N.N. manuscript No.
(will be inserted by the editor)

Computational Bounds for Elevator Control Policies by Large Scale Linear Programming

Stefan Heinz · Jörg Rambau · Andreas Tuchscherer

Received: date / Accepted: date

Abstract We computationally assess policies for the elevator control problem by a new column-generation approach for the linear programming method for discounted infinite-horizon Markov decision problems. By analyzing the optimality of given actions in given states, we were able to provably improve the well-known nearest-neighbor policy. Moreover, with the method we could identify an optimal parking policy. This approach can be used to detect and resolve weaknesses in particular policies for Markov decision problems.

Keywords Markov decision problem · bounds · large scale · column generation · approximation · performance guarantee

Mathematics Subject Classification (2000) MSC 90C40 · MSC 90C05 · 90C06

Partially supported by the DFG Research Center MATHEON "Mathematics for key technologies" in Berlin.

Stefan Heinz
Zuse-Institute Berlin
Tel.: +49 30 84185-428
Fax: +49 30 84185-269
E-mail: heinz@zib.de

Jörg Rambau
LS Wirtschaftsmathematik
Universität Bayreuth
Tel.: +49 921 55-7350
Fax.: +49 921 55-7352
E-mail: joerg.rambau@uni-bayreuth.de

Andreas Tuchscherer
Zuse-Institute Berlin
Tel.: +49 30 84185-353
Fax: +49 30 84185-269
E-mail: andreas.tuchscherer@zib.de

1 Introduction

Consider a simple cargo elevator system where a single elevator can carry at most one pallet from one floor to another without preemption. Future requests are not known, stochastic information maybe available. The goal is to find a policy to control the elevator system in such a way that the average or the maximal waiting time is minimized in expectation. This is the seemingly easiest of the many elevator control problems that have been studied the literature [28,6,16,23,21,22].

Even for this simple elevator problem an optimal policy is still unknown – the curse of dimensionality renders impossible any direct computation of an optimal policy for a corresponding Markov decisions problem (MDP) model (value iteration, policy iteration, or linear programming). Whereas practically satisfying policies could be identified, rigorous knowledge is scarce about which policy out of two is provably better or how close is a policy to an unknown optimal policy.

Our goal in this paper is to employ

- induced MDPs with "small" state spaces that can be handled and that yield upper and lower bounds for the original large MDP,
- a column generation framework generating increasingly suitable induced MDPs an their cost-to-go functions
- application and state dependent bounds for the future evolution of the system beyond a certain point to be utilized inside the column generation procedure

to obtain statements like the following:

- Policy A is better than Policy B when starting in State i.
- Policy A is not optimal.
- Action a in State i is not optimal.
- Policy A starting in State i is at most $x\,\%$ more expensive than an optimal policy

Our algorithm employs the linear programming characterization of optimal policies in discounted MDPs. It starts with a small part of the state space and adds states driven by the reduced-cost criterion from linear programming. The reduced cost of state variables is the additional information that comes for free in the linear programming setting. Our tool exploits this extra-information.

1.1 Related Work

A broad field of methods targeting large-scale MDPs (and generalizations) where exact methods become infeasible is *approximate dynamic programming (ADP)* [26, 30,5], which evolved in the computer science community under the name *reinforcement learning*. Contrary to the classical computational methods mentioned above, an advantage of many techniques in this area is that an explicit model of the environment, i. e., a precise specification of the MDP, is often not required. Instead, a simulator of the system can be employed. Similar to simulation, there is virtually no limit on the complexity of the state and transition structure. We refer to the books [26,30,5] for details concerning ADP.

A special version of ADP is *approximate linear programming (ALP)*, where the cost-to-go function is approximated by a weighted combination of basis functions. The weights are then computed by linear programming techniques. There are a few methods that provide performance guarantees, e. g., [12, 11, 14, 9].

The main difference of our effort to ADP and ALP is that we do not need to assume that the cost-to-go can be closely approximated by a space of prescribed basis functions. Our approach is complementary to ADP/ALP in the sense that we provide a tool to analyze policies in the original problem. In fact, policies stemming from ADP/ALP can be analyzed by our method to find bounds on their expected performance.

Our main tool, column generation, has been studied also in the context of ALP, see e. g., [1, 32]. We have not seen yet column generation as a tool for the exact LP formulation, which can in principle produce bounds that are independent of an approximation architecture.

The approach described in the literature that yields results closest to ours for the exact cost-to-go function is a *sparse sampling algorithm* proposed by Kearns et al. [24]. The authors also give theoretical bounds on the necessary size of a subset of the state space that is needed by their approach in order to obtain an ε-approximation, see Remark 3 on Page 15. However, for the applications we aim at, their bounds are substantially weaker than ours.

Other approaches to locally explore the state space have been proposed by Dean et al. [7] and Barto et al. [3]. The former employs policy iteration with a concept of locality similar to ours. This way, their method comes closest to our approach concerning the algorithm used. However, the method does not provide any approximation guarantees.

1.2 Our contribution

With our method we can show rigorously for an 8-floor elevator system with sparse requests that if the objective is to minimize the average waiting, the nearest-neighbor policy NN is better than many other policies. It is, however, provably non-optimal. This adds theoretical learnings to the simulation knowledge from [17]. Non-optimality is already implied by the property that NN never moves the elevator in an empty system. By evaluating this single action in the empty system state with our tool, we can guarantee that all policies that do not move in the empty system are suboptimal. We present a new policy NNPARK-f that positions the elevator optimally when no request is in the system.

In a similar fashion, we improve NN to a better policy NNMAXPARK-f when the goal is to minimize the maximum waiting time among all requests. And for this objective, we can show with our tool that NN is one of the weakest policies.

Although rigorous computational proofs could only be obtained for a relatively small discount factor of 0.8 emphasizing short-term effects, the new policies – with improvements guided by our analysis method – outperformed the original policies by a large margin also in long-term simulations. Most notably, NNMAXPARK-f showed the most balanced behaviour with good results for both average and maximal waiting times.

1.3 Outline of the Paper

The paper is organized as follows: In Section 2 we fix an MDP model of our elevator control problem. Section 3 introduces the three corner stones of our method. In Section 4, we present the new knowledge obtained for the elevator control problem. We conclude in Section 5.

2 Formal Problem Statement

In the following, we introduce a Markov decision problem (MDP) formulation of the elevator problem. We then briefly introduce the policies under investigation. In order to settle on notation, we repeat what a Markov Decision Problem is (see [4,27] for background on MDP-theory). A *Markov decision process* is a tuple $M = (\mathbb{S}, \mathbb{A}, p, c)$ with:

- $\mathbb{S}$ is a finite set of *states*.
- $\mathbb{A}$ is a mapping specifying for each state $i \in \mathbb{S}$ a non-empty and finite set $\mathbb{A}(i)$ of possible *actions* at state i.
- For all states $i, j \in \mathbb{S}$, the mapping $p_{ij} \colon \mathbb{A}(i) \to [0,1]$ gives the *transition probability* $p_{ij}(a)$ conditioned on the assumption that the system is in state i and moves to state j when using action $a \in \mathbb{A}(i)$. For each state $i \in \mathbb{S}$ and each action $a \in \mathbb{A}(i)$, we have $\sum_{j \in \mathbb{S}} p_{ij}(a) = 1$.
- For all $i \in \mathbb{S}$, the mapping $c_i \colon \mathbb{A}(i) \times \mathbb{S} \to \mathbb{R}_+$ specifies the *stage cost* $c_i(a, j)$ when action $a \in \mathbb{A}(i)$ is chosen and the system moves to state $j \in \mathbb{S}$. The *expected stage cost* of using action $a \in \mathbb{A}(i)$ at state $i \in \mathbb{S}$ is denoted by $c_i(a) := \sum_{j \in \mathbb{S}} p_{ij}(a) c_i(a, j)$.

A *policy* for M is a mapping $\pi : \mathbb{S} \to \mathbb{A}(\mathbb{S})$. It is *feasible* if $\pi(i) \in \mathbb{A}(i)$. Let P_M denote the set of all feasible policies for M. Let $M = (\mathbb{S}, \mathbb{A}, p, c)$ be a Markov decision process and let $\alpha \in [0,1)$. The *total expected α-discounted cost* of a policy π for M for an initial state $i \in \mathbb{S}$ is defined by

$$\begin{aligned} v_i^\alpha(\pi) &:= \sum_{t=0}^{\infty} \mathbb{E}_{i,\pi}[\alpha^t \cdot c_{X_t}(\pi(X_t))] \\ &= \sum_{t=0}^{\infty} \alpha^t \sum_{j \in \mathbb{S}} \mathbb{P}_{i,\pi}[X_t = j] \cdot c_j(\pi(j)) \end{aligned}$$

where X_t is a random variable stating the possible state at time t, $\mathbb{P}_{i,\pi}$ and $\mathbb{E}_{i,\pi}$ giving the probability of being in a certain state and having an expected value after starting at the initial state i and applying policy π. Let $V^\alpha : P_M \to \mathbb{R}^{\mathbb{S}}$ be the value vector function defined for each policy $\pi \in P_M$ by the value vector $v^\alpha(\pi)$ with elements $v_i^\alpha(\pi)$ for each $i \in \mathbb{S}$ as given above. The combination (M, V^α) of M and the value vector function V^α is called an *α-discounted cost Markov Decision Problem*, or short *discounted MDP*, and is denoted for short as (M, α). We denote with v^α the *optimal value vector* which is $v_i^\alpha = \min_{\pi \in P_M} v_i^\alpha(\pi)$ for all $i \in \mathbb{S}$. A policy π^* is *optimal* for (M, α) if $v_i^\alpha(\pi^*) = v^\alpha$.

In order to formulate a Markov decision process model for the elevator control problem, we consider the following situation. The system operates a set of elevators $E = \{1,\dots,n_E\}$ in a building with a set of floors $F = \{1,\dots,n_F\}$. Each elevator can load at most one request. At each floor there is a waiting area that accommodates at most $q \in \mathbb{N} \cup \{\infty\}$ transport requests. We limit our considerations to a discrete time model. At each time slot the current situation is described by the following data:

- Each elevator $e \in E$ is situated at one floor $f_e \in E$ and is either loaded or empty.
- For each floor $f \in F$, there exists a sequence $\sigma_f = r_1,\dots,r_{n_f}$ of waiting requests, where $n_f \in \{0,\dots,q\}$ is their number. Moreover, each request r_k for $k \in \{1,\dots,n_f\}$ is of the form $r_k = (f, f_k, w_k)$, where $f_k \in F \setminus \{f\}$ is its destination floor and $w_k \in \mathbb{N}_0$ is the waiting time of request r_k so far with $w_k \geq w_{k+1}$. Denote by $w_{\sigma_f} := w_1$ the maximum waiting time of a request in sequence σ_f if it is non-empty, and let Σ_f be the set of all possible sequences at floor f.

Feasible Actions If elevator $e \in E$ is loaded, let $d_e \in F$ be the destination floor of the request being transported, and let $d_e = 0$ otherwise. In one time unit an elevator $e \in E$ can execute exactly one of the following operations:

WAIT at its current floor f_e,
MOVE_UP one floor if $f_e < n_F$ (this is the only feasible action if $d_e > f_e$),
MOVE_DOWN one floor if $f_e > 1$ (this is the only feasible action if $0 < d_e < f_e$),
LOAD the next request at the current floor f_e if $d_e = 0$ and $\sigma_{f_e} \neq \emptyset$, i. e., the elevator is empty and there is at least one request waiting at floor f_e, or
DROP the loaded request if $f_e = d_e$, i. e., the elevator is loaded and its current floor equals the destination floor of the loaded request (this is the only feasible action if $d_e = f_e$).

State Space A state $i \in \mathbb{S}$ in the Markov decision process model $(\mathbb{S}, \mathbb{A}, p, c)$ is of the following form:

$$i = (w_{\max}, (\sigma_f)_{f\in F}, (f_e, d_e)_{e\in E}),$$

where $w_{\max} \in \mathbb{N}_0$ specifies the maximum waiting time of a request so far. Moreover, a state captures all data concerning waiting requests and possibly loaded requests as well as the positions of the elevators. We will also denote the parameters of a state i by $w_{\max}(i)$, $\sigma_f(i)$ for each $f \in F$, and $f_e(i), d_e(i)$ for each $e \in E$. The resulting state space $\mathbb{S}$ is given by:

$$\begin{aligned}\mathbb{S} = \{(w_{\max}, (\sigma_f)_{f\in F}, ((f_e, d_e)_{e\in E}) \mid\ & w_{\max} \in \mathbb{N}_0, w_{\max} \geq w_{\sigma_f}\ \forall f \in F\colon \sigma_f \neq \emptyset,\\ & \sigma_f \in \Sigma_f\ \forall f \in F,\\ & (f_e, d_e) \in F \times (\{0\} \cup F)\ \forall e \in E\}.\end{aligned}$$

As the stored waiting times in a state may become arbitrarily large even if the waiting queue length q is bounded, the state space $\mathbb{S}$ is infinite.

Each action in $\mathbb{A}(i)$ for a state $i \in \mathbb{S}$ is composed of one control decision $a(e)$ for each elevator $e \in E$, i. e., an action $a \in \mathbb{A}(i)$ is of the form $a = (a(e_1),\dots,a(e_{n_E}))$.

The control decision of an elevator may be any one of the operations mentioned above: WAIT, MOVE_UP, MOVE_DOWN, LOAD, DROP. However, we assume that a loaded elevator $e \in E$ immediately serves the request being transported: if $f_e < d_e$ or $f_e > d_e$, the elevator e will move up or down, respectively, and if $f_e = d_e$, the request will be dropped. This means that for a loaded elevator the set of feasible actions contains only one action. Thus, all requests are served without preemption.

Transitions In our model each transition between two states is assumed to last exactly one time step, moving from one time slot to the next one. Moreover, we assume that at most one new request is released at each time slot. Hence, all queued requests have a unique waiting time. We describe possible state transitions only for the case of a single elevator since the general case is obtained by handling the control decisions of all elevators consecutively. If no new request arrives, the deterministic successor $j \in \mathbb{S}$ of a state $i \in \mathbb{S}$ when using action $a = (a(e)) \in \mathbb{A}(i)$ is given by:

- The maximum waiting time at state j equals:

$$w_{\max}(j) = \max\{w_{\max}(i), \max_{f \in F:\, \sigma_f(j) \neq \emptyset} w_{\sigma_f(j)}\}.$$

- For each floor $f \in F \setminus \{f_e\}$, we have $\sigma_f(j) = \bar{r}_1, \ldots, \bar{r}_{n_{f_e}}$ with $\bar{r}_k = (f, f_k, w_k + 1)$ where f_k and w_k are the destination and the waiting time of the k-th request in $\sigma_f(i)$. If $a(e) = \text{LOAD}$, the update for the waiting queue at floor f_e is $\sigma_{f_e}(j) = \bar{r}_2, \ldots, \bar{r}_{n_{f_e}}$. Otherwise, we have $\sigma_{f_e}(j) = \bar{r}_1, \ldots, \bar{r}_{n_{f_e}}$. Again $\bar{r}_k = (f_e, f_k, w_k + 1)$.
- The current floor and load of elevator e are updated by:

$$(f_e(j), d_e(j)) = \begin{cases} (f_e(i), d_e(i)), & \text{if } a(e) = \text{WAIT}, \\ (f_e(i) + 1, d_e(i)), & \text{if } a(e) = \text{MOVE\_UP}, \\ (f_e(i) - 1, d_e(i)), & \text{if } a(e) = \text{MOVE\_DOWN}, \\ (f_e(i), f_1), & \text{if } a(e) = \text{LOAD}, \\ (f_e(i), 0), & \text{if } a(e) = \text{DROP}, \end{cases}$$

  where f_1 denotes the destination of the first request $r_1 = (f_e, f_1, w_1)$ in the sequence $\sigma_{f_e}(i)$ in the loading case.

When a new request $r = (a, b, 0)$ is released at a floor $a \in F$ with destination floor $b \in F \setminus \{a\}$, we obtain the successor $(w_{\max}(j), (\sigma'_f)_{f \in F}, (f_e(j), d_e(j)))$ of state i. In this state, we have $\sigma'_f = \sigma_f(j)$ for each floor $f \in F \setminus \{a\}$ and

$$\sigma'_a = \begin{cases} \sigma_a(j) + r, & \text{if } |\sigma_a(j)| < q, \\ \sigma_a(j), & \text{if } |\sigma_a(j)| = q, \end{cases}$$

where $\sigma_a(j) + r$ denotes the sequence with request r added to $\sigma_a(j)$.

The transition probabilities p are defined by a two step process. Firstly, we have a fixed probability that a new request is released at a state transition (Bernoulli distribution). If that is the case, the start and destination floor of the new request are determined according to some probability distribution in the second step.

Depending on the used objective function, the stage costs are given as follows. If we focus on minimizing the maximum waiting time of a request, it is always assumed that the waiting queues are unbounded, i. e., $q = \infty$. In this case, the stage cost $c_i(a, j) = c_i^{\max}(a, j)$ associated with states $i, j \in \mathbb{S}$ and action $a \in \mathbb{A}(i)$ equals the increase of the maximum waiting time due to action a:

$$c_i^{\max}(a, j) = w_{\max}(j) - w_{\max}(i).$$

Notice that the total sum of stage costs for the transitions of an (i, j)-path equals the total increase of the maximum waiting time in this sequence of states.

For minimizing the average waiting time, we assume the waiting queue length to be bounded, i. e., $q < \infty$. This way, we can work with the sum of all waiting times in our MDP formulation. Otherwise, the stage cost in such an MDP, which is the increment of the sum of all waiting times in a single time slot, may become infinite.

Whenever a request is released at a floor $f \in F$ where the waiting queue is full, i. e., $|\sigma_f| = q$, the request is rejected from the system at a penalty cost of $c_p \geq 1$. For each floor $f \in F$, let $0 \leq p_f \leq 1$ be the probability that a request is released at some time slot at floor f. Given states $i, j \in \mathbb{S}$ and an action $a \in \mathbb{A}(i)$, let $j' \in \mathbb{S}$ be the successor of i using action a if no new request arrives. Then, the stage cost $c_i(a, j) = c_i^{\text{avg}}(a, j)$ is defined as the sum of all requests waiting at state i that are not loaded by action a plus the expected penalty cost:

$$c_i^{\text{avg}}(a, j) = \sum_{f \in F} |\sigma_f(i)| - |\{e \in E \mid a(e) = \textsf{LOAD}\}| + c_p \cdot \sum_{f \in F \colon |\sigma_f(j')| = q} p_f$$

In the case the waiting queues of the states j and j' differ, a new request has been released at a floor where the waiting queue was not full w. r. t. state j'. Thus, the transition does not involve a penalty cost.

Notice that $c_i^{\text{avg}}(a, j)$ equals the increase of the sum of all waiting times plus the expected penalty cost. Thus the sum of the expected stage costs for all transitions of an (i, j)-path equals the sum of all accumulated waiting times and expected penalty costs during the associated time period. Minimizing this objective for a finite sequence of requests is equivalent to minimizing the average waiting time.

We want to point out, that the basic Markov decision process model we consider here differs substantially from the one used by Crites and Barto [6].

Originally the goal is to find an optimal policy for an MDP. Our goal is to obtain the following for the elevator control MDP above: Given a policy, and an $\varepsilon > 0$, find ε-exact performance guarantees for single start states, maybe relative to an unknown optimal policy or relative to some other policy. That is, more formally:

Problem 1 Given a policy π, a state i_0 with $v_{i_0}^{\alpha} > 0$, and an $\varepsilon > 0$, find in state i_0 a lower bound $\underline{v}_{i_0}$ for the optimal cost and an upper bound $\overline{v}_{i_0}(\pi)$ for the cost of π such that

$$\frac{\overline{v}_{i_0}(\pi) - \underline{v}_{i_0}}{\underline{v}_{i_0}} \leq \varepsilon. \qquad \text{(Relative Performance Guarantee)}$$

Alternatively, find in state i_0 a lower bound $\underline{v}_{i_0}(\pi)$ for the cost of π and an upper bound $\overline{v}_{i_0}$ for the optimal cost such that

$$\underline{v}_{i_0}(\pi) > \overline{v}_{i_0}. \qquad \text{(Non-Optimality Certificate)}$$

In this paper, we present an algorithm that can provide such bounds and related data without necessarily touching all states. States used for the computation are selected dynamically, dependent on the individual data of the instance. The algorithm detects automatically when the desired guarantee can be given and stops with a proven result.

We want to generate information of this type for the following policies that have been of interest, e. g., in [2,17,18]

FIRSTINFIRSTOUT (FIFO) Serve the request with the smallest current waiting time next. This request is unique by our assumption that at most one request is released at each time slot.

NEARESTNEIGHBOR (NN) Determine a waiting request whose start floor is located nearest to the current floor of the elevator. If there exists a unique request with this property, serve it next. Otherwise, such a request exists in both directions. Then, serve the one with smaller floor number next.

REPLAN Compute a schedule minimizing the makespan (without returning to some origin), i. e., the time needed to serve all waiting requests, and serve the requests according to this schedule. We implemented a branch-and-bound method to compute these schedules.

IGNORE As long as a schedule is available, serve the waiting requests accordingly. If no schedule is available, do the same as the policy REPLAN and store the schedule. The policy IGNORE requires a modified MDP where each state encodes a schedule containing a (possibly empty) subset of the waiting requests. Moreover, if for some state this schedule is empty and a request is waiting, each associated action has a second component that sets the schedule for all waiting requests.

Remark 1 The policies REPLAN and IGNORE may appear counter-intuitive at first glance: they repeatedly opimize the makespan instead of the original objective. This is motivated by the fact that for those policies there are positive theoretical results known (see, e.g., [2,19,25]). In contrast to this, for the variants optimizing the original objectives there are no positive results known yet. Moreover, examples are known where REPLAN with original objective defers some of the requests infinitely long. Therefore, we chose to investigate REPLAN and IGNORE as described above.

3 Methods

We use the following three ingredients:

- *induced MDPs* with "small" state spaces that can be handled and that yield upper and lower bounds for the original large MDP,
- a *column generation* framework generating increasingly suitable induced MDPs an their cost-to-go functions
- *state dependent bounds* for the future evolution of the system beyond a certain point to be utilized inside the column generation procedure

3.1 Induced MDPs

In this section, we derive from a given MDP new MDPs whose value functions:

– can be computed easier,
– yield bounds for the value function of the original MDP.

Our method also aims at classifying single actions into optimal and not-optimal, not only policies.

Definition 1 (Optimal actions) Let (M,α) be an discounted MDP with $\alpha \in [0,1)$. A possible action $a \in \mathbb{A}(i)$ at a state $i \in \mathbb{S}$ is called *optimal* if there exists an optimal deterministic policy π for M such that $\pi(i) = a$.

The classical methods for computing the optimal value vector v^α of a discounted MDP include *value iteration*, *policy iteration*, and *linear programming*. For details and possible variants and extensions of the methods, see [27, chapter 6], [15, chapter 2.3], or [4, volume 2, chapter 1.3].

The central theorem concerning the linear programming method for computing the optimal value vector of a discounted MDP reads as follows.

Theorem 1 (See, e.g., [4, Volume 2, Section 1.3.4]) *The optimal value vector $v^\alpha \in \mathbb{R}^{\mathbb{S}}$ of a discounted MDP (M,α) equals the unique optimal solution v of the following linear program:*

$$\begin{aligned} \max \quad & \sum_{i\in\mathbb{S}} v_i && && (\mathrm{P}^\Sigma)\\ \text{subject to} \quad & v_i - \alpha \sum_{j\in\mathbb{S}} p_{ij}(a) v_j \le c_i(a) && \forall i \in \mathbb{S}\ \forall a \in \mathbb{A}(i)\\ & v_i \in \mathbb{R} && \forall i \in \mathbb{S}. \end{aligned}$$

Therefore, one can obtain the optimal value vector by solving the linear program (P^Σ). This linear programming formulation was first proposed by d'Epenoux [8] and has been the starting point for several approaches, e. g., see [29,12,13].

We define the matrix Q with rows for each $(i,a) \in \mathbb{S}\times\mathbb{A}$ and columns for each state $j \in \mathbb{S}$ by:

$$Q_{(i,a),j} = \begin{cases} 1-\alpha p_{ij}(a), & \text{if } i = j,\\ -\alpha p_{ij}(a), & \text{if } i \neq j. \end{cases}$$

The components of the vector c are given by:

$$c_{ia} = c_i(a)$$

for each $(i,a) \in \mathbb{S}\times\mathbb{A}$. Now the linear program (P^Σ) can be written as:

$$\begin{aligned} \max \quad & \mathbb{1}^t v && (\mathrm{P}^\Sigma)\\ \text{subject to} \quad & Qv \le c\\ & v \in \mathbb{R}^{\mathbb{S}}, \end{aligned}$$

where $\mathbb{1}^t = (1,1,\ldots,1)$ denotes the all-ones vector.

The approximation algorithm to be proposed is motivated by the fact that for the huge state spaces arising in MDPs modeling practical problems, it is currently

impossible to solve the associated linear program (P^{Σ}) in reasonable time. Our idea is to evaluate the value vector at one particular state $i_0 \in \mathbb{S}$ alone. Since we are only interested in $v^{\alpha}_{i_0}$, we can restrict the objective function of (P^{Σ}) by maximizing the value v_{i_0} only:

$$\begin{aligned} \max \;& v_{i_0} \\ \text{subject to } & Qv \leq c \\ & v \in \mathbb{R}^{\mathbb{S}} \end{aligned} \tag{P^{i_0}}$$

In contrast to (P^{Σ}), there does not exist a unique solution for the linear program (P^{i_0}) in general for the following reasons. On the one hand, there may be states in $\mathbb{S}$ that cannot be reached from i_0. On the other hand, there are typically some actions that are not optimal. Such a state $j \in \mathbb{S}$, that is either not reached at all or only reached via non-optimal actions, is not required to have a maximized value v_j in order to maximize v_{i_0}, i. e., the objective function of (P^{i_0}). The value v_j may even be negative in an optimal solution.

Similar to the original linear programming formulation, solving the linear program (P^{i_0}) is still infeasible considering the huge state spaces for practical applications. In order to obtain a linear program that is tractable independently of the size of the state space $\mathbb{S}$, we reduce the set of variables and constraints in the linear program (P^{i_0}) by taking into account only a restricted state space. Given a subset of states $S \subseteq \mathbb{S}$ with $i_0 \in S$, consider the submatrix $Q^S \in \mathbb{R}^{(S\times\mathbb{A})\times S}$ of the constraint matrix Q consisting of all rows (i,a) with $i \in S$ and all columns j with $j \in S$. Moreover, let $c^S \in \mathbb{R}^{S\times\mathbb{A}}$ be the subvector of vector c consisting of all the components with indices (i,a) satisfying $i \in S$.

It is obvious that the α-discounted cost-to-go in any state is between zero and $\frac{c_{\max}}{1-\alpha}$, where $c_{\max}$ is the maximal stage cost. In order to make use of more sophisticated knowledge we assume that we are given application-specific, state-dependent lower and upper bounds $v^{\alpha}_{\min}(j)$ and $v^{\alpha}_{\max}(j)$, respectively, for the cost-to-go functions $0 \leq v^{\alpha}_{\min}(j) \leq v^{\alpha}_{j} \leq v^{\alpha}_{\max}(j) \leq \frac{c_{\max}}{1-\alpha}$ in State $j \in \mathbb{S}\setminus S$. From this information, we construct lower and upper bound vectors for each $(i,a) \in S \times \mathbb{A}$:

$$\underline{r}^S_{ia} = \alpha \sum_{j\in\mathbb{S}\setminus S} p_{ij}(a) v^{\alpha}_{\min}(j), \qquad \overline{r}^S_{ia} = \alpha \sum_{j\in\mathbb{S}\setminus S} p_{ij}(a) v^{\alpha}_{\max}(j).$$

With these data, we can derive the following straight-forward bounding scheme:

Lemma 1 *Define*

$$\begin{aligned} \underline{v} &:= \max\{\mathbb{1}^t v | Q^S v \leq c^S + \underline{r}^S, v \in \mathbb{R}^S\}, \\ \overline{v} &:= \max\{\mathbb{1}^t v | Q^S v \leq c^S + \overline{r}^S, v \in \mathbb{R}^S\}, \\ \underline{v}_{i_0} &:= \max\{v_{i_0} | Q^S v \leq c^S + \underline{r}^S, v \in \mathbb{R}^S\}, \qquad (L^{i_0}_S) \\ \overline{v}_{i_0} &:= \max\{v_{i_0} | Q^S v \leq c^S + \overline{r}^S, v \in \mathbb{R}^S\}. \qquad (U^{i_0}_S) \end{aligned}$$

Then the optimal values of (P^{Σ}) *and* (P^{i_0}) *can be bounded as follows:*

$$\begin{aligned} \underline{v} \leq \mathbb{1}^t v_{i_0} &\leq \overline{v} \\ \underline{v}_{i_0} \leq \quad v_{i_0} &\leq \overline{v}_{i_0} \end{aligned}$$

In the sequel, we refer to the linear programs in Lemma 1 as the *lower/upper-bound LP* and the *single-state lower/upper-bound LP*, respectively. We will make use of them in the following form:

Corollary 1 *The optimal value vector v^α is bounded componentwise by optimal solutions $\underline{v}$ and $\overline{v}$ of the bounding linear programs in Lemma 1.*

Of course, finding a suitable S for these bounds to be useful requires substantial work in both the lower and the upper bound LPs. Thus, in the following we aim at utilizing the lower bound LP for both lower and upper bounds.

By adding an absorbing state to S and adjusting its stage costs accordingly, we can construct a *lower-bound induced MDP* $\underline{M}(S)$ and an *upper-bound induced MDP* $\overline{M}(S)$ with optimal cost-to-go $\underline{v}$ and $\overline{v}$, respectively. Moreover, by extending any policy for one of the induced MDPs arbitrarily outside S, we obtain a policy π for (M,α). In particular, we can produce two special upper bounds: extend an optimal policy $\underline{\pi}$ for $\underline{M}(S)$ or extend an optimal policy $\overline{\pi}$ for $\overline{M}(S)$ to (M,α).

Since the policy $\overline{\pi}$ might produce lower cost outside S than estimated by $\overline{r}^S$, its cost is bounded from above by the value of the upper-bound LP.

On the other hand, the policy $\underline{\pi}$ is just some policy in $\overline{M}(S)$. Thus, its cost in $\overline{M}(S)$ is bounded from below by the value of the upper-bound LP. Its value lies in the fact that this way we can derive a lower and an upper bound from the same lower-bound LP.

Summarized, we obtain:

Lemma 2 *Given a discounted MDP (M,α), a state $i_0 \in \mathbb{S}$, a subset of states $S \subseteq \mathbb{S}$ with $i_0 \in S$, optimal policies $\overline{\pi}$ for $\overline{M}(S)$ and $\underline{\pi}$ for $\underline{M}(S)$, let $\overline{v}^{\overline{\pi}}$ be the cost of $\overline{\pi}$ in $\overline{M}(S)$ and $\overline{v}^{\underline{\pi}}$ be the cost of $\underline{\pi}$ in $\overline{M}(S)$. Moreover, let $\overline{v}_{i_0}$ be the optimal value of the single-state upper-bound LP. Then,*

$$v^\alpha_{i_0} \le \overline{v}^{\overline{\pi}}_{i_0} \le \overline{v}_{i_0} \le \overline{v}^{\underline{\pi}}_{i_0}.$$

Moreover, the value $\overline{v}^{\underline{\pi}}_{i_0}$ equals the optimal value of the linear program

$$\overline{v}^{\underline{\pi}}_{i_0} = \max\left\{ v_{i_0} | Q^{S,\underline{\pi}} v \le c^{S,\underline{\pi}} + \overline{r}^{S,\underline{\pi}}, v \in \mathbb{R}^S \right\},$$

whose optimal solution equals the unique solution of the linear system

$$Q^{S,\underline{\pi}} v = c^{S,\underline{\pi}} + \overline{r}^{S,\underline{\pi}}. \tag{1}$$

Here, $Q^{S,\underline{\pi}}$, $c^{S,\underline{\pi}}$, and $\overline{r}^{S,\underline{\pi}}$ are the submatrices of Q^S, c^S, and r^S corresponding to state-action pairs induced by $\underline{\pi}$.

Thus, by computing an optimal solution of the lower-bound LP (small dimension) and extracting an optimal policy $\underline{\pi}$ of $\underline{M}(S)$ we obtain by solving the system of linear equations describing the cost of $\underline{\pi}$ in $\overline{M}(S)$ (small dimension again) lower *and* upper bounds on $v^\alpha_{i_0}$ (large dimension) at the same time. In contrast to this, the (possibly tighter) bound $\overline{v}^{\overline{\pi}}_{i_0}$ requires to compute the cost of a policy in the original MDP (large dimension), and the other (possibly tigher) bound $\overline{v}_{i_0}$ requires the solution of another LP (the upper-bound LP).

So far, our goal was to approximate the optimal cost-to-go by smaller induced MDPs. By adding suitable further restrictions to the lower/upper bound LPs we can also assess given policies and actions. The value vector of an MDP M w. r. t. a given discount factor α will be denoted by v_M^α in the following.

We now address the local approximation of the value vector $v_M^\alpha(\pi)$ of a given policy π. The basic idea is to restrict the possible state-action pairs to the actions of the given policy. We call this restricted MDP *π-induced MDP*.

Lemma 3 *Given an MDP $M = (\mathbb{S}, \mathbb{A}, p, c)$ and a policy π, define the* policy induced *MDP $M(\pi) = (\mathbb{S}, \mathbb{A}', p', c')$ by $\mathbb{A}'(i) = \{\pi(i)\}$ for each state $i \in \mathbb{S}$ and suitable restrictions p' and c' of the transition probabilities and stage costs. Then, we have $v_M^\alpha(\pi) = v_{M(\pi)}^\alpha$ for any discount factor $\alpha \in [0,1)$.*

Thus, all the above local approximation tools apply in particular for policy evaluation.

Similarly, we can restrict the set of possible actions in a given state i_0 to a single action a_0. The corresponding MDP is denoted by $M(i_0, a_0)$. We define the cost-to-go $v_{M,i_0}^\alpha(a_0)$ of action a_0 in a state i_0 as follows:

$$v_{M,i_0}^\alpha(a_0) = v_{M(i_0,a_0),i_0}^\alpha.$$

With this we can characterize the optimality of given actions in given states. Since we are usually only computing bounds on v_M^α, the tool can mainly be used to certify that a given action is *not* optimal.

Lemma 4 *Given an MDP $M = (\mathbb{S}, \mathbb{A}, p, c)$ and a state $i_0 \in \mathbb{S}$, an action $a_0 \in \mathbb{A}(i_0)$ is optimal for a discount factor α if and only if $v_{M,i_0}^\alpha(a_0) = v_{M,i_0}^\alpha$.*

In the following we present a structural approximation theorem justifying the approach theoretically. It shows that – even without sophisticated state-dependent bounds – an ε-approximation of one component of the optimal value vector can be obtained by taking into account only a small local part of the entire state space, not depending on the total number of states. The bound can be seen as a yard stick for all application specific efforts: a tailor-made method is only useful if it beats the bound of the following theorem by a significant margin.

Definition 2 **(r-neighborhood)** For an MDP $(\mathbb{S}, \mathbb{A}, p, c)$, a particular state $i_0 \in \mathbb{S}$, and a number $r \in \mathbb{N}$, the *r-neighborhood $S(i_0, r)$ of i_0* is the subset of states that can be reached from i_0 within at most r transitions. That is, $S(i_0, 0) := \{i_0\}$ and for $r > 0$ we define:

$$S(i_0, r) := S(i_0, r-1) \cup \left\{ j \in \mathbb{S} \mid \exists i \in S(i_0, r-1)\, \exists a \in \mathbb{A}(i) : p_{ij}(a) > 0 \right\}.$$

It is straight-forward that the size of the r-neighborhood can not grow too fast under certain conditions.

Lemma 5 *Let $M = (\mathbb{S}, \mathbb{A}, p, c)$ be an MDP, $\alpha \in [0,1)$ a discount factor, and $D \in \mathbb{N}$ such that for each $i \in \mathbb{S}$ the number of states $j \in \mathbb{S}$ with positive transition probabilities $p_{ij}(a)$ for some $a \in \mathbb{A}(i)$ is bounded by D, then $|S| \leq \max\left\{D^{r+1}, r+1\right\}$.*

Such a D exists, e.g., when in each state there are at most b feasible actions and at most d possible successor states for that action. Then, D can be set to bd.

Proof Let $i_0 \in \mathbb{S}$. Since the number of possible successors is bounded by D we have:

$$|S| \leq \sum_{k=0}^{r} D^k = \frac{D^{r+1}-1}{D-1} \leq D^{r+1},$$

if $D \geq 2$. In the trivial case $D = 1$ we obviously have $|S| = r+1$. □

Note that the stage costs accounted in the total expected discounted cost decrease geometrically. Thus, for a given approximation guarantee ε it is clear that the r-neighborhood $S(i_0, r)$ of i_0 for some radius $r = r(\varepsilon) \in \mathbb{N}$ will provide an ε-approximation for $v_{i_0}^{\alpha}$ via the associated linear programs. The value of the following theorem lies in the explicit *formula* for the radius r required for a given approximation guarantee (we already documented a weaker version of this result in the preprint [20]).

Theorem 2 *Let $M = (\mathbb{S}, \mathbb{A}, p, c)$ be an MDP, $\alpha \in [0,1)$ a discount factor, and $D \in \mathbb{N}$ such that for each $i \in \mathbb{S}$ the number of states $j \in \mathbb{S}$ with positive transition probabilities $p_{ij}(a)$ for some $a \in \mathbb{A}(i)$ is bounded by D. Let $c_{\max} := \max_{i \in \mathbb{S}, a \in \mathbb{A}(i)} c_i(a)$ and $v_{\max}^{\alpha} := c_{\max}/(1-\alpha)$. Then, for each state $i_0 \in \mathbb{S}$ and for each $\varepsilon > 0$, the subset of states $S = S(i_0, r) \subseteq \mathbb{S}$ with*

$$r = \max\left\{0, \left\lceil \log\left(\frac{\varepsilon}{v_{\max}^{\alpha}}\right) / \log \alpha \right\rceil - 1\right\}$$

satisfies the following properties:

(i) $|S| \leq \max\left\{D^{r+1}, r+1\right\}$*, in particular, the number of states in S does not depend on* $|\mathbb{S}|$.

(ii) For state i_0, any optimal solution $\underline{v}$ of the lower-bound LP and the unique solution $\overline{v}^{\underline{\pi}}$ of system (1) *w. r. t. any optimal policy $\underline{\pi}$ for the lower-bound induced MDP $\underline{M}(S)$ satisfy:*

$$\overline{v}_{i_0}^{\underline{\pi}} - \underline{v}_{i_0} \leq \varepsilon.$$

In particular, $\underline{v}_{i_0}$ and $\overline{v}_{i_0}^{\underline{\pi}}$ themselves are ε-close lower and upper bounds on the optimal value vector v^{α} at state i_0, i. e.,

$$0 \leq v_{i_0}^{\alpha} - \underline{v}_{i_0} \leq \varepsilon,$$
$$0 \leq \overline{v}_{i_0}^{\underline{\pi}} - v_{i_0}^{\alpha} \leq \varepsilon.$$

Proof Part (i) follows from Lemma 5.

The proof of Property (ii) is as follows. Let $\varepsilon > 0$. Consider the extension $\underline{v}^{\text{ext}} \in \mathbb{R}^{\mathbb{S}}$ of the solution $\underline{v}$ of the lower-bound LP:

$$\underline{v}_i^{\text{ext}} = \begin{cases} \underline{v}_i, & \text{if } i \in S, \\ 0, & \text{if } i \in \mathbb{S} \setminus S. \end{cases}$$

Moreover, let $\underline{\pi}$ be an optimal policy for $\underline{M}(()S)$ and construct an extension $\overline{v}^{\text{ext}} \in \mathbb{R}^{\mathbb{S}}$ of the solution $\overline{v}^{\underline{\pi}}$ of system (1) w. r. t. policy $\underline{\pi}$ as follows:

$$\overline{v}_i^{\text{ext}} = \begin{cases} \overline{v}_i^{\underline{\pi}}, & \text{if } i \in S, \\ v_{\max}^{\alpha}, & \text{if } i \in \mathbb{S} \setminus S. \end{cases}$$

The optimal solution $\underline{v}$ of the lower-bound LP equals the optimal value vector of the MDP $\underline{M}(S)$. Since $\underline{\pi}$ is optimal for $\underline{M}(S)$, the corresponding constraints in the lower-bound LP are satisfied with equality by $\underline{v}$, i. e.,

$$\underline{v}_i = c_i(\underline{\pi}(i)) + \alpha \sum_{j \in S} p_{ij}\big(\underline{\pi}(i)\big)\underline{v}_j \quad \forall i \in S,$$

which implies for the extension $\underline{v}^{\text{ext}}$:

$$\underline{v}_i^{\text{ext}} = c_i(\underline{\pi}(i)) + \alpha \sum_{j \in \mathbb{S}} p_{ij}(\underline{\pi}(i))\underline{v}_j^{\text{ext}} \quad \forall i \in S. \tag{2}$$

On the other hand, since $\overline{v}^{\underline{\pi}}$ satisfies the system of equations (1) we have the following relation for the extension $\overline{v}^{\text{ext}}$:

$$\overline{v}_i^{\text{ext}} = c_i(\underline{\pi}(i)) + \alpha \sum_{j \in \mathbb{S}} p_{ij}(\underline{\pi}(i))\overline{v}_j^{\text{ext}} \quad \forall i \in S. \tag{3}$$

From the Equations (2) and (3) we obtain:

$$\overline{v}_i^{\text{ext}} - \underline{v}_i^{\text{ext}} = \alpha \sum_{j \in \mathbb{S}} p_{ij}(\underline{\pi}(i))(\overline{v}_j^{\text{ext}} - \underline{v}_j^{\text{ext}}) \quad \forall i \in S. \tag{4}$$

In the following, we show by reverse induction on $k = r, \ldots, 0$ for each state $i \in S(i_0, k)$:

$$\overline{v}_i^{\text{ext}} - \underline{v}_i^{\text{ext}} \leq \alpha^{r+1-k} v_{\max}^{\alpha}. \tag{5}$$

Note that all i to which (5) refers are contained in S because of $k \leq r$. For $k = r$ and for each state $i \in S(i_0, k)$, Inequality (5) follows from (4) due to $\overline{v}_j^{\text{ext}} \leq v_{\max}^{\alpha}$ and $\underline{v}_j^{\text{ext}} \geq 0$ for each $j \in \mathbb{S}$:

$$\begin{aligned}\overline{v}_i^{\text{ext}} - \underline{v}_i^{\text{ext}} &\leq \alpha \sum_{j \in \mathbb{S}} p_{ij}(\underline{\pi}(i))\big(v_{\max}^{\alpha} - 0\big) \\ &= \alpha v_{\max}^{\alpha}.\end{aligned}$$

Here, the equality follows from the fact that $\sum_{j \in \mathbb{S}} p_{ij}(\underline{\pi}(i)) = 1$ for each state $i \in \mathbb{S}$.

Now assume that Inequality (5) holds for each state $j \in S(i_0, k)$ with $0 < k \leq r$. For each $i \in S(i_0, k-1)$, we again apply Equality (4):

$$\begin{aligned}\overline{v}_i^{\text{ext}} - \underline{v}_i^{\text{ext}} &= \alpha \sum_{j \in \mathbb{S}} p_{ij}(\underline{\pi}(i))(\overline{v}_j^{\text{ext}} - \underline{v}_j^{\text{ext}}) \\ &= \alpha \sum_{j \in S(i_0,k)} p_{ij}(\underline{\pi}(i))(\overline{v}_j^{\text{ext}} - \underline{v}_j^{\text{ext}}),\end{aligned}$$

where the second identity is due to the fact that each state $j \in \mathbb{S}$ with $p_{ij}(\underline{\pi}(i)) > 0$ is contained in $S(i_0, k)$ since $i \in S(i_0, k-1)$. We can apply the induction hypothesis for each state $j \in S(i_0, k)$:

$$\begin{aligned}\overline{v}_i^{\text{ext}} - \underline{v}_i^{\text{ext}} &\leq \alpha \sum_{j \in S(i_0,k)} p_{ij}(\underline{\pi}(i))\alpha^{r+1-k} v_{\max}^{\alpha} \\ &= \alpha^{r+1-(k-1)} v_{\max}^{\alpha},\end{aligned}$$

which completes the inductive proof of (5).

For $i = i_0$ and $k = 0$, Inequality (5) implies:

$$\overline{v}_{i_0}^{\pi} - \underline{v}_{i_0} = \overline{v}_{i_0}^{\text{ext}} - \underline{v}_{i_0}^{\text{ext}} \leq \alpha^{r+1} v_{\max}^{\alpha}.$$

Finally, we distinguish two cases to show Property (ii). If $\varepsilon \geq \alpha v_{\max}^{\alpha}$, we have $r = 0$, and thus $\overline{v}_{i_0}^{\pi} - \underline{v}_{i_0} \leq \alpha v_{\max}^{\alpha} \leq \varepsilon$. Otherwise, if $\varepsilon < \alpha v_{\max}^{\alpha}$, it follows that $\log(\varepsilon / v_{\max}^{\alpha}) < \log \alpha < 0$ and $r = \lceil \log(\varepsilon / v_{\max}^{\alpha}) / \log \alpha \rceil - 1$ which implies:

$$\begin{aligned} \overline{v}_{i_0}^{\pi} - \underline{v}_{i_0} &\leq \alpha^{\lceil \log(\varepsilon / v_{\max}^{\alpha}) / \log \alpha \rceil} v_{\max}^{\alpha} \\ &\leq \alpha^{\log(\varepsilon / v_{\max}^{\alpha}) / \log \alpha} v_{\max}^{\alpha} \\ &= \varepsilon. \end{aligned}$$

It remains to be proven that $\underline{v}_{i_0}$ and $\overline{v}_{i_0}^{\pi}$ are ε-close lower and upper bounds for the component $v_{i_0}^{\alpha}$. From Lemmas 1 and 2 it is already known that $\underline{v}_{i_0} \leq v_{i_0}^{\alpha} \leq \overline{v}_{i_0}^{\pi}$. By these inequalities we obtain:

$$\begin{aligned} \overline{v}_{i_0}^{\pi} - v_{i_0}^{\alpha} &\leq \overline{v}_{i_0}^{\pi} - \underline{v}_{i_0} \leq \varepsilon, \\ v_{i_0}^{\alpha} - \underline{v}_{i_0} &\leq \overline{v}_{i_0}^{\pi} - \underline{v}_{i_0} \leq \varepsilon. \end{aligned}$$

□

We mention that Theorem 2 and its proof are still true (with minor modifications) in the case of an infinite state space $\mathbb{S}$ if there exists a finite upper bound for the expected stage costs, i. e., $\sup_{i \in \mathbb{S}, a \in \mathbb{A}(i)} c_i(a) < \infty$. This is because, under this assumption, the cost outside $S(i_0, r)$ can still be bounded from above by the infinite geometric series of the α-discounted supremum of stage costs.

Since the optimal value of the upper-bound LP is at least as tight as $\overline{v}_{i_0}^{\pi}$ (see Lemma 2), we also have the following result.

Corollary 2 *Under the same assumptions as used in Lemma 1, let $\overline{v}_{i_0}$ be the optimal value of the upper-bound LP for the subset of states $S = S(i_0, r)$. Then, we have:*

$$\overline{v}_{i_0} - \underline{v}_{i_0} \leq \varepsilon.$$

Particularly, $\overline{v}_{i_0}$ is also an ε-close upper bound on $v_{i_0}^{\alpha}$, i. e., $\overline{v}_{i_0} - v_{i_0}^{\alpha} \leq \varepsilon$.

Remark 2 The size of the restricted state space is optimal in some sense, as can be seen from the example of a "tree like" MDP, in which every state has exactly D possible successor states that can only be reached via this state. In this case, one can show that $S = S(i_0, r)$ as above is the smallest restricted state space to obtain the desired approximation. Of course, incorporating additional parameters of the MDP might give better results in special cases.

Remark 3 Of all the approaches from the literature the random sampling algorithm of Kearns et al. [24] gives the results most comparable to Theorem 2. However, the size of the restricted state space in our construction is significantly smaller than that

for random sampling. This algorithm samples states within the neighborhood of the considered state i_0 up to a radius r_s with:

$$r_s = \left\lceil \frac{\log x}{\log \alpha} \right\rceil, \quad \text{where } x := \frac{\varepsilon(1-\alpha)^3}{4c_{\max}}.$$

Obviously, this gives a considerably larger subset of states since r_s is greater than the radius $r = \lceil \log(\varepsilon(1-\alpha)/c_{\max})/\log\alpha \rceil - 1$ used in Theorem 2. For instance, if $c_{\max} = 1$, $\alpha = 0.7$, and $\varepsilon = 0.1$, the radius r_s equals $r_s = 21$, while the radius in our construction equals $r = 10$.

However, the setting considered in [24] is quite different as the authors assume the maximum number of successor states d for an action to be very large or even infinite. Indeed, the number of states sampled by their algorithm is independent of d. This way, their approach deals with the third curse of dimensionality also, i. e., a huge number of possible successors. They sample for each considered state in radius smaller than r_s, at most

$$T = x^{-2}\left[\ln\left(\frac{1-\alpha}{x}\right) + 2r\ln\left(x^{-2}br\ln\left(\frac{1-\alpha}{x}\right)\right)\right]$$

consecutive states if $T < d$. Note that this restriction only makes a difference when d is really large: even fairly simple situations imply huge values for T, e. g., if $c_{\max} = 1$, $b = 4$, $\alpha = 0.7$, and $\varepsilon = 0.1$, we obtain for T a value greater than 1.9 billion.

Our proposal is not to use the state space restricted by the bound on the necessary radius but a state space dynamically computed by column generation techniques. This will be the topic of the next section.

3.2 Column Generation

In order to compute local approximations of the optimal value vector $v^\alpha_{i_0}$ around a particular state of a given MDP, it is usually inappropriate to apply the construction of Theorem 2 directly.

The general idea of our approximation algorithm is to start with a small subset of states $S_1 \subset \mathbb{S}$ containing the considered state $i_0 \in \mathbb{S}$. The state space S_1 provides initial lower and upper bounds on $v^\alpha_{i_0}$ via the solution of the corresponding linear programs $(\mathrm{L}^{i_0}_{S_1})$ and $(\mathrm{U}^{i_0}_{S_1})$. Then, in order to improve the approximation on $v^\alpha_{i_0}$, the state space S_1 is successively extended by adding new states. Note that each newly added state $i \in \mathbb{S} \setminus S_1$ results in one additional variable and $|\mathbb{A}(i)|$ additional constraints in both linear programs $(\mathrm{L}^{i_0}_{S_1\cup\{i\}})$ and $(\mathrm{U}^{i_0}_{S_1\cup\{i\}})$. This way, the algorithm constructs a finite sequence of subsets $S_1 \subset S_2 \subset \cdots \subset S_n \subseteq \mathbb{S}$ for some $n \in \mathbb{N}$ together with a sequence of improving lower and upper bounds on $v^\alpha_{i_0}$ obtained as the optimal values of the corresponding linear programs. Using policy iteration instead of linear programming a similar algorithmic approach has already been proposed by Dean et al. [7]. However, our approach has several advantages as we will see later.

Recall that the theoretical approximation results given in Theorem 2 and Corollary 2 provide an approximation in terms of the absolute difference between upper

Algorithm 1 Generic approximation algorithm

1: **Input:** an MDP $(\mathbb{S}, \mathbb{A}, p, c)$ (given implicitly), a discount factor $\alpha \in [0,1)$, a state $i_0 \in \mathbb{S}$, a subset of states $S \subseteq \mathbb{S}$ with $i_0 \in S$, $\varepsilon > 0$
2: **Output:** lower and upper bounds $\underline{v}_{i_0}, \overline{v}_{i_0}$ on $v_{i_0}^{\alpha}$ with $(\overline{v}_{i_0} - \underline{v}_{i_0})/\underline{v}_{i_0} \leq \varepsilon$
3: compute $\underline{v}_{i_0}$ and $\overline{v}_{i_0}$ as the optimal values of the lower/upper-bound LPs
4: **if** $(\overline{v}_{i_0} - \underline{v}_{i_0})/\underline{v}_{i_0} \leq \varepsilon$ **then**
5: **return** $\underline{v}_{i_0}, \overline{v}_{i_0}$
6: **else**
7: $S \leftarrow S \cup S_{\text{new}}$ for some $S_{\text{new}} \subseteq \mathbb{S} \setminus S$
8: go to step 3
9: **end if**

and lower bounds. In practice, however, a relative guarantee is typically more suitable when $\underline{v}_{i_0} > 0$. Therefore, the usual goal of our algorithm is to obtain an approximation on $v_{i_0}^{\alpha}$, where the relative difference between the upper and lower bounds is less than a desired guarantee $\varepsilon > 0$, i. e.,

$$\frac{\overline{v}_{i_0} - \underline{v}_{i_0}}{\underline{v}_{i_0}} \leq \varepsilon \text{ for } \underline{v}_{i_0} > 0.$$

Once this approximation guarantee is obtained, the algorithm terminates. In the following, we tacitly assume that $\underline{v}_{i_0} > 0$ whenever the relative performance guarantee is referred to. The generic approximation algorithm is summarized in Algorithm 1. Clearly, Algorithm 1 terminates after a finite number of iterations whenever the state space $\mathbb{S}$ is finite.

Remark 4 It has been shown in Lemma 2 that by solving the upper-bound LP for some state space $S \subseteq \mathbb{S}$ with $i_0 \in S$, one can easily derive a policy π for the original MDP with the property $v_{i_0}^{\alpha}(\pi) \leq \overline{v}_{i_0}$. Consequently, our approximation algorithm also determines a *near-optimal* action a_0 at state i_0 in the sense that there exists a policy π with $\pi(i_0) = a_0$ such that $(v_{i_0}^{\alpha}(\pi) - v_{i_0}^{\alpha})/v_{i_0}^{\alpha} \leq \varepsilon$.

Our implementation of Algorithm 1 is based on the idea to extend the considered state space dynamically by means of column generation, which is a standard technique for solving large-scale linear programs. We refer to the book of Desaulniers et al. [10] for details about column generation. The original problem we aim to solve (approximately) here is $(\mathrm{L}_{\mathbb{S}}^{i_0})$, which equals the linear program (P^{i_0}). Consequently, the master problem that is to be solved in each iteration of the column generation is $(\mathrm{L}_{S}^{i_0})$ for some subset of states $S \subseteq \mathbb{S}$ with $i_0 \in S$. Thus, for computing the sequence of state spaces $S_1 \subset S_2 \subset \cdots \subset S_n \subseteq \mathbb{S}$ we solely consider the linear programs providing the lower bounds on $v_{i_0}^{\alpha}$. The upper-bound LP only contribute in terms of the computed upper bounds. We mention that it is not straight-forward to solve the pricing problem in a column generation algorithm w. r. t. the upper-bound LP with $S \subset \mathbb{S}$, since an associated feasible solution cannot be extended trivially to one for $(\mathrm{L}_{\mathbb{S}}^{i_0})$.

In order to keep the pricing problem of our column generation tractable, we employ *incomplete pricing*: as long as we find state-action pairs with positive reduced profits that can be reached from the current set of state-action pairs by one transition (distance one), we do not consider other state-action pairs. Only if we do not find positive

reduced profits at distance one, we extend the search to distance two, three, etc. In various tests with various pricing strategies it turned out that choosing state-action pairs with maximal reduced profits (inside the restricted search space) worked the best. (For a detailed documentation of these tests see [31].)

Finally, we briefly discuss our method compared to the approach of Dean et al. [7]. The aim of their method is to find an optimal policy for a state space restricted to those states which are likely to be encountered within a smaller number of transitions. Similar to our approach, their algorithm computes an optimal policy for the induced MDP in each iteration and extends the restricted state space dynamically depending on the obtained policy. Instead of linear programming, policy iteration is used to compute the optimal policies. The main advantages of Algorithm 1 compared to their method are the following. Firstly, in the approximation process we are able to monitor the current approximation guarantee, while the approach of Dean et al. only provides lower bounds on $v^{\alpha}_{i_0}$. Thus, they cannot determine how good the current approximation really is. Secondly, we are able to properly guide the expansion of the restricted state space as the reduced profits of the candidate states are available. This way, our approximation algorithm benefits substantially (see [31] for computational results). The method of Dean et al. must use heuristic ideas to increase S, in particular, one strategy aims to estimate the reduced profits. Probably, both algorithms have a similar run-time per iteration since the policy iteration method and linear programming method for computing the optimal value vector are in some sense equivalent. Our algorithm may be a bit slower per iteration when a second linear program is solved.

3.3 State-Dependent Bounds

We exploit involved lower and upper bounds on the components v^{α}_{i} of the optimal value vector. Recall that we consider two different elevator control MDPs, one for analyzing the average waiting time and another for dealing with the maximum waiting time.

3.3.1 Average waiting time

The construction of state-specific bounds for the MDP modeling the average waiting time is as follows. For each state $i \in \mathbb{S}$, we employ a lower bound $v^{\alpha}_{\min}(i) \leq v^{\alpha}_{i}$ consisting of two parts, i. e., $v^{\alpha}_{\min}(i) = v^{\alpha,1}_{\min}(i) + v^{\alpha,2}_{\min}(i)$.

The first lower bound $v^{\alpha,1}_{\min}(i)$ takes into account future requests arriving in the system. It is based on a lower bound for the probability $p^{\text{no\_elevator}}$ that a request arrives at a floor, where no elevator is located. Let again $0 \leq p_f \leq 1$ be the probability that a request with start floor $f \in F$ is released at a time slot. Consider a permutation $f_1, \ldots, f_{|F|} \in F$ of the floors such that the probabilities are non-decreasing w. r. t. the permutation: $p_{f_1} \leq \cdots \leq p_{f_{|F|}}$. Since in each state there exist at least $|F| - |E|$ floors where no elevator is located, the probability $p^{\text{no\_elevator}}$ is at least the sum of the $|F| - |E|$ smallest

arrival probabilities $p_{f_1}, \ldots, p_{f_{|F|-|E|}}$, i. e., we have:

$$p^{\text{no\_elevator}} \geq \sum_{k=1}^{|F|-|E|} p_{f_k}.$$

Since each request arriving at a floor where no elevator is located will have a waiting time greater or equal 1 and such a request can arrive at each time slot, we obtain:

$$v_i^\alpha \geq \frac{p^{\text{no\_elevator}}}{1-\alpha} \geq \frac{\sum_{k=1}^{|F|-|E|} p_{f_k}}{1-\alpha} =: v_{\min}^{\alpha,1}(i).$$

Note that the first inequality above is only valid since the penalty cost satisfies by assumption $c_p \geq 1 \geq p^{\text{no\_elevator}}$. This gives the first part of the lower bound.

The second part $v_{\min}^{\alpha,2}(i)$ of the lower bound on v_i^α for a state $i \in \mathbb{S}$ captures the total α-discounted cost resulting from the requests waiting in state i.

In the following, we restrict ourselves to the case of one elevator for the ease of exposition. (A similar bound can be obtained for more elevators.) We consider a relaxation of the elevator control problem where the elevator requires no time for moving empty, and all requests waiting at the same floor can be served in arbitrary order. Note that the resulting problem is equivalent to a scheduling problem where the machine corresponds to the elevator and the jobs correspond to the waiting requests. In the following, the current time slot at state i will be denoted by 0 and the consecutive time slots by $1, 2, \ldots$.

Theorem 3 *Let $i = (w_{\max}, (\sigma_f)_{f\in F}, (f_e, d_e)_{e\in E})$ be a state in a Markov decision process with a single elevator e and floor set F. Let $r_1, r_2, \ldots, r_n$ be the waiting requests sorted in non-decreasing order of transportation times, and let $\Delta_1 \leq \Delta_2 \leq \cdots \leq \Delta_n$ be the transportation times plus loading and dropping times. Moreover, let Δ_0 be the earliest boarding time of any request given the current state of the elevator and the origins of the requests. Then for each $0 < \alpha \leq 1$, the α-discounted cost ist at least*

$$v_i^\alpha \geq \sum_{i=1}^{n} (n-i) \sum_{k=\Delta_0+\Delta_1+\cdots+\Delta_{i-1}}^{\Delta_i-1} \alpha^k =: v_{\min}^{\alpha,2}(i). \tag{6}$$

Proof The bound $v_{\min}^{\alpha,2}(i)$ arises as the α-discounted sum of all already released waiting requests at a time slot over all time slots. It therefore equals the α-discounted sum of waiting times that is achieved when

- no further requests arrive;
- the elevator can move to the first request in time Δ_0 and from the destination of a request to the origin of the next request in time zero;
- the requests are served in the order of non-decreasing transportation times.

Thus, $v_{\min}^{\alpha,2}(i)$ is the α-discounted cost of a schedule that is feasible for a relaxed problem with only the known requests, minimal time for the starting move and zero time for empty moves and no precedence constraints for the requests waiting on the same floor. Assume, an optimal algorithm OPT to the relaxed problem does not schedule the requests in the order of non-decreasing transportation times. Assume,

moreover, that OPT incurs a strictly smaller cost than $v_{\min}^{\alpha,2}(i)$. Then there are two consecutive requests r_i and r_{i+1} with $\Delta_i < \Delta_{i+1}$ that are scheduled in the order (r_{i+1}, r_i) in OPT. Let w be the waiting time of r_{i+1} in OPT. Consider the modified schedule S that arises from OPT by switching r_i and r_{i+1}. Then at all time slots the numbers of waiting requests for OPT and S stay the same except for the non-empty set of time slots

$$\{w+\Delta_i, \ldots, w+\Delta_{i+1}-1\},$$

in which S produces one waiting request less. This is because r_{i+1} is not waiting anymore in S (r_i is completed and there is no transition time) but r_i is still waiting in OPT (r_{i+1} is not yet completed). Thus, the α-discounted sum of waiting times in S is strictly less than that of OPT: contradiction. Hence, $v_{\min}^{\alpha,2}(i)$ is the optimal value of a relaxed problem and, therefore, a lower bound to v_i^α. □

Remark 5 The exchange in the proof of Theorem 3 is completely analogous to the one in the usual optimality proof for the "shortest processing time first" rule in single-machine scheduling. Only the bookkeeping changed a little in order to account for the discounting.

Notice that $v_{\min}^{\alpha,2}(i)$ takes into account the costs incurred from currently waiting requests only, while $v_{\min}^{\alpha,1}(i)$ solely considers costs due to future requests. Therefore, their sum $v_{\min}^{\alpha}(i) := v_{\min}^{\alpha,1}(i) + v_{\min}^{\alpha,2}(i)$ is a valid lower bound for the component v_i^α of the optimal value vector, too.

Obviously, the trivial upper bound $v_{\max}^{\alpha} = c_{\max}/(1-\alpha)$ is very weak in the considered elevator control MDP for most states since the maximum expected stage cost equals $c_{\max} = |F|q + c_p \sum_{f\in F} p_f$. The approach to determine a suitable upper bound $v_{\max}^{\alpha}(i) \geq v_i^\alpha$ for each state $i \in \mathbb{S}$ is to compute the expected total number of waiting requests and the expected penalty for each future time slot t up to some limit assuming that no requests are served.

Let $N_t^{\text{wait}} \in \mathbb{N}_0$ and $N_{t,f}^{\text{wait}}$ denote the random variables for the total number of waiting requests and the number of requests waiting at floor $f \in F$ for time slot $t \in \mathbb{N}_0$, respectively. By the linearity of the expectation we have:

$$\mathbb{E}[N_t^{\text{wait}}] = \mathbb{E}[\sum_{f\in F} N_{t,f}^{\text{wait}}] = \sum_{f\in F} \mathbb{E}[N_{t,f}^{\text{wait}}].$$

For each $f \in F$, the expected value $\mathbb{E}[N_{t,f}^{\text{wait}}]$ can be computed according to the arrival probability p_f at floor f by:

$$\mathbb{E}[N_{t+1,f}^{\text{wait}}] = \min\{\mathbb{E}[N_{t,f}^{\text{wait}}] + p_f, q\}.$$

Moreover, let $P_t \geq 0$ denote the random penalty cost for a stage $t \in \mathbb{N}_0$. In order to determine the expected penalty $\mathbb{E}[P_t]$, we compute the probability $p_{t,f}^{\text{full}}$ that the waiting queue at a floor $f \in F$ is full at time slot t. Let $c_f := q - |\sigma_f(i)|$ be the remaining capacity at each floor $f \in F$ in state i. Note that we always have $p_{t,f}^{\text{full}} = 0$ as long as $t < c_f$ since at most one request is released each stage. Generally, $p_{t,f}^{\text{full}}$ equals the

probability that at least c_f new requests have arrived at floor f by time t. Therefore, we obtain for each $t \in \mathbb{N}_0$:

$$p_{t,f}^{\text{full}} = \sum_{k=c_f}^{t} \binom{t}{k} p_f^k (1-p_f)^{t-k}.$$

Again by the linearity of the expectation, the expected penalty $\mathbb{E}[P_t]$ at time $t \in \mathbb{N}_0$ equals:

$$\mathbb{E}[P_t] = c_p \sum_{f \in F} p_{t,f}^{\text{full}} \cdot p_f.$$

Given the expected number of waiting requests $\mathbb{E}[N_t^{\text{wait}}]$ and the expected penalty cost $\mathbb{E}[P_t]$ under the assumption that no requests are served, for each time slot $t \in \mathbb{N}_0$, we have:

$$v_i^\alpha \leq \sum_{t=0}^{\infty} \alpha^t (\mathbb{E}[N_t^{\text{wait}}] + \mathbb{E}[P_t]).$$

Clearly, it is impossible to determine an infinite number of expectations. We stop the expensive computation described above at a time slot $t_{\max}$ when the maximum total α-discounted expected cost $\alpha^{t_{\max}} v_{\max}^\alpha$ after time $t_{\max}$ falls below some threshold value (e. g., 0.1) and add $\alpha^{t_{\max}} v_{\max}^\alpha$. Thus, we obtain the upper bound:

$$v_i^\alpha \leq \alpha^{t_{\max}} v_{\max}^\alpha + \sum_{t=0}^{t_{\max}} \alpha^t (\mathbb{E}[N_t^{\text{wait}}] + \mathbb{E}[P_t]) =: v_{\max}^\alpha(i).$$

Notice that the construction above assumes that none of the requests currently waiting in a state are ever served. We mention that for approximating the component v_i^α of the optimal value vector for some $i \in \mathbb{S}$, it is possible to take into account the processing of the requests waiting in state i according to any feasible schedule. In doing so, the expected stage costs $\mathbb{E}[N_t^{\text{wait}}]$ and $\mathbb{E}[P_t]$ reduce for some time slots t due to serving requests in state i. Consequently, we obtain an improved upper bound $v_{\max}^\alpha(i)$. Our implementation applies the feasible schedule obtained by the policy NN. Note that this construction is generally infeasible when the goal is to approximate the value $v_{i_0}^\alpha(\pi)$ for a policy π since the schedule of π may change when additional requests arrive. Obviously, this is not the case for FIFO, i. e., we can employ the improved bound according to the schedule obtained by FIFO. For all other policies under consideration we have to assume that no requests are served.

3.3.2 Maximum waiting time

In the elevator control MDP for the maximum waiting time, a lower bound $v_{\min}^\alpha(i) \leq v_i^\alpha$ for a state $i \in \mathbb{S}$ is obtained as follows. Let $F_{\text{wait}}(i) \subseteq F$ be the subset of floors where at least one request is waiting in state i, i. e., $F_{\text{wait}}(i) = \{f \in F \mid \sigma_f(i) \neq \emptyset\}$. Moreover, for each floor $f \in F_{\text{wait}}(i)$, let r_f denote the first request in the waiting queue $\sigma_f(i)$ in state i.

The idea for constructing the lower bound on v_i^α is to determine for each floor $f \in F_{\text{wait}}(i)$ the smallest time t_f by which an elevator can reach floor f, after possibly having served a loaded request. That is, each request r_f for a floor $f \in F_{\text{wait}}(i)$ cannot

be loaded before time t_f. Consequently, the smallest possible final waiting time of request r_f equals $w_f^{\text{final}} := w_f(i) + t_f$, where $w_f(i)$ denotes the present waiting time of request r_f in state i. Note that the current maximum waiting time $w_{\max}(i)$ will increase if we have $\max_{f \in F_{\text{wait}}(i)} w_f^{\text{final}} > w_{\max}(i)$. By considering all floors $f \in F_{\text{wait}}(i)$ in order of decreasing current waiting times $w_f(i)$, one can determine a subset of time slots $T \subset \mathbb{N}_0$, where the maximum waiting time will increase, i. e., the associated stage cost equals 1.

Such a set T implies the following lower bound on v_i^α:

$$v_i^\alpha \geq \sum_{t \in T} \alpha^t =: v_{\min}^\alpha(i).$$

Clearly, an upper bound $v_{\max}^\alpha(i)$ for a state $i \in \mathbb{S}$ can be derived by arbitrarily serving the requests waiting in state i and assuming that another request is released at time slot 1 and never served. Consider any policy to compute a feasible schedule for all waiting requests. In our implementation we use FIFO. According to the schedule we obtain the subset of time slots $T \subset \mathbb{N}_0$, where the maximum waiting time will increase. For constructing the second part of the bound let $w_{\max}$ be the maximum final waiting time of a request in state i w. r. t. the used schedule. Note that the waiting time of a request released at time slot 1 will be bounded by $w_{\max}$ until time $w_{\max} + 1$. Therefore, never serving this request implies a stage cost of 1 for the time slots $w_{\max} + 1, w_{\max} + 2, \ldots$. Putting the two parts of the construction together, we obtain the following upper bound on v_i^α:

$$v_{\max}^\alpha(i) := \sum_{t \in T} \alpha^t + \sum_{t=w_{\max}+1}^{\infty} \alpha^t = \sum_{t \in T} \alpha^t + \frac{\alpha^{w_{\max}+1}}{1-\alpha} \geq v_i^\alpha.$$

Similar as before, this upper bound is in general not valid if a component $v_{i_0}^\alpha(\pi)$ of the value vector of a given policy π is to be approximated, although the bound is again valid for FIFO. For other policies, we use a simple upper bound for $v_i^\alpha(\pi)$ with $i \in \mathbb{S}$ obtained by computing the maximum current waiting time $w(i)$ of a request in state i. It is clear that the stage cost will equal 0 for the time slots $0, \ldots, w_{\max}(i) - w(i) - 1$. This implies the upper bound:

$$v_{\max}^\alpha(i) := \frac{\alpha^{w_{\max}(i)-w(i)}}{1-\alpha} \geq v_i^\alpha(\pi).$$

4 Computational Results

In this section, we present some selected results which show how our analysis tool can be used to assess and improve policies for the elevator control problem. For a more detailed, extensive computational study we refer to [31].

Table 1 Considered instances of the elevator control Markov decision processes. The considered start-to-destination floor probability distributions, denoted by `ud` and `sp` are given in Table 2 and Table 3, respectively.

instance	n_F	n_E	q	c_p	p^r	p^{sd}	$\lvert\mathbb{S}\rvert$
`ela-1-4-10-02-sp`	8	1	4	10	0.2	`sp`	2 086 898 858
`elm-1-02-ud`	8	1	∞	–	0.2	`ud`	∞

Table 2 The start-to-destination floor probability distribution `ud` representing combined up and down traffic of equal intensities. f_1 and f_2 denote arbitrary start and destination floors with $f_1, f_2 \in \{2,\ldots,8\}$.

start floor	destination floor	
	1	f_2
1	–	1/14
f_1	1/14	–

Table 3 The start-to-destination floor probability distribution `sp` representing a special situation.

start floor	destination floor							
	1	2	3	4	5	6	7	8
1	–	–	–	1/20	–	3/20	–	2/20
2	–	–	–	–	–	–	–	–
3	–	–	–	–	–	–	–	–
4	2/20	–	–	–	–	1/20	–	1/20
5	–	–	–	–	–	–	–	–
6	3/20	–	–	–	–	–	2/20	1/20
7	–	–	–	–	–	–	–	–
8	2/20	–	–	–	–	2/20	–	–

4.1 Studied Instances

We introduce the instances of the two described Markov decision processes for online elevator control that are studied in the sequel. Recall that the two models differ only in the stage costs. In each case we can specify an instance by the following data:

- a number of floors $n_F \in \mathbb{N}$ defining the set of floors $F := \{1,\ldots,n_F\}$,
- a number of elevators $n_E \in \mathbb{N}$ defining the elevator set $E := \{1,\ldots,n_E\}$,
- a waiting queue length $q \in \mathbb{N} \cup \{\infty\}$,
- a penalty cost $c_p \geq 1$,
- a probability $0 \leq p^r \leq 1$ that exactly one new request is released at a time slot, and
- a probability distribution for the start and destination floor of a new request given by a function $p^{sd}\colon F \times F \to \mathbb{R}$ with $p^{sd}(f,f) = 0$ for each floor $f \in F$ and $\sum_{f_1 \in F} \sum_{f_2 \in F} p^{sd}(f_1,f_2) = 1$, i. e., the probability that a new request has start floor $f_1 \in F$ and destination floor $f_2 \in F$ equals $p^{sd}(s,d)$.

The two instances we consider are given in Table 1. We keep the same names as in [31]. The instance `ela-1-4-10-02-sp` is a Markov decision process for the case

of minimizing the average waiting time, while the one for minimizing the maximum waiting time is called `elm-1-02-ud`. Here, we only look at problems featuring a single elevator (see [31] for more tests). We focus on two different distributions for the start and destination floors of new requests. On the one hand, we look at combined *up and down traffic*, i. e., for each transport request, Floor 1 is either its start floor or its destination floor (see Table 2). This setting is natural for a cargo elevator system in an automated warehouse, where goods are placed into storage and retrieved over time. On the other hand, Table 3 shows a special traffic situation that may be representative for some time in the course of a day. One can think of this situation as follows. Still, there are some requests arriving at Floor 1 to be placed into storage and some requests are retrieved, but only a subset of floors are currently utilized. Moreover, there is some *interfloor traffic*, i. e., requests have start and destination floors that are different from Floor 1. This may be due to production processes taking place or required relocations of the stored goods.

4.2 Reading the Charts

We aim at monitoring during the computation for a selected state $i_0 \in \mathbb{S}$ that is reached while running a simulation or real-world system the following quantity:

$$\varepsilon_{i_0}^{\alpha}(\pi) := \frac{v_{i_0}^{\alpha}(\pi) - v_{i_0}^{\alpha}}{v_{i_0}^{\alpha}} \quad \text{whenever } v_{i_0}^{\alpha} > 0, \tag{7}$$

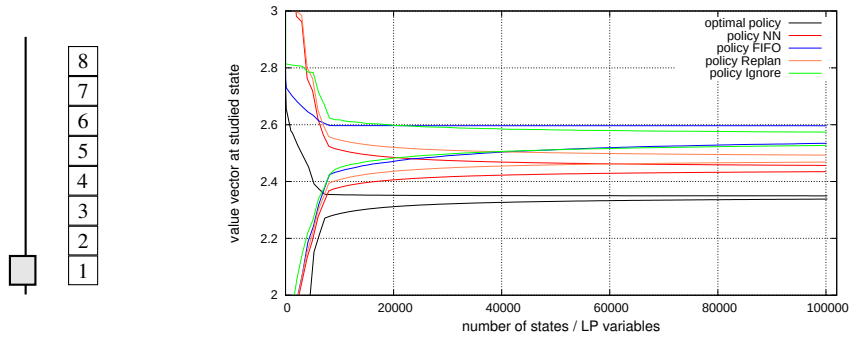
where π is a particular policy for the considered MDP. The value $\varepsilon_{i_0}^{\alpha}(\pi)$ gives the relative increase of the total α-discounted expected cost for the initial state $i_0 \in \mathbb{S}$ when using policy π or action $\pi(i_0)$ instead of an optimal policy. Since it is generally impossible to compute the quantities defined in Equation (7) exactly, we aim at providing lower bounds. This requires an upper bound on the component $v_{i_0}^{\alpha}$ of the optimal value vector and a lower bound on $v_{i_0}^{\alpha}(\pi)$ or $v_{i_0}^{\alpha}(\pi(i_0))$, respectively, which are all obtained by our approximation algorithm.

The evaluation figures are arranged as follows. One chart may show for one particular state i_0, the approximation progress of

- an optimal policy: $v_{i_0}^{\alpha}$ and
- a concrete policy π: $v_{i_0}^{\alpha}(\pi)$.

In the following we will refer to the values $v_{i_0}^{\alpha}$ and $v_{i_0}^{\alpha}(\pi)$ simply as the *optimal cost* and the *cost of policy* π, respectively.

For each cost value reported, we depict the progress of lower and upper bounds computed in the approximation process depending on the number of explored states and generated variables, respectively. Additionally, we will provide the best obtained lower bounds on the value $\varepsilon_{i_0}^{\alpha}(\pi)$ for each analyzed policy π.



(a) (b)

Fig. 1 (a) Trivial state i_2 in an elevator control MDP w. r. t. the average waiting time for a single elevator and 8 floors. The initial state i_2 has no waiting requests and the elevator at Floor 1 (b) Approximation results for the elevator control MDP with 8 floors, one elevator, 4 waiting slots per floor and the special request distribution. We have $\varepsilon^{\alpha}_{i_2}(\mathsf{NN}) \geq 3.6\,\%$, $\varepsilon^{\alpha}_{i_2}(\mathsf{FIFO}) \geq 7.9\,\%$, $\varepsilon^{\alpha}_{i_2}(\textsc{Replan}) \geq 5.1\,\%$, and $\varepsilon^{\alpha}_{i_2}(\textsc{Ignore}) \geq 7.5\,\%$.

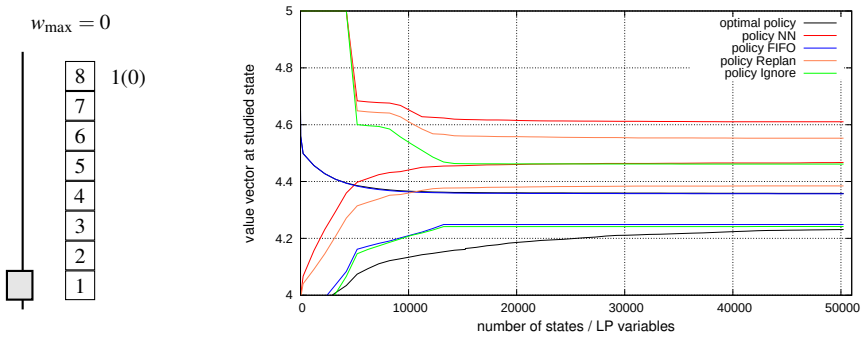
4.3 Approximation Results for the Average Waiting Time

For this test, we selected a discount factor of $\alpha = 0.8$ and the trivial initial state $i_2 = i_{\text{elv}}$[1] where no transport request is waiting and the elevator is situated at Floor 1. The associated approximation results for the MDP are depicted in Figure 1. Obviously, none of the considered policies (see Section 2 for a description of the policies) is really close to an optimal policy for the initial state i_2.

Observe the effectivity of the column generation: Our method proves that NN is not optimal using less than 10 000 states. It proves that NN is better than REPLAN using around 60 000 states. The proof that IGNORE and FIFO are worse than REPLAN takes around 50 000 states. After the generation of no more than 10 000 states, we know the cost of an optimal policy up to approximately 0.1, i.e., by then we have reached an accuracy of better than 5 %. Compared to this, the size of a static set of states determined by the formula of Theorem 2(i) for an approximation guarantee of 0.1 would be larger than the whole state space with 2 086 898 858 states.

The most interesting constructive observation we made relating to these results is the following: an optimal action at state i_2 is to move the elevator upwards. In the case no request is to be served by an elevator we face the task to position it such that future requests can be handled well. This issue is often referred to as the *parking policy* in the literature. Obviously, all of the considered policies do trivial parking, i. e., an elevator that is not dedicated to serve a request simply waits at its current floor. Our approximation method proves that this parking policy is not optimal for the state i_2. This result motivates to compare the actions WAIT, MOVE_DOWN, and MOVE_UP also for each state, where no request is waiting and the elevator is located at an arbitrary floor in $F \setminus \{1\}$. That is, for each state $i = ((\sigma_f)_{f \in F}, f_e, d_e)$ with $\sigma_f = \emptyset$ for each $f \in F$,

[1] We chose the notation i_2 instead of i_0 to be consistent with the states considered in [31]



(a) (b)

Fig. 2 (a) State i_1 in an elevator control MDP w. r. t. the maximum waiting time for a single elevator and 8 floors. The initial state i_1 has one waiting requests at Floor 8 with destination at Floor 1, zero waiting time so far, and the elevator in Floor 1. (b) Approximation results for the elevator control MDP with 8 floors, 1 elevator, infinity queuing capacity, and up-down traffic. We have $\varepsilon^{\alpha}_{i_1}(\mathsf{NN}) \geq 2.5\,\%$, $\varepsilon^{\alpha}_{i_1}(\mathsf{FIFO}) \geq 0\,\%$, $\varepsilon^{\alpha}_{i_1}(\textsc{Replan}) \geq 0.6\,\%$, and $\varepsilon^{\alpha}_{i_1}(\textsc{Ignore}) \geq 0\,\%$.

$d_e = 0$, and arbitrary floor $f_e \in F$, we evaluate the total expected 0.8-discounted costs of all feasible actions. This way, we could determine a unique optimal action for each of these states according to the approach due to Lemma 4. It turned out that the action WAIT is only optimal if the elevator is located at Floor 6. Otherwise, moving the elevator closer to Floor 6 can be proven to be optimal. Thus, we obtained an optimal parking policy for the corresponding MDP.

4.4 Approximation Results for the Maximum Waiting Time

Next, we analyze the performance of the policies NN, FIFO, REPLAN, and IGNORE when the objective is to minimize the maximum waiting time of a request. That is, we study the proposed elevator control MDP w. r. t. the maximum waiting time. We report on the results for the MDP with 8 floors, 1 elevator, infinity queuing capacity and the up-down traffic distribution from Table 2. The initial state i_1 has one waiting requests at Floor 8, a maximal waiting time of 0 so far, and the elevator in Floor 1. The discount factor α is set to 0.8 again.

Figure 2 shows the associated results obtained by our approximation algorithm for the initial state i_1. Obviously, the policy NN performs badly, and is provably worse than FIFO and IGNORE. Moreover, the cost of REPLAN is shown to be greater than the cost of FIFO and the optimal cost.

Although the studies for average and maximum waiting time elevator control MDPs can only partitially reflect the behavior observed in simulations, we want to point out that our analysis provided useful information to improve existing online algorithms. For instance, let us consider the policy NN. Computational results (see [31]) reveale that NN has the following weaknesses:

- NN does not employ a parking policy,
- its tie-breaking rule may lead to bad decisions, and
- the maximum waiting times achieved by NN are quite bad.

Due to these observations, we define the policy NNPARK-f as the following modification of NN:

- If the elevator is empty and there does not exist a waiting request, move the elevator towards Floor f.
- If the elevator is empty and the nearest waiting request is not unique, serve that request with the greater waiting time first.

The parameter f can in general be computed by checking the optimality of parking at every floor by our method. As already mentioned, $f = 6$ is optimal in our example.

In order to focus even more on good maximum waiting times, we propose the following extension of NNPARK-f: if the elevator is empty and there exists a waiting request whose current waiting time equals the maximum waiting so far, this request is served next ignoring all other requests. We denote this online algorithm by NNMAXPARK-f.

4.5 Assessment of Long-Term Effects

The weakness of our tool in the elevator control problem is the following: since a relatively small discounting factor of 0.8 was necessary to reach conclusive results in the computations, long-term effects that, e.g., would rule out FIFO as an efficient policy (compare [16]) cannot be detected. Maybe, the computation starting in a different start state (full system) can yield more information, but the general problem persists. In other words, further research is needed to capture long-term effects. So far, our method is – used in isolation – only well-suited to assess the short-term performance issues of policies.

However, once improved policies have been identified, it is possible to assess their long-term behavior by simulation. We therefore simulated our modifications of NN. The system defined by the Markov decision process cconsists of one elevator, eight floors, and the objective is to minimize the maximum waiting time, i. e., the queue length is infinite: We simulated for 10 000 time steps and computed average values for the observed average and maximum waiting times for 100 simulation runs. Table 4 shows simulation results for two Markov decision processes featuring a probability of $p^r = 0.1$ for the arrival of a new request at a time slot (this generates quite a high load). Obviously, NNPARK-f improves over NN for both, average and the maximum waiting times. Moreover, the NNMAXPARK-f achieves by far the best maximum waiting times, while the average waiting times are similar to those of the originial online algorithm NN, but slightly inferior compared to NNPARK-f.

We can therefore recommend to replace NN by NNMAXPARK-f on the basis of the evidence provided by our new analysis method and the evidence created by a standard simulation technique.

Table 4 Average value for the average and maximum waiting times achieved by the online algorithms NN and its variants NNPARK-f and NNMAXPARK-f according to 100 simulation runs for 10000 time steps. The parking floor is chosen to be $f = 6$ for the MDP with start-to-destination probability distributions `sp` (see Table 3) and $f = 1$ for `ud` (see Table 2).

Probability distribution	NN		NNPARK-f		NNMAXPARK-f	
	avg.	max.	avg.	max.	avg.	max.
`sp`	13.66	116.15	13.34	113.38	13.69	98.76
`ud`	12.26	139.33	11.93	128.59	12.26	97.94

5 Conclusion

In this paper we analyzed policies for the elevator control problem by a new approximation algorithm of the cost-to-go vector based on column generation for the exact linear programming formulation in MDP theory. The key-learnings led to the design of two new policies with provable better performance in critical states. This result was confirmed by a simulation study that is independent of our tool. We believe, that the ability of the tool to reveal weak spots (i.e., states in which decisions are far from optimal) of otherwise not-so-bad policies can help to selectively modify widely accepted policies in states in which they fail. Further research is needed in finding fast solutions to the pricing problem and state dependent bounds. It can be expected that only application specific methods will fit the bill.

6 Acknowledgements

We thank the referee for valuable suggestions on the presentation of the paper.

References

1. Adelman, D., Mersereau, A.: Relaxations of weakly coupled stochastic dynamic programs. Operations Research **56**(3), 712–727 (2008)
2. Ascheuer, N., Krumke, S.O., Rambau, J.: Online dial-a-ride problems: Minimizing the completion time. In: Proceedings of the 17th International Symposium on Theoretical Aspects of Computer Science, vol. 1770, pp. 639–650. Springer (2000)
3. Barto, A.G., Bradtke, S.J., Singh, S.P.: Learning to act using real-time dynamic programming. Artificial Intelligence **72**, 81–138 (1995)
4. Bertsekas, D.P.: Dynamic Programming and Optimal Control, vol. 1 and 2, 2nd edn. Athena Scientific, Belmont (2001)
5. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-Dynamic Programming, vol. 1, 1st edn. Athena Scientific, Belmont (1996)
6. Crites, R.H., Barto, A.G.: Elevator group control using multiple reinforcement learning agents. Machine Learning **33**(2–3), 235–262 (1998)
7. Dean, T.L., Kaelbling, L.P., Kirman, J., Nicholson, A.E.: Planning with deadlines in stochastic domains. In: AAAI, pp. 574–579 (1993)
8. d'Epenoux, F.: A probabilistic production and inventory problem. Management Science **10**(1), 98–108 (1963)
9. Desai, V.V., Farias, V.F., Moallemi, C.C.: Approximate dynamic programming via a smoothed linear program. Working paper, Graduate School of Business, Columnbia University (2009)

10. Desaulniers, G., Desrosiers, J., Solomon, M.M. (eds.): Column generation. GERAD 25th anniversary series. Springer (2005)
11. de Farias, D.P., van Roy, B.: A cost-shaping linear program for average-cost approximate dynamic programming with performance guarantees. Mathematics of Operations Research **31**(3), 597–620 (2006)
12. de Farias, D.P., Van Roy, B.: The linear programming approach to approximate dynamic programming. Operations Research **51**(6), 850–865 (2003)
13. de Farias, D.P., Van Roy, B.: On constraint sampling in the linear programming approach to approximate dynamic programming. Mathematics of Operations Research **29**(3), 462–478 (2004)
14. de Farias, D.P., Weber, T.: Choosing the cost vector of the linear programming approach to approximate dynamic programming. In: Decision and Control, pp. 67–72 (2008)
15. Feinberg, E.A., Shwartz, A. (eds.): Handbook of Markov Decision Processes: Methods and Applications. Kluwer Academic Publishers (2002)
16. Friese, P., Rambau, J.: Online-optimization of a multi-elevator transport system with reoptimization algorithms based on set-partitioning models. Discrete Appl. Math. **154**(13), 1908–1931 (2006). Also available as ZIB Report 05-03
17. Grötschel, M., Hauptmeier, D., Krumke, S.O., Rambau, J.: Simulation studies for the online dial-a-ride problem. Report 99–09, ZIB (1999)
18. Hauptmeier, D., Krumke, S.O., Rambau, J.: The online Dial-a-Ride problem under reasonable load. In: CIAC 2000, *Lecture Notes in Computer Science*, vol. 1767, pp. 125–136. Springer (2000)
19. Hauptmeier, D., Krumke, S.O., Rambau, J.: The online dial-a-ride problem under reasonable load. In: Proceedings of the 4th Italian Conference on Algorithms and Complexity, *Lecture Notes in Computer Science*, vol. 1767, pp. 137–149. Springer (2000)
20. Heinz, S., Kaibel, V., Peinhardt, M., Rambau, J., Tuchscherer, A.: LP-based local approximation for Markov decision problems. Report 06–20, ZIB (2006)
21. Hiller, B., Klug, T., Tuchscherer, A.: Improving the performance of elevator systems using exact reoptimization algorithms. In: Proceedings of MAPSP (2009)
22. Hiller, B., Klug, T., Tuchscherer, A.: Improved destination call elevator control algorithms for up peak traffic. In: Operations Research Proceedings 2011. Springer (2010). To appear
23. Hiller, B., Tuchscherer, A.: Real-time destination-call elevator group control on embedded microcontrollers. In: Operations Research Proceedings 2007. Springer (2008)
24. Kearns, M.J., Mansour, Y., Ng, A.J.: A sparse sampling algorithm for near-optimal planning in large Markov decision processes. In: International Joint Conferences on Artificial Intelligence, pp. 1324–1331 (1999)
25. Krumke, S.O., Rambau, J.: Stability with uniform bounds for on-line dial-a-ride problems under reasonable load. In: R. Johansson, A. Rantzer (eds.) Distributed Decision Making and Control, *Lecture Notes in Control and Information Sciences*, vol. 417, chap. 17, pp. 387–412. Springer (2012)
26. Powell, W.B.: Approximate Dynamic Programming: Solving the Curses of Dimensionality, 1st edn. John Wiley and Sons, Inc., Hoboken, New Jersey (2007)
27. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming, 2nd edn. John Wiley and Sons, Inc., Hoboken, New Jersey (2005)
28. Schröder, J.: Advanced dispatching: Destination hall calls + instant car-to-call assignments: M10. Elevator World pp. 40–46 (1990)
29. Schweitzer, P.J., Seidmann, A.: Generalized polynomial approximations in Markov decision processes. Journal of Mathematical Analysis and Applications **110**, 568–582 (1985)
30. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction, 1st edn. MIT Press, Cambridge (1998)
31. Tuchscherer, A.: Local evaluation of policies for discounted Markov decision problems. Ph.D. thesis, Technische Universität Berlin (2010)
32. Veatch, M.H., Walker, N.: Approximate linear programming for network control: Column generation and subproblems. Working paper, Gordon College (2008)