)$ with $\mathbf{b}_1 \trianglelefteq \mathbf{b}$. Transitivity of $\trianglelefteq$ yields $\mathbf{b}_3 \trianglelefteq \mathbf{b}$, i.e. $\mathbf{b} \in cl^{\Psi}(\mathscr{G}_1 \setminus \{\mathbf{b}_1\})$. It follows that $cl^{\Psi}(\mathscr{G}_1 \setminus \{\mathbf{b}_1\}) = cl^{\Psi}(\mathscr{G}_1)$ for the proper subset $\mathscr{G}_1 \setminus \{\mathbf{b}_1\}$ of $\mathscr{G}_1$. This contradicts our assumption that $\mathscr{G}_1$ is a generator of $\mathscr{B}$. □

We also refer to $gen^{\Psi}(Sat(\Psi))$ as the generator of quantile query Ψ. A generator is called *natural* if it only contains points in $\mathbb{N}^m$. The following example shows that quantile queries can have (non-)natural and (in-)finite generators.

Example 14 Ψ_{ex} from Example 13 has a finite natural generator:

$$gen^{\Psi_{ex}}(Sat(\Psi_{ex})) = \{\langle 2, 4\rangle, \langle 3, 3\rangle, \langle 4, 2\rangle, \langle 5, 1\rangle, \langle 6, 0\rangle\}$$

The generator of the complementary query $\overline{\Psi_{ex}}$ is still finite but not natural:

$$gen^{\overline{\Psi_{ex}}}\left(Sat(\overline{\Psi_{ex}})\right) = \{\langle 1, \infty\rangle, \langle 2, 3\rangle, \langle 3, 2\rangle, \langle 4, 1\rangle, \langle 5, 0\rangle\}.$$

The MDP M'_{Qu} from Fig. 9a and the quantile query

$$\Psi'_{ex} = Qu\left(\mathbb{P}^{max}_{M'_{Qu}}\left(\langle \mathrm{C}_1\rangle_{\geq ?}\{s_t\} \wedge \langle \mathrm{C}_2\rangle_{\leq ?}\{s_t\}\right) > 0.5\right).$$

yield the satisfying cost limits as shown in Figure 9b. Ψ'_{ex} does not have a finite generator:

$$gen^{\Psi'_{ex}}\left(Sat(\Psi'_{ex})\right) = \{\langle 2 \cdot n, 2 + n\rangle \mid n \in \mathbb{N}\}.$$

5.2 Computing Finite Natural Generators

We now present an algorithm to compute the set of satisfying cost limits $Sat(\Psi)$ for quantile queries Ψ where both Ψ and $\overline{\Psi}$ have a finite natural generator. In the subsequent subsections, we present suitable preprocessing steps to lift this algorithm to more general quantiles.

Our approach is sketched in Algorithm 2. Similarly to Algorithm 1 it analyses epoch models successively. However the sequence of analysed cost epochs is extended in an on-the-fly manner by considering more and more candidate epochs with an increasing maximum component $\max_i(\mathbf{e}_{cand}[i]) = b$. Whenever the algorithm finds an epoch $\mathbf{e}$ that is a valid cost

Input : MDP $M = \langle S, T, s_{init} \rangle$, m-dimensional quantile $\Psi = Qu\left(\mathbb{P}_M^{opt}(\varphi_?) \sim p\right)$
Output : A finite natural generator of Ψ

1 $\mathscr{B}^+ \leftarrow \emptyset$; $\mathscr{B}^- \leftarrow \emptyset$
2 $\mathfrak{E}_{checked} \leftarrow \emptyset$; $b \leftarrow 0$; *progress* $\leftarrow$ *TRUE*
3 **while** *progress* **do**
4 *progress* $\leftarrow$ *FALSE*
5 **foreach** $\mathbf{e}_{cand} \in \mathbb{N}^m$ with $\max_i(\mathbf{e}_{cand}[i]) = b$ **do**
6 **if** $\mathbf{e}_{cand} \notin cl^{\Psi}\left(\mathscr{B}^+\right)$ and $\mathbf{e}_{cand} \notin cl^{\overline{\Psi}}\left(\mathscr{B}^-\right)$ **then**
7 *progress* $\leftarrow$ *TRUE*
8 $\mathfrak{E} \leftarrow$ proper epoch sequence with $\text{last}(\mathfrak{E}) = \mathbf{e}_{cand}$
9 **foreach** $\mathbf{e} \in \mathfrak{E}$ in ascending order **do**
10 **if** $\mathbf{e} \notin \mathfrak{E}_{checked}$ **then**
11 $\mathfrak{E}_{checked} \leftarrow \mathfrak{E}_{checked} \cup \{\mathbf{e}\}$
 // Check epoch model as in Algorithm 1 for $\ell = 1$ objectives:
12 **foreach** $\mathbf{g} \in \mathbf{G}_m$, $\mu \in \{\nu \mid \exists s\colon \nu \in T(s)\}$ **do**
13 $\mathbf{z} \leftarrow 0$; $\mu_{\text{exit}} \leftarrow 0$
14 **foreach** $\langle \mathbf{c}, s' \rangle \in \text{support}(\mu)$ **do**
15 $\mathbf{e}' \leftarrow succ(\mathbf{e}, \mathbf{c})$; $\mathbf{g}' \leftarrow succ(\mathbf{g}, s', \mathbf{e}')$
16 **if** $\mathbf{e}' \neq \mathbf{e}$ **then**
17 $\mathbf{z} \leftarrow \mathbf{z} + \mu(\mathbf{c}, s') \cdot x^{\mathbf{e}'}[\langle s', \mathbf{g}' \rangle]$
18 $\mu_{\text{exit}} \leftarrow \mu_{\text{exit}} + \mu(\mathbf{c}, s')$
19 **if** $\mu_{\text{exit}} > 0$ **then**
20 $f(\mathbf{g}, \mu) \leftarrow \mathbf{z}/\mu_{\text{exit}}$
21 **else**
22 $f(\mathbf{g}, \mu) \leftarrow \mathbf{0}$
23 build epoch model $M_f^{\mathbf{e}} = \langle S^{\mathbf{e}}, T_f^{\mathbf{e}}, s_{init}^{\mathbf{e}} \rangle$ with a single cost structure
24 **foreach** $\tilde{s} \in S^{\mathbf{e}}$ **do**
25 $x^{\mathbf{e}}[\tilde{s}] \leftarrow \mathbb{E}^{opt}_{M_f^{\mathbf{e}}}(\langle 1 \rangle)[\tilde{s}]$ *// weighted expected costs over weight vector $\langle 1 \rangle$*
26 **if** $\mathbf{e}[i] \neq \perp$ for all i **then**
27 **if** $x^{\mathbf{e}}[s_{init}^{\mathbf{e}}] \sim p$ **then**
28 $\mathscr{B}^+ \leftarrow \mathscr{B}^+ \cup \{\mathbf{e}\}$
29 **else**
30 $\mathscr{B}^- \leftarrow \mathscr{B}^- \cup \{\mathbf{e}\}$
31 $b \leftarrow b + 1$
32 **return** $gen^{\Psi}\left(\mathscr{B}^+\right)$

Algorithm 2: Multi-dimensional quantile computation.

limit (i.e. $\mathbf{e} \in \mathbb{N}^m$), the epoch is added to either $\mathscr{B}^+$ or $\mathscr{B}^-$, depending on whether the probability threshold is satisfied in $\mathbf{e}$ or not (lines 26 to 30). In this way, $\mathscr{B}^+$ and $\mathscr{B}^-$ gather satisfying cost limits for the quantile query Ψ and its complementary query $\overline{\Psi}$, respectively. Due to the condition in line 10, we never analyse the same epoch model twice. The condition in line 6 ensures that we do not analyse epochs which are already known to be satisfying for either Ψ or $\overline{\Psi}$. The procedure stops as soon as we find a $b \in \mathbb{N}$ such that all new candidate epochs with maximum entry b are already in $cl^{\Psi}\left(\mathscr{B}^+\right) \cup cl^{\overline{\Psi}}(\mathscr{B}^-)$.

Lemma 6 *During the execution of Algorithm 2 the following invariant holds:*

$$cl^{\Psi}\left(\mathscr{B}^+\right) \subseteq cl^{\Psi}\left(Sat(\Psi)\right) \quad \textit{and} \quad cl^{\overline{\Psi}}\left(\mathscr{B}^-\right) \subseteq cl^{\overline{\Psi}}\left(Sat(\overline{\Psi})\right).$$

Proof As in the proof of Theorem 1, it can be shown that in lines 11–25, the algorithm computes $\mathbb{P}_M^{opt}(\varphi_{\mathbf{e}})$ where $\varphi_{\mathbf{e}}$ arises from $\varphi_?$ by inserting the cost limits from $\mathbf{e}$. The algorithm checks whether this probability meets the threshold given in Ψ and inserts it in either $\mathscr{B}^+$ or $\mathscr{B}^-$, accordingly. Hence, $\mathscr{B}^+$ ($\mathscr{B}^-$) only contains (non-)satisfying cost limits.

Lemma 7 *If Algorithm 2 terminates, it returns a generator of* Ψ.

Proof We first show that upon termination, $\mathbb{N}^m \subseteq cl^{\Psi}\left(\mathscr{B}^+\right) \cup cl^{\overline{\Psi}}\left(\mathscr{B}^-\right)$. The proof is based on the following observation: When the algorithm terminates, there is a $b \in \mathbb{N}$ such that

$$\max_i(\mathbf{b}[i]) = b \text{ implies } \mathbf{b} \in cl^{\Psi}\left(\mathscr{B}^+\right) \cup cl^{\overline{\Psi}}\left(\mathscr{B}^-\right) \text{ and } \mathbf{b} \notin \mathscr{B}^+ \cup \mathscr{B}^- \text{ for any epoch } \mathbf{b}.$$

In particular, for $\mathbf{b} \in \mathscr{B}^+ \cup \mathscr{B}^-$ we have $\max_i(\mathbf{b}[i]) < b$. Consider the smallest such b (i.e. the value of the variable b upon termination). Assume $\mathbf{b} \in \mathbb{N}^m$. If $\max_i(\mathbf{b}[i]) < b$, $\mathbf{b}$ has already been considered as a candidate epoch in line 6, therefore $\mathbf{b} \in cl^{\Psi}\left(\mathscr{B}^+\right) \cup cl^{\overline{\Psi}}\left(\mathscr{B}^-\right)$. Otherwise, $\max_i(\mathbf{b}[i]) \geq b$. Let $\mathbf{b}' \in \mathbb{N}^m$ be given such that $\mathbf{b}'[i] = \min(\mathbf{b}[i], b)$ for all i. Since $\max_i(\mathbf{b}'[i]) = b$, we have $\mathbf{b}' \in cl^{\Psi}\left(\mathscr{B}^+\right) \cup cl^{\overline{\Psi}}\left(\mathscr{B}^-\right)$. Assume $\mathbf{b}' \in cl^{\Psi}\left(\mathscr{B}^+\right)$ (the case where $\mathbf{b}' \in cl^{\overline{\Psi}}\left(\mathscr{B}^-\right)$ is completely analogous). According to the definition of closure, there has to be a $\mathbf{b}'' \in \mathscr{B}^+$ with $\mathbf{b}'' \trianglelefteq \mathbf{b}'$, where $\trianglelefteq$ is as in Definition 14. We also have $\mathbf{b}' \trianglelefteq \mathbf{b}$ since for all $i \in \{1, \ldots, m\}$ one of the following cases holds:

- If $\mathbf{b}[i] < b$, we have $\mathbf{b}[i] = \mathbf{b}'[i]$.
- If $\mathbf{b}[i] \geq b$, we have $\mathbf{b}'[i] = b$ and $b > \mathbf{b}''[i]$. The latter follows from $\mathbf{b}'' \in \mathscr{B}^+$, yielding $\max_i(\mathbf{b}''[i]) < b$. As $\mathbf{b}'' \trianglelefteq \mathbf{b}'$ and $\mathbf{b}''[i] < \mathbf{b}'[i]$, we have to be in a case where either ($\sim_i \in \{<, \leq\}$ and $\sim \in \{>, \geq\}$) or ($\sim_i \in \{>, \geq\}$ and $\sim \in \{<, \leq\}$).

By transitivity of $\trianglelefteq$, we get $\mathbf{b}'' \trianglelefteq \mathbf{b}$ and thus $\mathbf{b} \in cl^{\Psi}\left(\mathscr{B}^+\right)$.

Next, we show that upon termination $cl^{\Psi}\left(\mathscr{B}^+\right) = cl^{\Psi}\left(Sat(\Psi)\right)$. Lemma 7 follows as sets with the same closure must have the same unique generator. $cl^{\Psi}\left(\mathscr{B}^+\right) \subseteq cl^{\Psi}\left(Sat(\Psi)\right)$ follows already from Lemma 6. For the other direction, let $\mathbf{b} \in cl^{\Psi}\left(Sat(\Psi)\right)$, i.e. there exists $\mathbf{b}' \in Sat(\Psi)$ with $\mathbf{b}' \trianglelefteq \mathbf{b}$. As $\mathbf{b}'$ is satisfying for Ψ, $\mathbf{b}' \notin cl^{\overline{\Psi}}\left(\mathscr{B}^-\right)$ holds. With $\mathbf{b}' \in \mathbb{N}^m \subseteq cl^{\Psi}\left(\mathscr{B}^+\right) \cup cl^{\overline{\Psi}}\left(\mathscr{B}^-\right)$ (as shown above), this yields $\mathbf{b}' \in cl^{\overline{\Psi}}\left(\mathscr{B}^+\right)$. It follows that there is $\mathbf{b}'' \in \mathbb{B}^+$ with $\mathbf{b}'' \trianglelefteq \mathbf{b}' \trianglelefteq \mathbf{b}$. Hence, $\mathbf{b} \in cl^{\Psi}\left(\mathscr{B}^+\right)$. □

Lemma 8 *Algorithm 2 terminates if* Ψ *and* $\overline{\Psi}$ *have finite natural generators.*

Proof Eventually all points of the finite set $gen^{\Psi}\left(Sat(\Psi)\right) \subseteq \mathbb{N}^m$ are considered in line 6 and inserted in $\mathscr{B}^+$, yielding $cl^{\Psi}\left(\mathscr{B}^+\right) = cl^{\Psi}\left(Sat(\Psi)\right)$. Similarly, $cl^{\overline{\Psi}}\left(\mathscr{B}^-\right) = cl^{\overline{\Psi}}\left(Sat(\overline{\Psi})\right)$ holds eventually. Hence, all $\mathbf{b} \in \mathbb{N}^m$ are contained in $cl^{\Psi}\left(\mathscr{B}^+\right) \cup cl^{\overline{\Psi}}\left(\mathscr{B}^-\right)$, which leads to termination of the procedure. □

Lemmas 7 and 8 yield correctness.

Theorem 3 *Algorithm 2 yields a generator of* Ψ*, if* Ψ *and* $\overline{\Psi}$ *have finite natural generators.*

If the algorithm terminates, it analyses at most $(b_{\max})^m$ epoch models, where

$$b_{\max} = \max\left\{\mathbf{b}[i] \mid 1 \leq i \leq m, \mathbf{b} \in gen^{\Psi}\left(Sat(\Psi)\right) \cup gen^{\overline{\Psi}}\left(Sat(\overline{\Psi})\right)\right\}.$$

Remark 1 (on quantiles with partially known cost limits) Algorithm 2 can be extended to quantiles of the form $Qu\left(\mathbb{P}_M^{opt}(\varphi_? \wedge \varphi) \sim p\right)$, where $\varphi_? = \bigwedge_{i=1}^{m\in\mathbb{N}}(\langle \mathrm{C}_{j_i}\rangle_{\sim_i ?}\, G_i)$ has unknown cost limits while $\varphi = \bigwedge_{i=m+1}^{n\in\mathbb{N}}(\langle \mathrm{C}_{j_i}\rangle_{\sim_i b_i}\, G_i)$ considers cost limits b_i that are known in advance. This can be achieved by considering proper epoch sequences $\mathfrak{E}$ in line 8 with $\mathrm{last}(\mathfrak{E}) = \langle \mathbf{e}_{cand}[1], \ldots, \mathbf{e}_{cand}[m], b_{m+1}, \ldots, b_n\rangle$.

Finite natural generators are only required for Lemma 8, i.e. to guarantee termination of the algorithm. Intuitively, Algorithm 2 lacks a mechanism to analyse the resulting reachability probability when one or more cost limits approach infinity. For the single-cost case, [2,57] suggest preprocessing steps to check whether $Sat(\Psi) \neq \emptyset$ and $Sat(\overline{\Psi}) \neq \emptyset$ holds before invoking their variant of sequential value iteration. The preprocessing steps ensures that sequential value iteration always finds a $b \in \mathbb{N}$ with $gen^{\Psi}(Sat(\Psi)) = \{b\}$. We next lift these ideas to multi-dimensional quantiles.

5.3 Upper Cost-Bounded Quantiles

Here, we consider upper cost-bounded quantile queries of the form

$$\Psi = Qu\left(\mathbb{P}_M^{opt}\left(\bigwedge_{i=1}^{m\in\mathbb{N}}(\langle \mathrm{C}_{j_i}\rangle_{\leq ?}\, G_i)\right) > p\right).$$

For the single-cost case the preprocessing checks the unbounded reachability probability [2, 57]. Example 14 shows that the complementary query $\overline{\Psi}$ might have a non-natural generator. To guarantee termination of Algorithm 2, it suffices to initialise the set $\mathscr{B}^-$ to the points from

$$\mathscr{B}^-_{\text{init}} = \{\mathbf{b} \in gen^{\overline{\Psi}}\left(Sat(\overline{\Psi})\right) \mid \mathbf{b}[i] = \infty \text{ for some } i\}, \text{ i.e. } \mathscr{B}^-_{\text{init}} = gen^{\overline{\Psi}}\left(Sat(\overline{\Psi})\right) \setminus \mathbb{N}^m.$$

Lemma 9 *Algorithm 2 terminates and returns a generator of upper cost-bounded quantile query Ψ if initially $\mathscr{B}^- = gen^{\overline{\Psi}}\left(Sat(\overline{\Psi})\right) \setminus \mathbb{N}^m$.*

Proof Ψ can only have a finite natural generator. Naturality follows from the definition of closure. We show finiteness by contradiction. Suppose that Ψ has an infinite generator $\mathscr{G} \subseteq \mathbb{N}^m$. As $\mathscr{G}$ is infinite, we can show that it contains two different points $\mathbf{b}_1$ and $\mathbf{b}_2$ with $\mathbf{b}_1[i] \leq \mathbf{b}_2[i]$ for all i, i.e. $\mathbf{b}_1 \trianglelefteq \mathbf{b}_2$ with $\trianglelefteq$ as in Definition 14. Since $\mathbf{b}_1 \in \mathscr{G}$, the definition of closure yields $cl^{\Psi}(\mathscr{G} \setminus \{\mathbf{b}_2\}) = cl^{\Psi}(\mathscr{G})$ which contradicts the assumption that $\mathscr{G}$ is a generator. Similarly, we can show that the generator of $\overline{\Psi}$ is finite. It follows that all points in $gen^{\Psi}(Sat(\Psi))$ and $gen^{\overline{\Psi}}\left(Sat(\overline{\Psi})\right) \cap \mathbb{N}^m$ are eventually considered in line 6 of Algorithm 2 and inserted into either $\mathscr{B}^+$ or $\mathscr{B}^-$. □

We now discuss how to compute $\mathscr{B}^-_{\text{init}}$: Consider $\mathbf{b} \in \mathscr{B}^-_{\text{init}}$, i.e. there is a non-empty set $\mathscr{I} \subseteq \{1, \ldots, m\}$ with $\mathbf{b}[i] = \infty$ if and only if $i \in \mathscr{I}$. Intuitively, this reflects a situation in which the cost-bounded reachability probability is always below the threshold p, for any amount of cost collected towards the cost limits given by $\mathscr{I}$. Formally, we have

$$\forall b \in \mathbb{N}\colon \mathbb{P}_M^{opt}\Big(\bigwedge_{i\notin\mathscr{I}}(\langle \mathrm{C}_{j_i}\rangle_{\leq \mathbf{b}[i]}\, G_i) \wedge \bigwedge_{i\in\mathscr{I}}(\langle \mathrm{C}_{j_i}\rangle_{\leq b}\, G_i)\Big) \leq p$$

$$\Leftrightarrow \lim_{b\to\infty}\left(\mathbb{P}_M^{opt}\Big(\bigwedge_{i\notin\mathscr{I}}(\langle \mathrm{C}_{j_i}\rangle_{\leq \mathbf{b}[i]}\, G_i) \wedge \bigwedge_{i\in\mathscr{I}}(\langle \mathrm{C}_{j_i}\rangle_{\leq b}\, G_i)\Big)\right) \leq p$$

Input : MDP $M = \langle S, T, s_{init} \rangle$, quantile $\Psi = Qu\left(\mathbb{P}_M^{opt}(\bigwedge_{i=1}^{m\in\mathbb{N}}(\langle \mathrm{C}_{j_i}\rangle_{\leq ?}\, G_i)) > p\right)$
Output : A generator of Ψ

1 $\mathscr{B}^- \leftarrow \emptyset$
2 **if** $m = 1$ **then**
3 **if** $\mathbb{P}_M^{opt}(\langle\cdot\rangle_{\geq 0}\, G_m) \leq p$ **then**
4 $\mathscr{B}^- = \{\, \langle \infty \rangle \,\}$
5 **else**
  // Solve $m-1$-dimensional quantile queries
6 **foreach** $k \in \{1, \ldots, m\}$ **do**
7 $\Psi_k \leftarrow Qu\left(\mathbb{P}_M^{opt}(\bigwedge_{i\in\{1,\ldots,m\}\setminus\{k\}}(\langle \mathrm{C}_{j_i}\rangle_{\leq ?}\, G_i) \wedge \langle\cdot\rangle_{\geq 0}\, G_k) > p\right)$
8 $\mathscr{B}^- \leftarrow \mathscr{B}^- \cup \{\langle b_1, \ldots, b_{k-1}, \infty, b_{k+1}, \ldots, b_{m-1}\rangle \mid \langle b_1, \ldots, b_{m-1}\rangle \in gen^{\overline{\Psi}_k}\left(Sat(\overline{\Psi}_k)\right)\}$
9 Call Algorithm 2 on M, Ψ with $\mathscr{B}^+ = \emptyset$ and $\mathscr{B}^-$ initialised as above
10 **return** $gen^{\Psi}\left(\mathscr{B}^+\right)$

Algorithm 3: Quantile computation with upper cost bounds.

$$\Leftrightarrow \quad \mathbb{P}_M^{opt}\Big(\bigwedge_{i\notin\mathscr{I}}(\langle \mathrm{C}_{j_i}\rangle_{\leq \mathbf{b}[i]}\, G_i) \wedge \bigwedge_{i\in\mathscr{I}}(\langle\cdot\rangle_{\geq 0}\, G_i)\Big)\Big) \leq p.$$

Here, upper cost bounds with arbitrarily high cost limits are replaced by unbounded reachability, following ideas from the single-dimensional case [2,57]. This transformation yields an $(m - |\mathscr{I}|)$-dimensional quantile query for which the point $\langle \mathbf{b}[i_1], \ldots, \mathbf{b}[i_n]\rangle$ for $\{1, \ldots, m\} \setminus \mathscr{I} = \{i_1, \ldots, i_n\}$ is a satisfying cost limit.

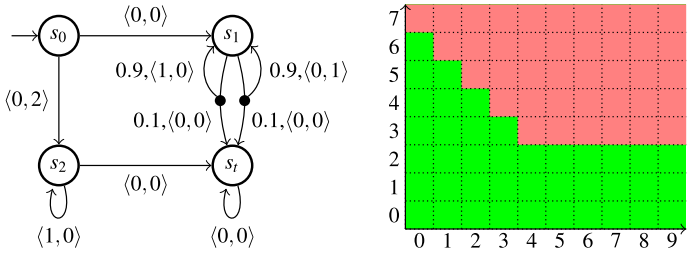
This observation provides the basis of our algorithm, outlined as Algorithm 3. In case of single-cost queries, it analyses the unbounded reachability probability to check whether there is some cost limit that satisfies the probability threshold in lines 2-4. For quantiles with $m \geq 2$ cost dimensions, we find the points $\mathbf{b} \in gen^{\overline{\Psi}}\left(Sat(\overline{\Psi})\right)$ with $\mathbf{b}[k] = \infty$ by checking all m many $(m-1)$-dimensional quantile queries Ψ_k where the k-th cost bound is replaced by an unbounded reachability constraint (lines 6-8). These quantiles with partially known cost limits (cf. Remark 1) can be solved by calling Algorithm 3, recursively. We cache the results of the recursive calls, i.e. each of the occurring quantile queries is only processed once. Since each of the m cost bounds can be replaced by an unbounded reachability constraint or not, this results in roughly 2^m different calls to Algorithm 2. However, the recursive calls consider a simpler quantile with only $m' < m$ dimensions.

Theorem 4 *Algorithm 3 yields a generator of upper cost-bounded quantile query Ψ.*

Proof We show that before calling Algorithm 2 in line 9, $\mathscr{B}^- = \mathscr{B}^-_{\text{init}}$ holds. For $m = 1$, this is ensured in lines 2-4. For $m \geq 2$, observe that

$$\mathbf{b} = \langle b_1, \ldots, b_m\rangle \in \mathscr{B}^- \Leftrightarrow \exists k : b_k = \infty \text{ and } \langle b_1, \ldots, b_{k-1}, b_{k+1}, b_m\rangle \in gen^{\overline{\Psi}_k}\left(Sat(\overline{\Psi}_k)\right)$$
$$\Leftrightarrow \mathbf{b} \in gen^{\overline{\Psi}}\left(Sat(\overline{\Psi})\right) \setminus \mathbb{N}^m = \mathscr{B}^-_{\text{init}}.$$

The statement than follows from Lemma 9. □



(a) MDP M'' **(b)** $Sat(\Psi''_{ex})$

Fig. 10 Example MDP M'' and satisfying cost limits for Ψ''_{ex}

5.4 Lower Cost-Bounded Quantiles

We now consider lower cost-bounded quantile queries of the form

$$\Psi = Qu\left(\mathbb{P}^{\max}_{M}\Big(\bigwedge_{i=1}^{m\in\mathbb{N}}(\langle \mathrm{C}_{j_i}\rangle_{\geq ?}\, G_i)\Big) > p\right).$$

Remark 2 We restrict to maximising schedulers. In Sect. 5.5, we discuss issues with lower cost-bounded quantiles under minimising schedulers, i.e. quantile queries with $\mathbb{P}^{\min}_{M}(\varphi)$.

The following example shows that Ψ might have a finite non-natural generator.

Example 15 Consider the MDP M''_{Qu} given in Fig. 10a and the quantile query

$$\Psi''_{ex} = Qu\left(\mathbb{P}^{max}_{M''_{Qu}}\ \big(\langle \mathrm{C}_1\rangle_{\geq ?}\,\{\,s_t\,\} \wedge \langle \mathrm{C}_2\rangle_{\geq ?}\,\{\,s_t\,\}\big)\ > 0.5\right).$$

The (lower-left, brighter) green area in Fig. 10b indicates the set of satisfying cost limits for Ψ''_{ex}, given by

$$Sat(\Psi''_{ex}) = \{\,\mathbf{c}\in\mathbb{N}^2 \mid \exists\,\mathbf{b}\in\{\,\langle 0,6\rangle,\langle 1,5\rangle,\langle 2,4\rangle,\langle 3,3\rangle,\langle\infty,2\rangle\,\}\colon \mathbf{b}[1]\geq\mathbf{c}[1]\wedge\mathbf{b}[2]\geq\mathbf{c}[2]\,\}.$$

Similar to quantiles with upper cost bounds, we ensure termination of Algorithm 2 by initialising $\mathscr{B}^{+}$ with $\mathscr{B}^{+}_{\text{init}} = gen^{\Psi}\,(Sat(\Psi))\setminus\mathbb{N}^m$.

Lemma 10 *Algorithm 2 terminates and returns a generator of lower cost-bounded quantile query Ψ if initially $\mathscr{B}^{+} = gen^{\Psi}\,(Sat(\Psi))\setminus\mathbb{N}^m$.*

Points $\mathbf{b}\in\mathscr{B}^{+}_{\text{init}}$ satisfy

$$\lim_{b\to\infty}\left(\mathbb{P}^{\max}_{M}\Big(\bigwedge_{i\notin\mathscr{I}}(\langle \mathrm{C}_{j_i}\rangle_{\geq \mathbf{b}[i]}\, G_i)\wedge\bigwedge_{i\in\mathscr{I}}(\langle \mathrm{C}_{j_i}\rangle_{\geq b}\, G_i)\Big)\right) > p,$$

where $\mathscr{I}\subseteq\{1,\ldots,m\}$ is a non-empty set with $\mathbf{b}[i]=\infty$ if and only if $i\in\mathscr{I}$. For the single-cost case, the computation of $\lim_{b\to\infty}\mathbb{P}^{\max}_{M}(\langle \mathrm{C}_j\rangle_{\geq b}\,G)$ has been addressed in [2]: The approach relies on the notion of end components (Definition 3). For a finite path $\pi_{\text{fin}} = s_0\,\mu_0\,\mathbf{c}_0\,s_1\,\ldots\mu_{n-1}\,\mathbf{c}_{n-1}\,s_n$, we consider the costs accumulated in end components, given by

$$\text{cost}^{EC}_{i}(\pi_{\text{fin}}) = \sum_{j=0}^{n-1}\mathbf{c}_j[i]\cdot[\mu_j\in T'(s_j)\text{ for some EC }T'].$$

Input : MDP $M = \langle S, T, s_{init} \rangle$, quantile $\Psi = Qu\left(\mathbb{P}_M^{\max}(\bigwedge_{i=1}^{m\in\mathbb{N}}(\langle \mathrm{C}_{j_i}\rangle_{\geq ?}\, G_i)) > p\right)$
Output : A generator of Ψ
$\mathscr{B}^+ \leftarrow \emptyset$
if $m = 1$ **then**
  if $\mathbb{P}_M^{\max}(\langle \mathrm{C}_{j_1}^{EC}\rangle_{>0}\, G_m) > p$ **then**
    $\mathscr{B}^+ = \{\langle \infty \rangle\}$
else
  // Solve $m-1$-dimensional quantile queries
  foreach $k \in \{1, \ldots, m\}$ **do**
    $\Psi_k \leftarrow Qu\left(\mathbb{P}_M^{\max}(\bigwedge_{i\in\{1,\ldots,m\}\setminus\{k\}}(\langle \mathrm{C}_{j_i}\rangle_{\geq ?}\, G_i) \wedge \langle \mathrm{C}_{j_k}^{EC}\rangle_{>0}\, G_k) > p\right)$
    $\mathscr{B}^+ \leftarrow \mathscr{B}^+ \cup \{\langle b_1, \ldots, b_{k-1}, \infty, b_{k+1}, \ldots, b_{m-1}\rangle \mid \langle b_1, \ldots, b_{m-1}\rangle \in gen^{\Psi_k}(Sat(\Psi_k))\}$
Call Algorithm 2 on M, Ψ with $\mathscr{B}^- = \emptyset$ and $\mathscr{B}^+$ initialised as above
return $gen^{\Psi}\left(\mathscr{B}^+\right)$

Algorithm 4: Quantile computation with lower cost bounds.

We write $\langle \mathrm{C}_j^{EC}\rangle_{>b}\, G$ to characterise the set of paths Π where every $\pi \in \Pi$ has a prefix π_{fin} with $\text{last}(\pi_{\text{fin}}) \in G$ and $\text{cost}_j^{EC}(\pi_{\text{fin}}) > b$. If we reach an EC in which costs can be collected, we can accumulate arbitrarily more cost by staying in that EC. We thus have $\mathbb{P}_M^{\max}(\langle \mathrm{C}_j^{EC}\rangle_{>0}\, G) = \mathbb{P}_M^{\max}(\langle \mathrm{C}_j^{EC}\rangle_{>b}\, G)$ for all $b \in \mathbb{N}$ as shown in [2].

We lift this observation to multiple lower cost bounds. In this case, any visit of an EC can be extended to accumulating more costs, without violating other (lower) cost bounds. Thus, we get:

$$\lim_{b\to\infty} \mathbb{P}_M^{\max}\left(\bigwedge_{i=1}^{m-1}(\langle \mathrm{C}_{j_i}\rangle_{\geq \mathbf{b}[i]}\, G_i) \wedge \langle \mathrm{C}_{j_m}\rangle_{\geq b}\, G_m\right) = \mathbb{P}_M^{\max}\left(\bigwedge_{i=1}^{m-1}(\langle \mathrm{C}_{j_i}\rangle_{\geq \mathbf{b}[i]}\, G_i) \wedge \langle \mathrm{C}_{j_m}^{EC}\rangle_{>0}\, G_m\right)$$

Our procedure for lower cost-bounded quantiles is shown in Algorithm 4. Similar to Algorithm 3, it adds points $\mathbf{b}$ to $\mathscr{B}_{\text{init}}^+$ with $\mathbf{b}[i] = \infty$ for some i. In order to compute probabilities for arbitrarily high cost limits, it replaces the corresponding cost bound by $\langle \mathrm{C}_{j_k}^{EC}\rangle_{>0}\, G_k$.

Theorem 5 *Algorithm 4 yields a generator of lower cost-bounded quantile query Ψ.*

5.5 Intricate Quantile Queries

Above, we have considered only a subset of the possible quantile queries. Many more combinations are possible. These combinations do not easily fit into the framework provided above. We illustrate this mismatch with some cases.

Lower Cost-Bounded Quantiles Under Minimising Schedulers To lift the results from the previous section to quantile queries that consider $\mathbb{P}_M^{\min}(\varphi)$, we need to compute probabilities when one or more cost limits approach infinity. For the single-cost case, [2] proposes a transformation that instead computes the maximal probability to reach an end component in which either no goal state is visited or no cost is accumulated. However, in the multi-cost case, such a transformation does not preserve the other cost bounds.

Mixtures of Lower and Upper Cost Bounds Quantile queries that consider mixtures of lower- and upper cost bounds might have infinite generators, as shown in Example 14. In this case, a finite representation of the satisfying cost limits cannot be achieved with our explicit

construction of the generator. A procedure to check the subset of such queries that still yield finite generators is left for future work.

Quantiles over Multi-objective Tradeoffs We considered quantile queries with a single probability operator $\mathbb{P}^{opt}_{M}(\varphi)$. An extension inspired by multi-objective queries considers quantiles[2] over several conflicting objectives:

$$Qu\left(\exists\, \mathfrak{S} : \mathbb{P}^{\mathfrak{S}}_{M}(\varphi_{?,1}) \sim_1 p_1 \wedge \cdots \wedge \mathbb{P}^{\mathfrak{S}}_{M}(\varphi_{?,\ell}) \sim_\ell p_\ell\right)$$

Such a query asks for the cost limits for which there is a scheduler that satisfies all probability thresholds. This introduces two sources of tradeoffs: A tradeoff between different resolutions of nondeterminism and a tradeoff between different cost limits. Handling such a tradeoff requires to analyse the tradeoffs at each epoch model, and propagating the results through the epochs requires great care and is outside the scope of this paper.

6 Visualisations

The aim of visualising the results of a multi-objective model checking analysis is to present the tradeoffs between the different objectives such that the user can make an informed decision about the system design or pick a scheduler for implementation. However, the standard Pareto set visualisations alone may not provide sufficient information, about e.g. which objectives are aligned or conflicting (see e.g. [43] for a discussion in the non-probabilistic case). Cost bounds furthermore add an extra dimension for each cost structure. In particular, for each Pareto-optimal scheduler, our method has implicitly computed the probabilities of all objectives for all reachable epochs as well, i.e. for all bounds on all quantities below the bounds required in the tradeoff. In this section, we first show the standard Pareto curve visualisation, which provides the user with an easy-to-understand but very high-level overview of the solution space of a multi-objective query. We then propose a way to visualise the behaviour of individual Pareto-optimal schedulers w.r.t. the probabilities of the individual objectives and the bound values in two-dimensional heatmap plots. These plots provide deep insights into the behaviour of each scheduler, its robustness w.r.t. the bounds, and its preferences for certain objectives depending on the remaining budget for each quantity. Yet due to the need to reduce the dimensionality of the available information, they can be difficult to understand at a first glance. We offer them both as a first straightforward attempt at visualising all the available data as well as an urgent call to visualisation experts to develop more perspicuous ways to present this wealth of data to users.

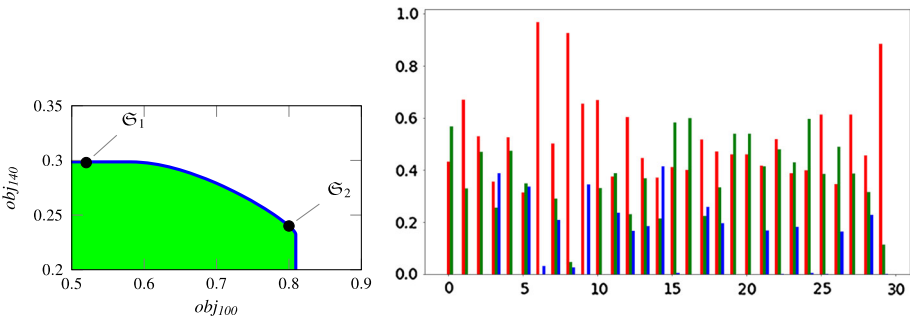
6.1 Pareto Curves

The results of a multi-objective model checking analysis are typically presented as a single (approximation of a) Pareto curve. As a running example for this section, consider the Mars rover MDP M_r and tradeoff $multi\big(obj_{100}, obj_{140}\big)$ with

$$obj_v = \mathbb{P}^{\max}_{M_r}(\langle \mathrm{C}_{time}\rangle_{\leq 175}\, B \wedge \langle \mathrm{C}_{energy}\rangle_{\leq 100}\, B \wedge \langle \mathrm{C}_{value}\rangle_{\geq v}\, B)$$

where B is the set of states where the rover has safely returned to its base. That is, we ask for the tradeoff between performing experiments of scientific value at least 100 before returning to base within 175 time units and maximum energy consumption of 100 units (obj_{100}) versus

[2] Technically, these objects are not quantiles in the classical sense.



(a) Pareto curve for $multi(obj_{100}, obj_{140})$ **(b)** Optimal schedulers for 3 objectives

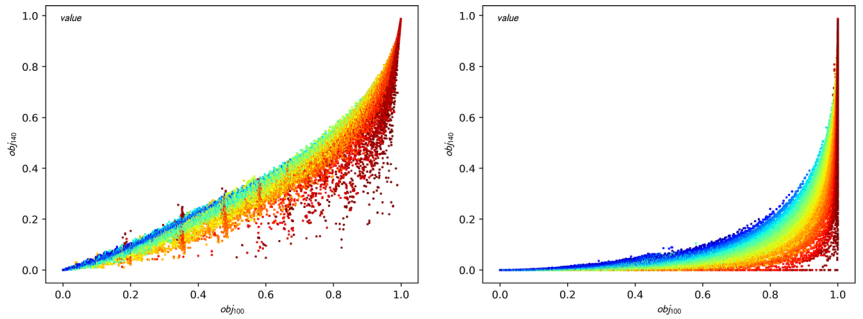
Fig. 11 Pareto curves

achieving the same with scientific value at least 140 (obj_{140}). The corresponding Pareto curve is shown in Fig. 11a. Every point $\langle x, y \rangle$ in the green area on the bottom left corresponds to a scheduler under which the probability to satisfy obj_{100} is x and the probability to satisfy obj_{140} is y. The thick blue line is the frontier of Pareto-optimal schedulers: for any scheduler on this line, there is no other scheduler that achieves strictly higher probabilities for *both* objectives (cf. Sect. 2.4). Overall, this Pareto curve clearly shows that there is a tradeoff between achieving obj_{100} and obj_{140}; more risky behaviour is necessary to increase the chance of reaching obj_{140}, thereby decreasing the chance of reaching the "easier" objective obj_{100}. For more than two objectives, the performance of a set of concrete Pareto-optimal schedulers can be displayed in a bar chart as in Fig. 11b, where the colours reflect different objectives and the groups different schedulers.

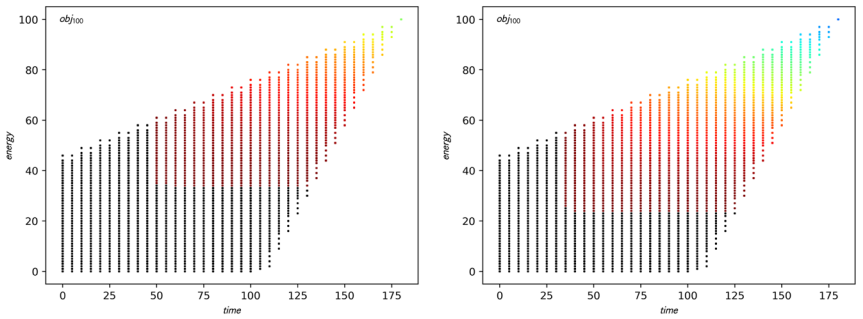
6.2 Visualising Bounded Multi-Objective Schedulers

Pareto curves and bar charts as presented above reduce schedulers to the probabilities of reaching each of the objectives. However, in a cost-bounded setting, users may arguably not only be interested in the probability for the exact cost limit, but also in the behaviour of the scheduler for lower limits: in essence, the probability distribution over the limits. As our method implicitly computes the probabilities of the objectives for *all reachable* epochs, this information is to a large extent available "for free" at no extra computational effort (limited only by which epochs are reachable).

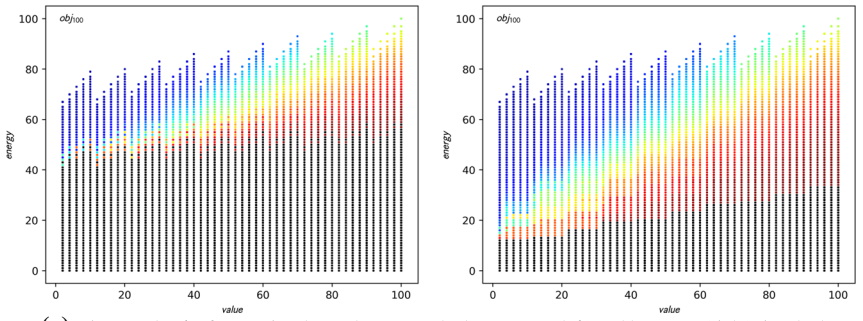
We visualise this information via plots for individual Pareto-optimal schedulers as shown in Fig. 12. We restrict to two-dimensional plots since they are easier to grasp than complex three-dimensional visualisations. In each plot, we can thus show the relationship between three different quantities: one on the x-axis, one on the y-axis, and one encoded as the colour of the points (z, where we use blue for high values, red for low values, black for probability zero, and white for unreachable epochs). Our example tradeoff $multi(obj_{100}, obj_{140})$ however already contains five quantities: the probability for obj_{100}, the probability for obj_{140}, the available time and energy to be spent, and the remaining scientific value to be accumulated. We thus need to project out some quantities. We do this by showing at every $\langle x, y \rangle$ coordinate the *maximum* or *minimum* value of the z quantity when ranging over *all* reachable values of the hidden *costs* at this coordinate. That is, we show a best- or worst-case situation, depending on the semantics of the respective quantities. Out of the 30 possible combinations of quantities



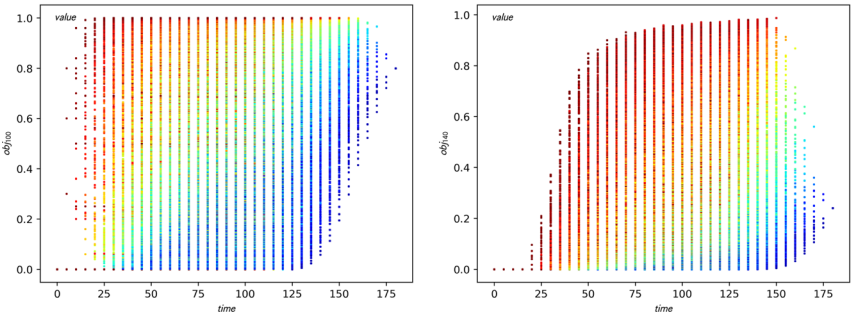
(a) Maximum sci. value budget to achieve probabilities of x: obj_{100} and y: obj_{140}, $\mathfrak{S}_1$ (left) and $\mathfrak{S}_2$ (right)



(b) Worst-case probability of obj_{100} achieved for x: time and y: energy budget under $\mathfrak{S}_1$ (left) and $\mathfrak{S}_2$ (right)



(c) obj_{100} under $\mathfrak{S}_1$ for x: sci. value and y: energy budget, worst- (left) and best-case (right) time budget



(d) Min. sci. value requirement under $\mathfrak{S}_2$ with x: time budget to achieve y: obj_{100} (left) resp. y: obj_{140} (right)

Fig. 12 Two-dimensional plots of individual Pareto-optimal schedulers for different quantities

for $multi(obj_{100}, obj_{140})$, we showcase four in Fig. 12 to illustrate the added value of the obtained information.

Comparing Schedulers on Value Required to Achieve Objectives First, in Fig. 12a, we plot for the two Pareto-optimal schedulers $\mathfrak{S}_1$ and $\mathfrak{S}_2$ (cf. Fig. 11a) the probabilities of the two objectives on the x- and y-axes versus the scientific value that still needs to be accumulated in the z (colour) dimension. White areas thus indicate that no epoch for the particular combination of probabilities is reachable from the tradeoff's cost limits of 175 for time, 100 for energy, and 140 as the higher limit for scientific value. In principle, a point $\langle x, y, z \rangle$ in these two plots reads as follows: there is an epoch in which scientific value 140-z has already been achieved, the probability to reach obj_{100} (i.e. achieve scientific value 100) is x, and the probability to reach obj_{140} is y. However, two cost dimensions—the remaining budget for time and the remaining budget for energy—are not shown and need to be projected out. The two plots in fact show in the z dimension the *maximum* scientific value that still needs to be accumulated over all reachable time and energy budgets for the respective probability values. To be precise, a point $\langle x, y, z \rangle$ thus actually needs to be read as follows: among all epochs in which the probability to reach obj_{100} is x and the probability to reach obj_{140} is y, the maximum difference between 140 and the already accumulated scientific value is z. Since a *higher* amount of scientific value to be accumulated is *easier* to reach (less value needs to be accumulated), these plots actually show a "best-case" scenario: the point where enough scientific value has been accumulated to achieve a certain combination of probabilities. The plots for the minimum values are almost the same in this case, though.

We see that $\mathfrak{S}_1$ and $\mathfrak{S}_2$ are white above the diagonal, as are in fact all other Pareto-optimal schedulers, which means that obj_{100} implies obj_{140}, i.e. the objectives are aligned. For $\mathfrak{S}_1$, we further see that all blue-ish areas are associated to low probabilities for both objectives: this scheduler achieves only low probabilities when it still needs to make the rover accumulate a high amount of value. However, it overall achieves higher probabilities for obj_{140} at medium value requirements, whereas $\mathfrak{S}_2$ is "safer" and focuses on satisfying obj_{100}. The erratic spikes for $\mathfrak{S}_1$ correspond to combinations of probabilities for the objectives that are reached only via unlikely paths.

Comparing Schedulers on Objectives Depending on Budgets Figure 12b again contrasts schedulers $\mathfrak{S}_1$ and $\mathfrak{S}_2$, but this time in terms of the probability for obj_{100} (z colour) only, depending on the remaining time (x-axis) and energy budget (y-axis). We plot the *minimum* probability over the hidden scientific value requirement, i.e. a worst-case view. The plots show that time is of little use in case of low remaining energy but helps significantly when there is sufficient energy. The comparison of the two schedulers mainly confirms, but also explains, their positions in Fig. 11a: $\mathfrak{S}_1$ achieves overall lower probabilities for obj_{100} than $\mathfrak{S}_2$. If we were to compare each of these plots with the corresponding plot where the z colour is the probability for obj_{140} (not shown here), we would see that the visual difference between the plots for $\mathfrak{S}_1$ is small whereas the two plots for $\mathfrak{S}_2$ are noticeably different—again confirming the Pareto curve plot.

Comparing Best and Worst Case Projections In Fig. 12c, we show for $\mathfrak{S}_1$ the probability to achieve obj_{100} (z colour) depending on the remaining scientific value to be accumulated (x-axis) and the remaining energy budget (y-axis). There is a white vertical line for every odd x-value: over all branches in the model, the gcd of all scientific value costs is 2. The remaining time has to be projected out. The left plot shows the minimum probabilities over the hidden costs, i.e. we see the probability for the worst-case remaining time; the right plot shows the best-case scenario. Not surprisingly, we see that when time is low, only a lot of energy makes it possible to reach the objective with non-zero probability. We also observe that without significant time restrictions (i.e. in the best case), only very limited energy is

required to obtain positive probabilities (especially, of course, if only little more scientific value needs to be accumulated, i.e. on the left side of the right-hand plot).

Comparing one Scheduler Over Two Objectives Finally, in Fig. 12d, we depict for scheduler $\mathfrak{S}_2$ the minimum remaining scientific value required (z colour) such that a certain probability for obj_{100} (left) or obj_{140} (right) can be achieved (y-axis), given a certain remaining time budget (x-axis). Note that this is a worst-case view: minimum remaining means maximum accumulated scientific value, i.e. we show the value needed to achieve a probability for the worst choice in hidden costs (i.e. in energy budget). The upper left corner for obj_{100} shows that a *high* probability in *little* time is only achievable if we need to collect *little* more value (red points); the value requirement gradually relaxes as we aim for lower probabilities or have more time. We see white areas in the rightmost parts of both plots; for obj_{100}, they are in the lower probability ranges, while for obj_{140}, they are in the higher probability ranges. This reflects the tradeoff made by $\mathfrak{S}_2$ to strongly favour obj_{100} over obj_{140}: with a high time budget, it has many choices available (on which experiments to perform), and it makes them such that it achieves obj_{100} with a high probability, at the cost of obj_{140}—and with a high time budget, the probabilities are independent of the value and energy budget.

7 Experiments

We implemented the presented approaches into Storm [21], available at [20]. For multi-cost bounded queries, the implementation computes extremal probabilities for the single-objective and Pareto curves for the multi-objective case. In addition, Storm computes generators for two-dimensional quantile queries with only upper or only lower cost bounds.

We evaluate the approaches on a wide range of Markov chain and MDP case studies. Our benchmark selection includes DTMC models (*Crowds* and *Nand*) and MDP models (*FireWire* and *Wlan*) from the Prism benchmark suite [40]. Moreover, we consider MDP models that have been studied in the context of multi-objective model checking (*Service* and *UAV*) and in the context of cost-bounded reachability (*JobSched* and *Resources*). Finally, we consider the MDP model of the *Rover* from Sect. 1. The models are given in Prism's [39] guarded command language. Except for *Crowds*, all epoch models for cost epochs $\mathbf{e} \in \mathbb{N}^m$ are acyclic.

We ran our experiments on a single core (2 GHz) of a HP BL685C G7 system with 192 GB of memory. We stopped each experiment after a time limit of 2 h.

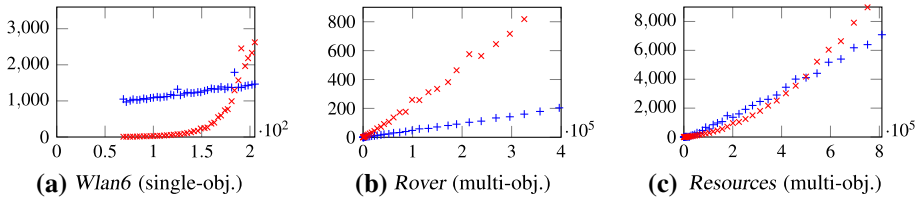
Details on replicating the tables, as well as details on how to analyse multi-cost bounded properties using Storm in general, are enclosed in the artifact [33].

7.1 Multi-cost Bounded Reachability Queries

Implementation Details We use the *sparse* engine of Storm, i.e. explicit data structures such as sparse matrices. The expected costs (lines 14 to 16 of Algorithm 1) are computed either numerically (via interval iteration over finite-precision floats) or exactly (via policy iteration [26] over infinite-precision rationals[3]). To reduce memory consumption, the analysis result of an epoch model $M_f^{\mathbf{e}}$ is erased once its predecessor epochs have been processed.

Set-Up We compare the naïve unfolding approach (UNF) as in Sect. 3 with the sequential approach (SEQ) as in Sect. 4. Globally, we considered precision $\eta = 10^{-4}$ for the Pareto curve approximation and precision $\varepsilon = 10^{-6}$ for interval iteration.

[3] Storm uses the GNU MP Arithmetic Library available at https://gmplib.org.



(a) *Wlan6* (single-obj.) **(b)** *Rover* (multi-obj.) **(c)** *Resources* (multi-obj.)

Fig. 13 Runtime (y-axis) of SEQ (+) and UNF (×) for increasing cost bounds (x-axis) on three benchmarks

– For UNF, the unfolding of the model is applied at the PRISM language level, by considering a parallel composition with cost counting structures. For *Crowds*, *Nand*, and *Wlan*, these unfolded models are part of the PRISM benchmark suite [40]. On the unfolding we apply the algorithms for unbounded reachability as available in STORM.
– For SEQ, we increased the precision for single epoch models as in Theorem 2.

For all MDP case studies we consider single- and multi-objective cost-bounded reachability queries that yield non-trivial results, i.e. probabilities strictly between zero and one. For DTMCs, we only consider single-objective queries: there are no tradeoffs between schedulers since there is no nondeterminism.

Results Tables 1 and 2 show results for single- and multi-objective queries, respectively. The first columns yield the number of states and transitions of the original MDP, then for the query, the number of bounds m, the number of *different* cost structures r, and the number of reachable cost epochs $|\mathfrak{E}|$ (reflecting the magnitude of the bound values). $|S_{unf}|$ denotes the number of reachable states in the unfolding. For multi-objective queries, we additionally give the number of objectives and the number of analysed weight vectors $\mathbf{w}$. The remaining columns depict the runtimes of the different approaches in seconds. For UNF, we considered both the sparse (sp) and symbolic (dd) engine of STORM. The symbolic engine neither supports multi-objective model checking nor exact policy iteration. For experiments that completed within the time limit, we observed a memory consumption of up to 110 GB for UNF and up to 8 GB for SEQ.

Evaluation On the majority of benchmarks, SEQ performs better than UNF. Typically, SEQ is less sensitive to increases in the magnitude of the cost bounds, as illustrated in Fig. 13. For three benchmark and query instances, we plot the runtime of both approaches against different numbers $|\mathfrak{E}|$ of reachable epochs. While for small cost bounds, UNF is sometimes faster compared to SEQ, SEQ scales better with increasing $|\mathfrak{E}|$. It is not surprising that SEQ scales better: ultimately, the increased state space size and the accompanying memory consumption in UNF is a bottleneck. The most important reason that UNF performs better for some (smaller) cost bounds is the induced overhead of checking the full epoch. In particular, the epoch contains (often many) states that are not reachable from the initial state (in the unfolding).

7.2 Multi-dimensional Quantiles

Implementation Details STORM computes generators for 2-dimensional queries with either upper or lower cost bounds as presented in Sects. 5.3 and 5.4. We also allow for additional cost bounds with fixed cost limits as in Remark 1. Similar to multi-cost bounded queries, we consider STORM's sparse engine and compute expected costs (line 25 of Algorithm 2)

Table 1 Runtime comparison for multi-cost single-objective queries

Benchmark instance							Interval It			Policy It.	
Case study		$\|S\|$	$\|T\|$	r-m	$\|\mathfrak{E}\|$	$\|S_{unf}\|$	UNF-dd	UNF-sp	SEQ	UNF-sp	SEQ
Crowds	[50]	143	563	2–2	24	$1 \cdot 10^7$	34	249	**< 1**	748	**< 1**
Crowds		143	563	2–2	66	$2 \cdot 10^{13}$	2515	TO	**< 1**	TO	**< 1**
Crowds		143	563	2–2	306	?	TO	TO	**< 1**	TO	**5**
Nand	[45]	$2 \cdot 10^6$	$4 \cdot 10^6$	1–2	10	$2 \cdot 10^7$	TO	145	**55**	757	**551**
Nand		$2 \cdot 10^6$	$4 \cdot 10^6$	1–2	22	$5 \cdot 10^7$	TO	363	**60**	3360	**2390**
Nand		$2 \cdot 10^6$	$4 \cdot 10^6$	1–2	202	$5 \cdot 10^8$	TO	4082	**118**	TO	TO
Service	[42]	$8 \cdot 10^4$	$2 \cdot 10^5$	1–1	162	$6 \cdot 10^6$	47	136	**10**	1945	**48**
JobSched2	[37]	349	660	2–2	503	$2 \cdot 10^4$	**< 1**	**< 1**	**< 1**	1	**< 1**
JobSched3		4584	$1 \cdot 10^5$	2–2	922	$3 \cdot 10^6$	**4**	10	**4**	26	**13**
JobSched5		$1 \cdot 10^6$	$4 \cdot 10^6$	2–2	2114	$4 \cdot 10^8$	**2944**	TO	3220	TO	TO
FireWire	[54]	776	1411	2–2	6024	$7 \cdot 10^5$	7	8	**2**	274	**144**
FireWire		776	1411	2–2	$1 \cdot 10^5$	$1 \cdot 10^7$	165	147	**45**	TO	**2803**
Resources	[6]	94	326	3–3	$2 \cdot 10^4$	$6 \cdot 10^5$	**< 1**	18	5	46	**9**
Resources		94	326	3–3	$1 \cdot 10^7$	$6 \cdot 10^8$	TO	TO	**2693**	TO	TO
Rover		16	30	3–3	$9 \cdot 10^4$	$1 \cdot 10^6$	38	24	**4**	704	**106**
Rover		16	30	3–3	$1 \cdot 10^7$	$2 \cdot 10^8$	TO	6040	**713**	TO	TO
UAV	[24]	$1 \cdot 10^5$	$6 \cdot 10^4$	1–1	52	$4 \cdot 10^4$	**1**	**1**	**1**	**4**	27
UAV		$1 \cdot 10^5$	$6 \cdot 10^4$	1–1	102	$4 \cdot 10^5$	7	16	**2**	72	**46**
Wlan3	[41]	$1 \cdot 10^5$	$2 \cdot 10^5$	1–1	82	$3 \cdot 10^6$	9	63	**8**	**126**	800
Wlan3		$1 \cdot 10^5$	$2 \cdot 10^5$	1–1	202	$1 \cdot 10^7$	820	293	**14**	**848**	2155
Wlan6		$5 \cdot 10^6$	$1 \cdot 10^7$	1–1	82	$2 \cdot 10^7$	**12**	363	989	**643**	TO
Wlan6		$5 \cdot 10^6$	$1 \cdot 10^7$	1–1	202	$6 \cdot 10^8$	2292	TO	**1399**	TO	TO

Bold numbers denote the fastest approach for each benchmark instance

Table 2 Runtime comparison for multi-cost multi-objective queries

Benchmark instance							Interval It.		Policy It.	
Case Study	$\|S\|$	$\|T\|$	ℓ-r-m	$\|\mathfrak{E}\|$	#w	$\|S_{unf}\|$	UNF-sp	SEQ	UNF-sp	SEQ
Service	$8 \cdot 10^4$	$2 \cdot 10^5$	2–1–2	162	34	$6 \cdot 10^6$	1918	**543**	TO	**4679**
JobSched2	349	660	2–4–4	$4 \cdot 10^4$	2	$1 \cdot 10^5$	**3**	54	**15**	183
JobSched3	4584	$1 \cdot 10^5$	2–4–4	$1 \cdot 10^6$	35	$2 \cdot 10^6$	**96**	TO	**6239**	TO
JobSched5	$1 \cdot 10^6$	$4 \cdot 10^6$	2–4–4	$3 \cdot 10^5$	?	?	TO	TO	TO	TO
FireWire	776	1411	2–2–2	6024	3	$7 \cdot 10^5$	32	**17**	TO	**1159**
FireWire	776	1411	2–2–2	$1 \cdot 10^5$	2	$1 \cdot 10^7$	863	**225**	TO	TO
Resources	94	326	2–3–4	$2 \cdot 10^5$	3	$6 \cdot 10^5$	25	**16**	2047	**52**
Resources	94	326	2–3–4	$1 \cdot 10^8$	?	?	TO	TO	TO	TO
Rover	16	30	2–3–3	$9 \cdot 10^5$	7	$1 \cdot 10^6$	177	**39**	5817	**3328**
Rover	16	30	2–3–3	$1 \cdot 10^8$	7	$2 \cdot 10^8$	TO	**5785**	TO	TO
UAV	$1 \cdot 10^5$	$6 \cdot 10^4$	2–1–2	52	18	$4 \cdot 10^4$	**2**	24	**102**	1098
UAV	$1 \cdot 10^5$	$6 \cdot 10^4$	2–1–2	102	22	$4 \cdot 10^5$	70	**39**	**2282**	3062
Wlan3	$1 \cdot 10^5$	$2 \cdot 10^5$	3–1–2	82	68	$3 \cdot 10^6$	5239	**2231**	TO	TO
Wlan3	$1 \cdot 10^5$	$2 \cdot 10^5$	3–1–2	202	4	$1 \cdot 10^7$	1769	**185**	TO	TO
Wlan6	$5 \cdot 10^6$	$1 \cdot 10^7$	3–1–2	82	?	$2 \cdot 10^7$	TO	TO	TO	TO

Bold numbers denote the fastest approach for each benchmark instance

via interval iteration over finite precision floats or via policy iteration over infinite precision rationals.

Set-Up For interval iteration, we used precision $\varepsilon = 10^{-6}$. The probability threshold for all quantile queries was set to 0.95. We considered the benchmarks from Table 1 where supported queries with a non-trivial generator (i.e. generators that contain a non-zero cost limit for each cost bound) could be assembled. For *FireWire*, *Resources*, and *Rover*, we consider the same cost-bounded reachability formulas as in Table 1.

Results Table 3 shows our results for multi-dimensional quantile queries. The columns depict the number of states and transitions for each model, the dimension of the quantile query m, the number of additional cost bounds with *fixed* cost limits n, the number of analysed cost epochs $|\mathfrak{E}|$, the number of points in the computed generator $|gen|$, and the runtimes for interval iteration and policy iteration, respectively. Experiments that finished within the time limit required at most 7 *GB* of memory.

Evaluation Our experiments indicate the practicability of our approach. Naturally, the runtime largely depends on the epochs analysed. Comparing with the cost-bounded reachability (single objective), we can see that the overhead of not knowing a priori which epochs to check is significant, but manageable (a rough estimate is a factor of 2). The generators for the considered quantile queries only contain a small number of points, allowing for a concise representation of the set of satisfying cost limits. This small number raises hopes that a good heuristic for selecting candidates might be able to properly approximate such a generator quite fast.

Table 3 Runtime comparison for multi-dimensional quantile queries

Benchmark Instance						Interval It.	Policy It.
Case Study	$\|S\|$	$\|T\|$	m-n	$\|\mathfrak{E}\|$	$\|gen\|$		
JobSched2	349	660	2–1	8311	5	13	19
JobSched3	4584	$1 \cdot 10^5$	2–1	$4 \cdot 10^4$	5	289	631
JobSched5	$1 \cdot 10^6$	$4 \cdot 10^6$	2–1	?	?	TO	TO
FireWire	776	1411	2–0	512	1	< 1	1
Resources	94	326	2–1	9791	7	3	6
Resources	94	326	2–1	$1 \cdot 10^5$	16	104	146
Resources	94	326	2–1	$1 \cdot 10^6$	31	2931	3489
Rover	16	30	2–1	$4 \cdot 10^4$	12	27	33
Rover	16	30	2–1	$3 \cdot 10^5$	24	873	889
Rover	16	30	2–1	$8 \cdot 10^5$	34	6148	5315
Wlan3	$1 \cdot 10^5$	$2 \cdot 10^5$	2–0	2428	8	82	728
Wlan6	$5 \cdot 10^6$	$1 \cdot 10^7$	2–0	2428	8	5005	TO

8 Conclusion

Many real-world planning problems consider several limited resources and contain tradeoffs. This article presented a practically efficient approach to analyse these problems. It has been implemented in the STORM model checker and shows significant performance benefits. The extension to quantiles enables the user to tackle the planning and optimisation problem from an orthogonal angle. Our new algorithm implicitly computes a large amount of information that is hidden in the standard plots of Pareto curves shown to visualise the results of a multi-objective analysis. We have developed a new set of visualisations that exploit all the available data to provide new insights to decision makers even for problems with many objectives and cost dimensions; yet we also call for experts to improve on these visualisations, as we believe that an intuitive presentation of the vast amount of result data needs to accompany an efficient algorithm like ours to exploit its full usage potential.

References

1. Andova, S., Hermanns, H., Katoen, J.P.: Discrete-time rewards model-checked. In: FORMATS, LNCS, vol. 2791, pp. 88–104. Springer (2003)
2. Baier, C., Daum, M., Dubslaff, C., Klein, J., Klüppelholz, S.: Energy-utility quantiles. In: NFM, LNCS, vol. 8430, pp. 285–299. Springer (2014)

3. Baier, C., Dubslaff, C.: From verification to synthesis under cost-utility constraints. SIGLOG News **5**(4), 26–46 (2018)
4. Baier, C., Klein, J., Klüppelholz, S., Wunderlich, S.: Maximizing the conditional expected reward for reaching the goal. In: TACAS (2), LNCS, vol. 10206, pp. 269–285 (2017)
5. Baier, C., Klein, J., Leuschner, L., Parker, D., Wunderlich, S.: Ensuring the reliability of your model checker: Interval iteration for Markov decision processes. In: CAV (1), LNCS, vol. 10426, pp. 160–180. Springer (2017)
6. Barrett, L., Narayanan, S.: Learning all optimal policies with multiple criteria. In: ICML, AICPS, vol. 307, pp. 41–47. ACM (2008)
7. Berthon, R., Randour, M., Raskin, J.F.: Threshold constraints with guarantees for parity objectives in Markov decision processes. In: ICALP, LIPIcs, vol. 80, pp. 121:1–121:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2017)
8. Brázdil, T., Brozek, V., Chatterjee, K., Forejt, V., Kucera, A.: Two views on multiple mean-payoff objectives in Markov decision processes. LMCS **10**(1) (2014)
9. Brázdil, T., Chatterjee, K., Chmelik, M., Forejt, V., Kretínský, J., Kwiatkowska, M.Z., Parker, D., Ujma, M.: Verification of Markov decision processes using learning algorithms. In: ATVA, LNCS, vol. 8837, pp. 98–114. Springer (2014)
10. Brázdil, T., Chatterjee, K., Forejt, V., Kucera, A.: Trading performance for stability in Markov decision processes. J. Comput. Syst. Sci. **84**, 144–170 (2017)
11. Bresina, J.L., Jónsson, A.K., Morris, P.H., Rajan, K.: Activity planning for the Mars exploration rovers. In: ICAPS, pp. 40–49. AAAI (2005)
12. Bryce, D., Cushing, W., Kambhampati, S.: Probabilistic planning is multi-objective. Technical Report, Arizona State Univ, CSE (2007)
13. Cao, Z., Guo, H., Zhang, J., Oliehoek, F.A., Fastenrath, U.: Maximizing the probability of arriving on time: a practical q-learning method. In: AAAI, pp. 4481–4487. AAAI Press (2017)
14. Chatterjee, K., Chmelik, M., Gupta, R., Kanodia, A.: Optimal cost almost-sure reachability in POMDPs. Artif. Intell. **234**, 26–48 (2016)
15. Chatterjee, K., Majumdar, R., Henzinger, T.A.: Markov decision processes with multiple objectives. In: STACS, LNCS, vol. 3884, pp. 325–336. Springer (2006)
16. Chen, T., Forejt, V., Kwiatkowska, M., Simaitis, A., Wiltsche, C.: On stochastic games with multiple objectives. In: MFCS, LNCS, vol. 8087, pp. 266–277. Springer (2013)
17. Cheng, L., Subrahmanian, E., Westerberg, A.W.: Multiobjective decision processes under uncertainty: applications, problem formulations, and solution strategies. Ind. Eng. Chem. Res. **44**(8), 2405–2415 (2005)
18. Christman, A., Cassamano, J.: Maximizing the probability of arriving on time. In: ASMTA, LNCS, vol. 7984, pp. 142–157. Springer (2013)
19. Dai, P., Mausam, Weld, D.S., Goldsmith, J.: Topological value iteration algorithms. J. JAIR **42**, 181–209 (2011)
20. Dehnert, C., Junges, S., Katoen, J.P., Quatmann, T., Volk, M.: Storm website (2018). http://stormchecker.org
21. Dehnert, C., Junges, S., Katoen, J.P., Volk, M.: A Storm is coming: a modern probabilistic model checker. In: CAV (2), LNCS, vol. 10427, pp. 592–600. Springer (2017)
22. Eastwood, R., Alexander, R., Kelly, T.: Safe multi-objective planning with a posteriori preferences. In: HASE, pp. 78–85. IEEE Computer Society (2016)
23. Etessami, K., Kwiatkowska, M., Vardi, M.Y., Yannakakis, M.: Multi-objective model checking of Markov decision processes. LMCS **4**(4) (2008)
24. Feng, L., Wiltsche, C., Humphrey, L., Topcu, U.: Controller synthesis for autonomous systems interacting with human operators. In: ICCPS, pp. 70–79. ACM (2015)
25. Forejt, V., Kwiatkowska, M., Parker, D.: Pareto curves for probabilistic model checking. In: ATVA, LNCS, vol. 7561, pp. 317–332. Springer (2012)
26. Forejt, V., Kwiatkowska, M.Z., Norman, G., Parker, D.: Automated verification techniques for probabilistic systems. In: SFM, LNCS, vol. 6659, pp. 53–113. Springer (2011)
27. Haddad, S., Monmege, B.: Reachability in MDPs: refining convergence of value iteration. In: RP, LNCS, vol. 8762, pp. 125–137. Springer (2014)
28. Hahn, E.M., Hartmanns, A.: A comparison of time- and reward-bounded probabilistic model checking techniques. SETTA, LNCS **9984**, 85–100 (2016)
29. Hahn, E.M., Hartmanns, A., Hermanns, H., Katoen, J.P.: A compositional modelling and analysis framework for stochastic hybrid systems. Formal Methods in Syst. Des. **43**(2), 191–232 (2013)

30. Hahn, E.M., Hashemi, V., Hermanns, H., Lahijanian, M., Turrini, A.: Multi-objective robust strategy synthesis for interval Markov decision processes. In: QEST, LNCS, vol. 10503, pp. 207–223. Springer (2017)
31. Hartmanns, A., Hermanns, H.: The Modest Toolset: An integrated environment for quantitative modelling and verification. In: TACAS, LNCS, vol. 8413, pp. 593–598. Springer (2014)
32. Hartmanns, A., Junges, S., Katoen, J.P., Quatmann, T.: Multi-cost bounded reachability in MDP. In: TACAS, LNCS, vol. 10806, pp. 320–339. Springer (2018). https://doi.org/10.1007/978-3-319-89963-3_19
33. Hartmanns, A., Junges, S., Katoen, J.P., Quatmann, T.: Multi-cost bounded tradeoff analysis in MDP—Artifact. Zenodo (2020). https://doi.org/10.5281/zenodo.3894716
34. Hartmanns, A., Kaminski, B.L.: Optimistic value iteration. In: CAV, Lecture Notes in Computer Science, vol 12225, pp. 488–511. Springer (2020). https://doi.org/10.1007/978-3-030-53291-8_26
35. Hou, P., Yeoh, W., Varakantham, P.: Revisiting risk-sensitive MDPs: New algorithms and results. In: ICAPS. AAAI (2014)
36. Junges, S., Jansen, N., Dehnert, C., Topcu, U., Katoen, J.: Safety-constrained reinforcement learning for MDPs. In: TACAS, LNCS, vol. 9636, pp. 130–146. Springer (2016)
37. Klein, J., Baier, C., Chrszon, P., Daum, M., Dubslaff, C., Klüppelholz, S., Märcker, S., Müller, D.: Advances in probabilistic model checking with PRISM: variable reordering, quantiles and weak deterministic Büchi automata. STTT pp. 1–16 (2017)
38. Kolobov, A., Mausam, Weld, D.S.: A theory of goal-oriented MDPs with dead ends. In: UAI, pp. 438–447. AUAI Press (2012)
39. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: CAV, LNCS, vol. 6806, pp. 585–591. Springer (2011)
40. Kwiatkowska, M., Norman, G., Parker, D.: The PRISM benchmark suite. In: QEST, pp. 203–204. IEEE CS Press (2012)
41. Kwiatkowska, M.Z., Norman, G., Sproston, J.: Probabilistic model checking of the IEEE 802.11 wireless local area network protocol. In: PAPM-PROBMIV, LNCS, vol. 2399, pp. 169–187. Springer (2002)
42. Lacerda, B., Parker, D., Hawes, N.: Multi-objective policy generation for mobile robots under probabilistic time-bounded guarantees. In: ICAPS, pp. 504–512. AAAI Press (2017)
43. Lankaites Pinheiro, R., Landa-Silva, D., Atkin, J.: A technique based on trade-off maps to visualise and analyse relationships between objectives in optimisation problems. J. Multi-Criteria Decis. Anal. **24**(1–2), 37–56 (2017)
44. Laroussinie, F., Sproston, J.: Model checking durational probabilistic systems. In: FoSSaCS, LNCS, vol. 3441, pp. 140–154. Springer (2005)
45. Norman, G., Parker, D., Kwiatkowska, M.Z., Shukla, S.K.: Evaluating the reliability of NAND multiplexing with PRISM. IEEE Trans. CAD of Integ. Circuits Syst. **24**(10), 1629–1637 (2005)
46. Puterman, M.L.: Markov Decision Processes. Wiley, HobokenD (1994)
47. Quatmann, T., Junges, S., Katoen, J.P.: Markov automata with multiple objectives. In: CAV (1), LNCS, vol. 10426, pp. 140–159. Springer (2017)
48. Quatmann, T., Katoen, J.P.: Sound value iteration. In: CAV, LNCS, vol. 10981, pp. 643–661. Springer (2018)
49. Randour, M., Raskin, J.F., Sankur, O.: Percentile queries in multi-dimensional Markov decision processes. FMSD **50**(2–3), 207–248 (2017)
50. Reiter, M.K., Rubin, A.D.: Crowds: anonymity for web transactions. ACM Trans. Inf. Syst. Secur. **1**(1), 66–92 (1998)
51. Roijers, D.M., Vamplew, P., Whiteson, S., Dazeley, R.: A survey of multi-objective sequential decision-making. J. Artif. Intell. Res. **48**, 67–113 (2013)
52. Sardar, M.U., Dubslaff, C., Klüppelholz, S., Baier, C., Kumar, A.: Performance evaluation of thermal-constrained scheduling strategies in multi-core systems. In: EPEW, *LNCS*, vol. 12039, pp. 133–147. Springer (2019). https://doi.org/10.1007/978-3-030-44411-2_9
53. Steinmetz, M., Hoffmann, J., Buffet, O.: Goal probability analysis in probabilistic planning: exploring and enhancing the state of the art. J. Artif. Intell. Res. **57**, 229–271 (2016)
54. Stoelinga, M., Vaandrager, F.W.: Root contention in IEEE 1394. In: ARTS Formal Methods for Real-Time and Probabilistic Systems, LNCS, vol. 1601, pp. 53–74. Springer (1999)
55. Teichteil-Königsbuch, F.: Stochastic safest and shortest path problems. In: AAAI. AAAI Press (2012)
56. The International Probabilistic Planning Competition. http://www.icaps-conference.org/index.php/Main/Competitions
57. Ummels, M., Baier, C.: Computing quantiles in Markov reward models. In: FOSSACS, *LNCS*, vol. 7794, pp. 353–368. Springer (2013)

58. Vamplew, P., Dazeley, R., Berry, A., Issabekov, R., Dekker, E.: Empirical evaluation methods for multi-objective reinforcement learning algorithms. Mach. Learn. **84**(1–2), 51–80 (2011)
59. Yu, S.X., Lin, Y., Yan, P.: Optimization models for the first arrival target distribution function in discrete time. J. Math. Anal. Appl. **225**(1), 193–223 (1998)