Minimizing Running Costs in Consumption Systems

Tomáš Brázdil, David Klaška, Antonín Kučera, and Petr Novotný

Faculty of Informatics, Masaryk University, Brno, Czech Republic

Abstract. A standard approach to optimizing long-run running costs of discrete systems is based on minimizing the *mean-payoff*, i.e., the long-run average amount of resources ("energy") consumed per transition. However, this approach inherently assumes that the energy source has an unbounded capacity, which is not always realistic. For example, an autonomous robotic device has a battery of finite capacity that has to be recharged periodically, and the total amount of energy consumed between two successive charging cycles is bounded by the capacity. Hence, a controller minimizing the mean-payoff must obey this restriction. In this paper we study the controller synthesis problem for *consumption systems* with a finite battery capacity, where the task of the controller is to minimize the mean-payoff while preserving the functionality of the system encoded by a given linear-time property. We show that an optimal controller always exists, and it may either need only finite memory or require infinite memory (it is decidable in polynomial time which of the two cases holds). Further, we show how to compute an effective description of an optimal controller in polynomial time. Finally, we consider the limit values achievable by larger and larger battery capacity, show that these values are computable in polynomial time, and we also analyze the corresponding rate of convergence. To the best of our knowledge, these are the first results about optimizing the long-run running costs in systems with bounded energy stores.

1 Introduction

A standard tool for modelling and analyzing the long-run average running costs in discrete systems is *mean-payoff*, i.e., the average amount of resources (or "energy") consumed per transition. More precisely, a system is modeled as a finite directed graph C, where the set of states S corresponds to configurations, and transitions model the discrete computational steps. Each transition is labeled by a non-negative integer specifying the amount of energy consumed by a given transition. Then, to every run ϱ in C, one can assign the associated *mean-payoff*, which is the limit of average energy consumption per transition computed for longer and longer prefixes of ϱ. A basic algorithmic task is to find a suitable *controller* for a given system which minimizes the mean-payoff. Recently, the problem has been generalized by requiring that the controller should also achieve a given *linear time property* φ, i.e., the run produced by a controller should satisfy φ while minimizing the mean-payoff (see, e.g., [14]). This is motivated by the fact that the system is usually required to achieve some functionality, and not just "run" with minimal average costs.

Note that in the above approach, it is inherently assumed that all transitions are always enabled, i.e., the amount of energy consumed by a transition is always available. In this paper, we study the long-run average running costs in systems where the energy

stores ("tanks" or "batteries") have a *finite* capacity $cap \in \mathbb{N}$. As before, the energy stored in the battery is consumed by performing transitions, but if the amount of energy currently stored in the battery is smaller than the amount of energy required by a given transition, then the transition is disabled. From time to time, the battery must be reloaded, which is possible only in certain situations (e.g., when visiting a petrol station). These restrictions are directly reflected in our model, where some states of C are declared as *reload states*, and the run produced by a controller must be *cap-bounded*, i.e., the total amount of energy consumed between two successive visits to reload states does not exceed *cap*.

The main results of this paper can be summarized as follows. Let C be a system (with a given subset of reload states) and φ a linear-time property encoded as a non-deterministic Büchi automaton.

(A) We show that for a given capacity $cap \in \mathbb{N}$ and a given state s of C, there exists a controller μ *optimal* for s which produces a *cap*-bounded run satisfying φ while minimizing the mean payoff. Further, we prove that there is a dichotomy in the structural complexity of μ, i.e., one of the following possibilities holds:
   - The controller μ can be constructed so that it has finitely many memory elements and can be compactly represented as a *counting controller* κ which is computable in time polynomial in the size of C and *cap* (all integer constants are encoded in *binary*).
   - The controller μ *requires* infinite memory (i.e., every optimal controller has infinite memory) and there exists an optimal *advancing controller* π which admits a finite description computable in time polynomial in the size of C and *cap*.

   Further, we show that it is decidable in polynomial time which of the two possibilities holds.

(B) For every state s of C, we consider its *limit value*, which is the *inf* of all mean-payoffs achievable by controllers for larger and larger battery capacity. We show that the limit value is computable in polynomial time. Further, we show that the problem whether the limit value is achievable by some *fixed* finite battery capacity is decidable in polynomial time. If it is the case, we give an explicit upper bound for *cap*; and if not, we give an upper bound for the difference between the limit value and the best mean-payoff achievable for a given capacity *cap*.

Technically, the most difficult part is (A), where we need to analyze the structure of optimal controllers and invent some tricks that allow for compact representation and computation of optimal controllers. Note that all constants are encoded in binary, and hence we cannot afford to construct any "unfoldings" of C where the current battery status (i.e., an integer between 0 and *cap*) is explicitly represented, because such an unfolding is exponentially larger than the problem instance. This is overcome by non-trivial insights into the structure of optimal controllers.

Previous and related work. A combination of mean-payoff and linear-time (parity) objectives has been first studied in [14] for 2-player games. It has been shown that optimal strategies exist in such games, but they may require infinite memory. Further, the values can be computed in time which is pseudo-polynomial in the size of the game and exponential in the number of priorities. Another closely related formalisms are *energy*

games and *one-counter games*, where each transition can both increase and decrease the amount of energy, and the basic task of the controller is to avoid the situation when the battery is empty. Energy games with parity objectives have been considered in [10]. In these games, the controller also needs to satisfy a given parity condition apart of avoiding zero. Polynomial-time algorithms for certain subclasses of "pure" energy games (with zero avoidance objective only) have recently been designed in [13]. Energy games with capacity constraints were studied in [17]. Here it was shown, that deciding whether a given one-player energy game admits a run along which the accumulated reward stays between 0 and a given positive capacity is already an NP-hard problem. *One-counter Markov decision processes* and *one-counter stochastic games*, where the counter may change at most by one in each transition, have been studied in [6, 5] for the objective of *zero reachability*, which is dual to zero avoidance. It has been shown that for one-counter MDPs (both maximizing and minimizing), the existence of a controller that reaches zero with probability one is in **P**. If such a controller exists, it is computable in polynomial time. For one-counter stochastic games, it was shown that the same problem is in **NP** $\cap$ **co-NP**. In [9], it was shown how to compute an ε-optimal controller minimizing the expected number of transitions needed to visit zero in one-counter MDPs. Another related model with only one counter are *energy Markov decision processes* [11], where the counter updates are arbitrary integers encoded in binary, and the controller aims at maximizing the probability of all runs that avoid visiting zero and satisfy a given parity condition. The main result of [11] says that the existence of a controller such that the probability of all runs satisfying the above condition is equal to one for a sufficiently large initial counter value is in **NP** $\cap$ **co-NP**. Yet another related model are *solvency games* [3], which can be seen as rather special one-counter Markov decision processes (with counter updates encoded in binary). The questions studied in [3] concern the structure of an optimal controller for maximizing the probability of all runs that avoid visiting negative values, which is closely related to zero avoidance.

There are also results about systems with more than one counter (resource). Examples include games over vector addition systems with states [8], *multiweighted energy games* [17, 4], *generalized energy games* [12], *consumption games* [7], etc. We refer to [18] for a more detailed overview.

2 Preliminaries

The sets of all integers, positive integers, and non-negative integers are denoted by $\mathbb{Z}$, $\mathbb{N}$, and $\mathbb{N}_0$, respectively. Given a set A, we use $|A|$ to denote the cardinality of A. The encoding size of a given object B is denoted by $\|B\|$. In particular, all integer numbers are encoded in *binary*, unless otherwise stated. The i-th component of a vector (or tuple) v is denoted by $v[i]$.

A *labelled graph* is a tuple $G = (V, \to, L, \ell)$ where V is a non-empty finite set of *vertices*, $\to \subseteq V \times V$ is a set of *edges*, L is a non-empty finite set of *labels*, and ℓ is a function which to every edge assigns a label of L. We write $s \xrightarrow{a} t$ if $s \to t$ and a is the label of (s, t).

A *finite path* in G of *length* $n \in \mathbb{N}_0$ is a finite sequence $\alpha \equiv v_0 \dots v_n$ of vertices such that $v_i \to v_{i+1}$ for all $0 \leq i < n$. The length of α is denoted by $len(\alpha)$, and the label of

$v_i \to v_{i+1}$ is denoted by a_i. An *infinite path* (or *run*) in G is an infinite sequence of vertices ϱ such that every finite prefix of ϱ is a finite path in G. Finite paths and runs in G are also written as sequences of the form $v_0 \xrightarrow{a_0} v_1 \xrightarrow{a_1} v_2 \xrightarrow{a_2} \cdots$. Given a finite or infinite path $\varrho \equiv v_0 v_1 \ldots$ and $i \in \mathbb{N}_0$, we use $\varrho(i)$ to denote the i-th vertex v_i of ϱ, and $\varrho_{\leq i}$ to denote the prefix $v_0 \ldots v_i$ of ϱ of length i.

A finite path $\alpha \equiv v_0 \ldots v_n$ in G is a *cycle* if $n \geq 1$ and $v_0 = v_n$, and a *simple cycle* if it is a cycle and $v_i \neq v_j$ for all $0 \leq i < j < n$. Given a finite path $\alpha \equiv v_0 \ldots v_n$ and a finite or infinite path $\varrho \equiv u_0 u_1 \ldots$ such that $v_n = u_0$, we use $\alpha \cdot \varrho$ to denote the *concatenation* of α and ϱ, i.e., the path $v_0 \ldots v_n u_1 u_2 \ldots$ Further, if α is a cycle, we denote by α^ω the infinite path $\alpha \cdot \alpha \cdot \alpha \cdots$.

In our next definition, we introduce consumption systems that have been informally described in Section 1. Recall that an optimal controller for a consumption system should minimize the mean-payoff of a *cap*-bounded run and satisfy a given linear-time property φ (encoded by a non-deterministic Büchi automaton $\mathcal{B}$). For technical convenience, we assume that $\mathcal{B}$ has already been multiplied with the considered consumption system (i.e., the synchronized product has already been constructed[1]). Technically, we declare some states in consumption systems as accepting and require that a *cap*-bounded run visits an accepting state infinitely often.

Definition 1. *A* consumption system *is a tuple $C = (S, \to, c, R, F)$ where S is a finite non-empty set of* states*, $\to \subseteq S \times S$ is a* transition relation*, c is a function assigning a non-negative integer* cost *to every transition, $R \subseteq S$ is a set of* reload states*, and $F \subseteq S$ a non-empty set of* accepting states*. We assume that $\to$ is* total*, i.e., for every $s \in S$ there is some $t \in S$ such that $s \to t$.*

The encoding size of C is denoted by $\|C\|$ (transition costs are encoded in binary). All notions defined for labelled graphs naturally extend to consumption systems.

The *total cost* of a given finite path $\alpha \equiv s_0 \xrightarrow{c_0} s_1 \xrightarrow{c_1} \cdots \xrightarrow{c_n} s_{n+1}$ is defined as $c(\alpha) = \sum_{i=0}^{n} c_i$, and the *mean cost* of α as $MC(\alpha) = c(\alpha)/(n+1)$. Further, we define the *end cost* of α as the total cost of the longest suffix $s_i \xrightarrow{c_i} \cdots \xrightarrow{c_n} s_{n+1}$ of α such that $s_{i+1}, \ldots, s_{n+1} \notin R$ (intuitively, the end cost of α is the total amount of resources consumed since the last reload).

Let $cap \in \mathbb{N}$. We say that a finite or infinite path $\varrho \equiv s_0 \xrightarrow{c_0} s_1 \xrightarrow{c_1} s_2 \xrightarrow{c_2} \cdots$ is *cap-bounded* if the end cost of every finite prefix of ϱ is bounded by *cap* (intuitively, this means that the total amount of resources consumed between two consecutive visits to reload states in ϱ is bounded by *cap*). Further, we say a run ϱ in C is *accepting* if $\varrho(i) \in F$ for infinitely many $i \in \mathbb{N}$. For every run ϱ in C we define

$$Val_C^{cap}(\varrho) = \begin{cases} \limsup_{i \to \infty} MC(\varrho_{\leq i}) & \text{if } \varrho \text{ is } cap\text{-bounded and accepting;} \\ \infty & \text{otherwise.} \end{cases}$$

The *cap-value* of a given state $s \in S$ is defined by

$$Val_C^{cap}(s) = \inf_{\varrho \in Run(s)} Val_C^{cap}(\varrho)$$

[1] It will become clear later that $\mathcal{B}$ being non-deterministic is not an obstacle here, since we work in a non-stochastic one-player setting.

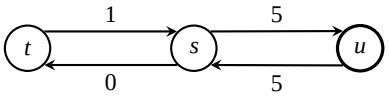

Fig. 1: An optimal controller may require memory of exponential size. Here $R = \{u\}$ and $F = S$.

where $Run(s)$ is the set of all runs in C initiated in s. Intuitively, $Val_C^{cap}(s)$ is the minimal mean cost of a *cap*-bounded accepting run initiated in s. The *limit value* of s is defined by $Val_C(s) = \lim_{cap\to\infty} Val_C^{cap}(s)$.

Definition 2. *Let $C = (S, \to, c, R, F)$ be a consumption system. A* controller *for C is a tuple $\mu = (M, \sigma_n, \sigma_u, m_0)$ where M is a set of* memory elements, *$\sigma_n : S \times M \to S$ is a* next function *satisfying $s \to \sigma_n(s, m)$ for every $(s, m) \in S \times M$, $\sigma_u : S \times M \to M$ is an* update function, *and m_0 is an* initial memory element. *If M is finite, we say that μ is a* finite-memory *controller (FMC).*

For every finite path $\alpha = s_0 \dots s_n$ in C, we use $\hat{\sigma}_u(\alpha)$ to denote the unique memory element "entered" by μ after reading α. Formally, $\hat{\sigma}_u(\alpha)$ is defined inductively by $\hat{\sigma}_u(s_0) = \sigma_u(s_0, m_0)$, and $\hat{\sigma}_u(s_0 \dots s_{n+1}) = \sigma_u(s_{n+1}, \hat{\sigma}_u(s_0 \dots s_n))$. Observe that for every $s_0 \in S$, the controller μ determines a unique run $run(\mu, s_0)$ defined as follows: the initial state of $run(\mu, s_0)$ is s_0, and if $s_0 \dots s_n$ is a prefix of $run(\mu, s_0)$, then the next state is $\sigma_n(s_n, \hat{\sigma}_u(s_0 \dots s_n))$. The size of a given FMC μ is denoted by $\|\mu\|$ (in particular, note that $\|\mu\| \geq |M|$).

Definition 3. *Let C be a consumption system, μ a controller for C, and $cap \in \mathbb{N}$. We say that μ is cap-*optimal *for a given state s of C if $Val_C^{cap}(run(\mu, s)) = Val_C^{cap}(s)$.*

As we shall see, an optimal controller for s always exists, but it may require infinite memory. Further, even if there is a FMC for s, it may require exponentially many memory elements. To see this, consider the simple consumption system of Fig. 1. An optimal controller for s has to (repeatedly) perform $cap - 10$ visits to t and then one visit to the only reload state u, which requires $cap - 10$ memory elements (recall that *cap* is encoded in binary). Further examples of a non-trivial optimal behaviour can be found in Appendix A.

To overcome these difficulties, we introduce a special type of finite-memory controllers called *counting controllers*, and a special type of infinite memory controllers called *advancing controllers*.

Intuitively, memory elements of a counting controller are pairs of the form (r, d) where r ranges over a finite set *Mem* and d is a non-negative integer of a bounded size. The next and update functions depend only on r and the information whether d is zero or positive. The update function may change (r, d) to some (r', d') where d' is obtained from d by performing a *counter action*, i.e., an instruction of the form *dec* (decrement), *noc* (no change), or *reset*(n) where $n \in \mathbb{N}$ (reset the value to n). Hence, counting controllers admit a compact representation which utilizes the special structure of memory elements and the mentioned restrictions.

Definition 4. *Let $C = (S, \to, c, R, F)$ be a consumption system. A* counting controller *for C is a tuple $\kappa = (Mem, \sigma_n^+, \sigma_n^0, Act, \sigma_u^+, \sigma_u^0, r_0)$ where*

– *Mem is a finite set of* basic memory elements,
– $\sigma_n^+, \sigma_n^0 : S \times Mem \to S$ *are* positive *and* zero next functions *satisfying* $s \to \sigma_n^+(s,r)$ *and* $s \to \sigma_n^0(s,r)$ *for every* $(s,r) \in S \times Mem$*, respectively,*
– *Act is a finite set of* counter actions *(note that Act may contain instructions of the form reset*(n) *for different constants* n*);*
– $\sigma_u^+ : S \times Mem \to Mem \times Act$ *is a* positive update function,
– $\sigma_u^0 : S \times Mem \to Mem \times (Act \setminus \{dec\})$ *is a* zero update function,
– $r_0 \in Mem$ *is an initial basic memory element.*

The encoding size of a counting controller κ is denoted by $||\kappa||$, where all constants used in counter actions are encoded in binary.

The functionality of a counting controller $\kappa = (Mem, \sigma_n^+, \sigma_n^0, Act, \sigma_u^+, \sigma_u^0, r_0)$ is determined by its associated finite-memory controller $\mu_\kappa = (M, \sigma_n, \sigma_u, m_0)$ where

– $M = Mem \times \{0, \ldots, k_{max}\}$ where k_{max} is the largest n such that $reset(n) \in Act$ (or 0 if no such n exists);
– $\sigma_n(s,(r,d)) = \sigma_n^{\odot}(s,r)$, where $\odot$ is either $+$ or 0 depending on whether $d > 0$ or $d = 0$, respectively;
– $\sigma_u(s,(r,d)) = (r',d')$, where r' is the first component of $\sigma_u^{\odot}(s,r)$, and d' is either d, $d-1$, or n, depending on whether the counter action in the second component of $\sigma_u^{\odot}(s,r)$ is *noc*, *dec*, or $reset(n)$, respectively (again, $\odot$ is either $+$ or 0 depending on whether $d > 0$ or $d = 0$);
– $m_0 = (r_0, 0)$.

Observe that $||\kappa||$ can be exponentially smaller than $||\mu_\kappa||$. Slightly abusing our notation, we write $run(\kappa, s_0)$ instead of $run(\mu_\kappa, s_0)$.

A counting controller κ can be seen as a program for a computational device with $O(||Mem||)$ control states and $\log(k_{max})$ bits of memory needed to represent the bounded counter. This device "implements" the functionality of μ_κ.

Definition 5. *Let* $C = (S, \to, c, R, F)$ *be a consumption system and* $s \in S$*. An* advancing controller *for* C *and* s *is a controller* π *for* C *such that* $run(\pi, s)$ *takes the form* $\alpha \cdot \beta \cdot \gamma \cdot \beta^2 \cdot \gamma \cdot \beta^4 \cdots \gamma \cdot \beta^{2^i} \cdots$ *where* $\beta(0) \neq \beta(i)$ *for all* $0 < i < len(\beta)$*.*

The encoding size of an advancing controller π, denoted by $||\pi||$, is given by the total encoding size of α, β, and γ. Typically, α and γ will be of polynomial length, but the length of β is sometimes exponential and in this case we use a counting controller to represent β compactly. Formally, we say that $||\pi||$ *is polynomial* in $||C||$ and $||cap||$ if α and γ are of polynomial length and there exists a counting controller $\kappa[\beta]$ such that $run(\kappa[\beta], \beta(0)) = \beta^\omega$ and $||\kappa||$ is polynomial in $||C||$ and $||cap||$.

An advancing controller π can be seen as a program for a computational device equipped with two unbounded counters (the first counter maintains the current i and the other counter is used to count from 2^i to zero; if the device cannot implement the '2^x' function directly, an auxiliary third counter may be needed). Also note that the device can use the program of $\kappa[\beta]$ as a subroutine to produce the finite path β (and hence also finite paths of the form β^{2^i}). Since $\beta(0) \neq \beta(i)$ for all $0 < i < len(\beta)$, the device simply simulates $\kappa[\beta]$ until revisiting $\beta(0)$.

3 The Results

In this section, we present the main results of our paper. Our first theorem concerns the existence and computability of values and optimal controllers in consumption systems.

Theorem 6. *Let C be a consumption system, $cap \in \mathbb{N}$, and s a state of C. Then $Val_C^{cap}(s)$ is computable in polynomial time (i.e., in time polynomial in $||C||$ and $||cap||$, where cap is encoded in binary). Further, there exists an optimal controller for s. The existence of an optimal* finite memory *controller for s is decidable in polynomial time. If there exists an optimal FMC for s, then there also exists an optimal* counting *controller for s computable in polynomial time. Otherwise, there exists an optimal* advancing *controller for s computable in polynomial time.*

Our second theorem concerns the limit values, achievability of limit values, and the rate of convergence to limit values.

Theorem 7. *Let C be a consumption system, $cap \in \mathbb{N}$, and s a state of C. Then $Val_C(s)$ can be computed in polynomial time (i.e., in time polynomial in $||C||$).*

Further, the problem whether $Val_C(s) = Val_C^{cap}(s)$ for some sufficiently large $cap \in \mathbb{N}$ is decidable in polynomial time. If the answer is positive, then $Val_C(s) = Val_C^{cap}(s)$ for every $cap \geq 3 \cdot |S| \cdot c_{\max}$, where $c_{\max}$ is the maximal cost of a transition in C. Otherwise, for every $cap > 4 \cdot |S| \cdot c_{\max}$ we have that $Val_C^{cap}(s) - Val_C(s) \leq (3 \cdot |S| \cdot c_{\max})/(cap - 4 \cdot |S| \cdot c_{\max})$.

The next subsections are devoted to the proofs of Theorems 6 and 7. Due to space constrains, some proofs and algorithms have been shifted to Technical Appendix.

3.1 A Proof of Theorem 6

For the rest of this section, we fix a consumption system $C = (S, \to, c, R, F)$, a capacity $cap \in \mathbb{N}$, and an initial state $s \in S$.

An *admissibility witness* for a state $q \in S$ is a cycle γ initiated in q such that γ contains an accepting state and there is a cap-bounded run initiated in s of the form $\alpha \cdot \gamma^\omega$. We say that $q \in S$ is *admissible* if there is at least one admissibility witness for q.

Observe that if γ is an admissibility witness for a reload state q, then γ can be freely "inserted" into any cap-bounded run of the form $\xi \cdot \delta$ where $\delta(0) = q$ so that the run $\xi \cdot \gamma \cdot \delta$ is again cap-bounded. Such simple observations about admissibility witnesses are frequently used in our proof of Theorem 6, which is obtained in several steps:

(1) We show how to compute all states $t \in S$ such that $Val_C^{cap}(t) = \infty$. Note that if $Val_C^{cap}(t) = \infty$, then *every* controller is optimal in t. Hence, if $Val_C^{cap}(s) = \infty$, we are done. Otherwise, we remove all states with infinite value from C together with their adjacent transitions.
(2) We compute and remove all states $t \in S$ that are not reachable from s via a cap-bounded finite path. This "cleaning" procedure simplifies our considerations and it can be performed in polynomial time.
(3) We show that $Val_C^{cap}(s) = 0$ iff C contains a *simple* cycle with zero total cost initiated in an admissible state (such a cycle is called a *zero-cost* cycle). Next, we show that if there is a zero-cost cycle β containing an accepting state, then there

is an optimal FMC μ for s of *polynomial* size such that $run(\mu, s) = \alpha \cdot \beta^\omega$. Otherwise, *every* optimal controller for s has infinite memory, and we show how to compute finite paths α, γ of polynomial length such that the (*cap*-bounded) run $\varrho \equiv \alpha \cdot \beta \cdot \gamma \cdot \beta^2 \cdot \gamma \cdot \beta^4 \cdots \gamma \cdot \beta^{2^i} \cdots$ initiated in s satisfies $Val_C^{cap}(\varrho) = 0$. Thus, the finite paths α, β, and γ represent an optimal advancing controller of polynomial size. The existence of a zero-cost cycle (and the existence of a zero-cost cycle that contains an accepting state) is decidable in polynomial time. If a zero-cost cycle exists, we are done. Otherwise, we proceed to the next step.

(4) Now we assume that C does not contain a zero-cost cycle. We show that there exist
- a *cap*-bounded cycle β initiated in an admissible state such that $MC(\beta) \leq MC(\delta)$ for every *cap*-bounded cycle δ initiated in an admissible state, and $\beta(0) \neq \beta(i)$ for all $0 < i < len(\beta)$;
- a *cap*-bounded cycle $\hat{\beta}$ containing an accepting state such that $MC(\hat{\beta}) \leq MC(\hat{\delta})$ for every *cap*-bounded cycle $\hat{\delta}$ containing an accepting state.

We prove that $Val_C^{cap}(s) = MC(\beta)$. Further, we show the following:
- If $MC(\beta) = MC(\hat{\beta})$, then there exists an optimal FMC μ for s such that $run(\mu, s) = \alpha \cdot \hat{\beta}^\omega$, where α is a finite path of polynomial length. In general, $len(\hat{\beta})$ (and hence also $||\mu||$) is *exponential* in $||C||$ and $||cap||$. We show how to compute a *counting* controller $\kappa[\hat{\beta}]$ of *polynomial* size such that $run(\kappa[\hat{\beta}], \hat{\beta}(0)) = \hat{\beta}^\omega$. Since α is a finite path of polynomial length, we also obtain a counting controller κ of polynomial size such that $run(\kappa, s) = run(\mu, s)$.
- If $MC(\beta) < MC(\hat{\beta})$, then *every* optimal controller for s has infinite memory, and we show how to efficiently compute finite paths α, γ of polynomial length and a counting controller $\kappa[\beta]$ of polynomial size such that $run(\kappa[\beta], \beta(0)) = \beta^\omega$ and the run $\varrho \equiv \alpha \cdot \beta \cdot \gamma \cdot \beta^2 \cdot \gamma \cdot \beta^4 \cdots \gamma \cdot \beta^{2^i} \cdots$ initiated in s satisfies $Val_C^{cap}(\varrho) = Val_C^{cap}(s)$. Thus, we obtain an optimal advancing controller π for s of polynomial size.

We start with step (1).

Lemma 8. *Let $t \in S$. The problem whether $Val_C^{cap}(t) = \infty$ is decidable in polynomial time.*

The next lemma implements step (2).

Lemma 9. *Let $t \in S$. The existence of a cap-bounded path from s to t is decidable in polynomial time. Further, an example of a cap-bounded path from s to t (if it exists) is computable in polynomial time.*

We also need the following lemma which says that for every admissible state, there is an efficiently computable admissibility witness.

Lemma 10. *The problem whether a given $q \in S$ is admissible is decidable in polynomial time. Further, if q is admissible, then there are finite paths α, γ computable in polynomial time such that $\alpha \cdot \gamma^\omega$ is a cap-bounded run initiated in s and γ is an admissibility witness for q of length at most $6 \cdot |S|^2$.*

As we already indicated in the description of step (2), from now on we assume that all states of C have a finite value and are reachable from s via a *cap*-bounded finite path. Recall that a *zero-cost* cycle is a cycle in C initiated in an admissible state with zero total cost. Now we proceed to step (3).

Lemma 11. *We have that $Val_C^{cap}(s) = 0$ iff there exists a zero-cost cycle. Further, the following holds:*

1. *If there is a zero-cost cycle β containing an accepting state, then the run $\varrho \equiv \alpha \cdot \beta^\omega$, where α is a cap-bounded finite path from s to $\beta(0)$, satisfies $Val_C^{cap}(\varrho) = Val_C^{cap}(s)$. Hence, there is a FMC μ optimal for s where $||\mu||$ is polynomial in $||C||$ and $||cap||$.*
2. *If there is a zero-cost cycle β but no zero-cost cycle contains an accepting state, then every cap-optimal controller for s has infinite memory. Further, for a given zero-cost cycle β there exist finite paths α and γ computable in polynomial time such that the run $\varrho \equiv \alpha \cdot \beta \cdot \gamma \cdot \beta^2 \cdots \gamma \cdot \beta^{2^i} \cdots$ satisfies $Val_C^{cap}(\varrho) = Val_C^{cap}(s)$. Hence, there exist an advancing controller π optimal for s where $||\pi||$ is polynomial in $||C||$ and $||cap||$.*

Proof. If $Val_C^{cap}(s) = 0$, there is an accepting run ϱ initiated in s such that $Val_C^{cap}(\varrho) < 1/|S|$. Let ϱ' be an infinite suffix of ϱ such that all states that appear in ϱ' appear infinitely often in ϱ'. This means that all states that appear in ϱ' are admissible. Obviously, there is $k \in \mathbb{N}$ such that the cost of every transition $\varrho'(k+i) \to \varrho'(k+i+1)$, where $0 \leq i \leq |S| - 1$, is zero (otherwise, we would have $Val_C^{cap}(\varrho) = Val_C^{cap}(\varrho') \geq 1/|S|$), and hence there exists a zero-cost cycle.

Now assume that C contains a zero-cost cycle β containing an accepting state. Since there is a *cap*-bounded finite path α form s to $\beta(0)$ (see step (2) and Lemma 9), the run $\varrho \equiv \alpha \cdot \beta^\omega$ is *cap*-bounded and satisfies $Val_C^{cap}(\varrho) = 0$. Since the length of α and β is polynomial in $||C||$ and $||cap||$ (see Lemma 9), we obtain Claim 1.

Finally, assume that C contains a zero-cost cycle β but no zero-cost cycle in C contains an accepting state. Since $\beta(0)$ is admissible, there is a *cap*-bounded run $\alpha \cdot \gamma^\omega$ initiated in s where γ is an admissibility witness for $\beta(0)$. Note that the length of α and γ is polynomial in $||C||$ and $||cap||$ by Lemma 10, and the run $\varrho \equiv \alpha \cdot \beta \cdot \gamma \cdot \beta^2 \cdots \gamma \cdot \beta^{2^i} \cdots$ is accepting and *cap*-bounded. Further, a simple computation shows that $Val_C^{cap}(\varrho) = 0$. Hence, there exists an advancing controller π optimal for s such that $||\pi||$ is polynomial in $||C||$ and $||cap||$. It remains to show that there is no optimal finite memory controller for s. However, it suffices to realize that if μ is a finite memory controller, then $run(\mu, s)$ takes the form $\hat{\alpha} \cdot \hat{\beta}^\omega$, where $\hat{\beta}$ contains an accepting state. By our assumption, $c(\hat{\beta}) \neq 0$, which means that $Val_C^{cap}(\hat{\alpha} \cdot \hat{\beta}^\omega) \neq 0$. □

In the next lemma we show how to decide the existence of a zero-cost cycle efficiently, and how to construct an example of a zero-cost cycle if it exists. The same is achieved for zero-cost cycles containing an accepting state. Thus, we finish step (3).

Lemma 12. *The existence of a zero-cost cycle is decidable in polynomial time, and an example of a zero-cost cycle β (if it exists) is computable in polynomial time. The same holds for zero-cost cycles containing an accepting state.*

It remains to complete step (4), which is the most technical part of our proof. From now on we assume that C does not contain any zero-cost cycles.

We say that a cycle β in C is *reload-simple*, if every reload state appears at most once in β, i.e., for every $t \in R$ there is at most one $0 \leq i < len(\beta)$ satisfying $\beta(i) = t$. Further, we say that a cycle β is *T-visiting*, where $T \subseteq S$, if β is a *cap*-bounded reload-simple cycle initiated in an admissible reload state such that β contains a state of T. We say that β is an *optimal T-visiting cycle* if $MC(\beta) \leq MC(\delta)$ for every T-visiting cycle δ. Note that every state of a T-visiting cycle β is admissible.

Lemma 13. *If C does not contain any zero-cost cycle, then it contains an optimal F-visiting cycle and an optimal S-visiting cycle.*

Proof. We give an explicit proof just for F-visiting cycles (the argument for S-visiting cycles is very similar). First, we show that there is at least one F-visiting cycle, and then we prove that every F-visiting cycle has a bounded length. Thus, the set of all F-visiting cycles is finite, which implies the existence of an optimal one.

Since $Val_C^{cap}(s) < \infty$, there is a *cap*-bounded accepting run ϱ initiated in s. Note that if ϱ contained only finitely many occurrences of reload states, it would have to contain zero-cost cycle, which contradicts our assumption. Hence, ϱ contains infinitely many occurrences of a reload state and infinitely many occurrences of an accepting state. Let ϱ' be a suffix of ϱ such that every state that appears in ϱ' appears infinitely often in ϱ' (hence, all states that appear in ϱ' are admissible). We say that a subpath $\varrho'(i)\dots\varrho'(j)$ of ϱ' is *useless* if $\varrho'(i) = \varrho'(j) \in R$ and no accepting state is visited along this subpath. Let $\hat{\varrho}$ be a run obtained from ϱ' by removing all useless subpaths (observe that $\hat{\varrho}$ is still a *cap*-bounded accepting run). Then, there must be a subpath $\hat{\varrho}(i)\dots\hat{\varrho}(j)$ of $\hat{\varrho}$ such that the length of this subpath is positive, $\hat{\varrho}(i) = \hat{\varrho}(j) \in R$, the subpath visits an accepting state, and no reload state is visited more than once along $\hat{\varrho}(i)\dots\hat{\varrho}(j-1)$. Hence, this subpath is an F-visiting cycle.

Now let β be an F-visiting cycle. Then every state on β is admissible, which means that every simple cycle δ that is a subpath of β has positive cost, otherwise δ would be a zero-cost cycle. This implies that a maximal length of a subpath of β which does not contain any reload state is $(|S|+1)\cdot(cap+1)$ (because β is *cap*-bounded). From the reload-simplicity of β we get that $len(\beta) \leq |R|\cdot(|S|+1)\cdot(cap+1)$. □

We use *MCF* and *MCS* to denote the mean cost of an optimal F-visiting cycle and the mean cost of an optimal S-visiting cycle, respectively. Now we prove the following:

Lemma 14. *Suppose that C does not contain any zero-cost cycle. Then $Val_C^{cap}(s) = MCS \leq MCF$. Moreover, the following holds:*

1. *If $MCF = MCS$, then for every optimal F-visiting cycle β and every cap-bounded path α from s to $\beta(0)$ we have that the run $\varrho \equiv \alpha\cdot\beta^\omega$ satisfies $Val_C^{cap}(\varrho) = Val_C^{cap}(s)$. Hence, there exists an optimal FMC for s.*
2. *If $MCS < MCF$, then every cap-optimal controller for s has infinite memory. Further, for a given optimal S-visiting cycle β there exist finite paths α and γ computable in polynomial time such that the run $\varrho \equiv \alpha\cdot\beta\cdot\gamma\cdot\beta^2\cdots\gamma\cdot\beta^{2^i}\cdots$ satisfies $Val_C^{cap}(\varrho) = Val_C^{cap}(s)$. Hence, there exists an optimal advancing controller for s.*

Proof. Clearly, $MCS \leq MCF$, because every F-visiting cycle is also S-visiting. Now we show that for every run ϱ we have that $Val_C^{cap}(\varrho) \geq MCS$. This clearly holds for all non-accepting runs. Every accepting run ϱ must contain infinitely many occurrences of a reload state, otherwise it would contain a zero-cost cycle as a subpath, which contradicts our assumption. Let ϱ' be a suffix of ϱ initiated in a reload state such that every state which appears in ϱ' appears infinitely often in ϱ'. Then ϱ' takes the form $\beta_0\cdot\beta_1\cdot\beta_2\cdots$, where for every $i \geq 0$, the subpath β_i is a cycle initiated in a reload state. Every β_i can be decomposed into reload-simple cycles $\beta_{i,1},\beta_{i,2},\dots,\beta_{i,i_m}$ that are initiated in reload states (here the decomposition is meant in a graph-theoretical sense, i.e., a transition appears

b times on β_i if and only if $b = b_1 + \cdots + b_m$, where b_j is a number of occurrences of this transition on $\beta_{i,j}$). Each of these cycles is an S-visiting cycle (since every state on ϱ' is admissible) and clearly $MC(\varrho) = MC(\varrho') \geq \min_{i \geq 1} MC(\beta_i) \geq \min_{i \geq 0, 1 \leq j \leq i_m} MC(\beta_{i,j}) \geq MCS$.

The rest of the proof closely follows the proof of Lemma 11. First we consider the case when $MCF = MCS$, i.e., for every optimal F-visiting cycle β we have that $MC(\beta) = MCS$. If α is a *cap*-bounded path from s to $\beta(0)$, then we have that the run $\varrho \equiv \alpha \cdot \beta^\omega$ satisfies $Val_C^{cap}(\alpha \cdot \beta^\omega) = MCS = Val_C^{cap}(s)$, and hence there exists an optimal FMC for s.

If $MCS < MCF$, consider an optimal S-visiting cycle β. Since $\beta(0)$ is admissible, there is a *cap*-bounded run $\alpha \cdot \gamma^\omega$ initiated in s where γ is an admissibility witness for $\beta(0)$ and α and γ are computable in polynomial time (see Lemma 10. Further, the run $\varrho \equiv \alpha \cdot \beta \cdot \gamma \cdot \beta^2 \cdots \gamma \cdot \beta^{2^i} \cdots$ is accepting and *cap*-bounded, and one can easily show that $Val_C^{cap}(\varrho) = MCS = Val_C^{cap}(s)$. Hence, there exists an optimal advancing controller for s. Since every finite memory controller μ satisfies $run(\mu, s) \equiv \hat{\alpha} \cdot \hat{\beta}^\omega$ and $||cap||$. It remains to show that there is no optimal finite memory controller for s. For every FMC μ we have that $run(\mu, s) \equiv \hat{\alpha} \cdot \hat{\beta}^\omega$, where $\hat{\beta}$ is a cycle on a reload state containing an accepting state. Further, $Val_C^{cap}(\mu) = MC(\hat{\beta})$. The cycle $\hat{\beta}$ can be decomposed into reload-simple cycles on reloading states whose mean cost is at least MCS. Since at least one of these cycles is accepting and $MCF > MCS$, we obtain $MC(\hat{\beta}) > MCS$. □

Note that Lemma 14 does not specify any bound on the length of β and in general, this length can be exponential. Now we show that an optimal F-visiting cycle and an optimal S-visiting cycle can be represented by a counting controller constructible in polynomial time. This is the technical core of our construction which completes the proof of Theorem 6.

Lemma 15. *Suppose that C does not contain any zero-cost cycle, and let T be either S or R. Then there exist a counting controller κ and a reload state r computable in polynomial time such that $run(\kappa, r) = \beta^\omega$ where β is an optimal T-visiting cycle.*

3.2 A Proof of Lemma 15

We start by refining the notion of an optimal T-visiting cycle and identifying those cycles that can be represented by counting controllers of polynomial size.

A *segment* of a path β is a finite subpath η of β such that the first and the last state of η are reload states and η does not contain any other occurrence of a reload state. Note that every reload-simple cycle is composed of at most $|R|$ segments. Furthermore, we say that a finite path is *compact*, if it is a *cap*-bounded path of the form $\gamma \cdot \delta^k \cdot \gamma'$, where γ and γ' are finite paths satisfying $len(\gamma) + len(\gamma') \leq 5|S|^3$, δ is either a cycle of length at most $|S|$ or a path of length 0 (i.e., a state), and $k \leq cap$. A *compact segment* is a compact path that is also a segment.

Later we show that there is an optimal T-visiting cycle β such that every segment of β is a compact segment. Intuitively, such a cycle can be produced by a counting controller of polynomial size which has at most $|R|$ reset actions. However, this does not yet imply that such a counting controller can be efficiently constructed, because there are exponentially many possible compact segments. Hence, we need to show that we can restrict our attention to some set of compact segments of polynomial size.

We say that a compact segment $\gamma \cdot \delta^k \cdot \gamma'$ has a *characteristic* (r, q, t, m, n, b), where $r, t \in R$, $q \in S$, $m, n \in \mathbb{N}$ are such that $0 \leq m \leq 5|S|^3$ and $0 \leq n \leq |S|$, and $b \in \{0, 1\}$, if the following holds:

- $\gamma(0) = r$, $last(\gamma) = \gamma'(0) = q$, $last(\gamma') = t$, and $len(\gamma \cdot \gamma') = m$;
- $\delta(0) = q$, $len(\delta) = n$;
- we either have that $n = 0$ and $k = 1$, or $n > 0$ and then $c(\delta) > 0$ and k is the maximal number such that $\gamma \cdot \delta^k \cdot \gamma$ is a *cap*-bounded path;
- if $b = 1$, then $\gamma \cdot \gamma'$ contains a state of T;
- if δ contains a state of T, then $\gamma \cdot \gamma'$ also contains a state of T.

Note that for a given consumption system there are at most polynomially many distinct characteristics of compact segments. Also note that not all compact segments have a characteristic (because of the third and the fifth condition in the above definition), and conversely, some compact segments may have multiple characteristics (e.g., if a compact segment has a characteristic where $b = 1$, then it also has one where $b = 0$). Finally, note that for any compact segment $\gamma \cdot \delta^k \cdot \gamma'$ with a characteristic (r, q, t, m, n, b), the path $\gamma \cdot \gamma'$ is a compact segment with the characteristic $(r, q, t, m, 0, b)$.

A characteristic χ of a compact segment $\gamma \cdot \delta^k \cdot \gamma'$ imposes certain restrictions on the form of $\gamma \cdot \gamma'$ and δ. Such a compact segment is *optimal* for χ if $\gamma \cdot \gamma'$ and δ are paths of minimal cost among those that meet this restriction. Formally, a compact segment $\gamma \cdot \delta^k \cdot \gamma'$ with a characteristic $\chi = (r, q, t, m, n, b)$ is *optimal for* χ if

- $c(\gamma \cdot \gamma')$ is minimal among the costs of all segments with the characteristic $(r, q, t, m, 0, b)$, and
- $c(\delta)$ is minimal among the costs of all cycles of length n and positive cost, that are initiated in q, and that do not contain any reload state with a possible exception of q (if $n = 0$, we consider this condition to be satisfied trivially).

Lemma 16. *If there is at least one compact segment with a given characteristic χ, then there is also an optimal compact segment for χ. Moreover, all compact segments optimal for a given characteristic have the same total cost and length.*

Hence, to each of the polynomially many characteristics χ we can assign a segment optimal for χ and thus form a polynomial-sized candidate set of compact segments. The following lemma, which is perhaps the most intricate step in the proof of Lemma 15, shows that there is an optimal T-visiting cycle β such that every segment of β belongs to the aforementioned candidate set.

Lemma 17. *There is an optimal T-visiting cycle β whose every segment is a compact segment optimal for some characteristic.*

Given a characteristic χ, it is easy to compute a succinct representation of some compact segment optimal for χ, as the next lemma shows.

Lemma 18. *Given a characteristic χ, the problem whether the set of all compact segments with the characteristic χ is non-empty is decidable in polynomial time. Further, if the set is non-empty, then a tuple $(\gamma, \gamma', \delta, k)$ such that $\gamma \cdot \delta^k \cdot \gamma'$ is a compact segment optimal for χ is computable in polynomial time.*

For a given characteristic χ, we denote by $CTuple(\chi)$ the tuple $(\gamma, \gamma', \delta, k)$ returned for χ by the algorithm of Lemma 18 (if an optimal compact segment for χ does not exist, we put $CTuple(\chi) = \bot$), and by $CPath(\chi)$ the corresponding compact segment $\gamma \cdot \delta^k \cdot \gamma'$ ($CTuple(\chi) = \bot$, we put $CPath(\chi) = \bot$). The next lemma is a simple corollary to Lemma 16 and Lemma 17.

Lemma 19. *There is an optimal T-visiting cycle β such that every segment of β is of the form $CPath(\chi)$ for some characteristic χ.*

Now we can easily prove the existence of a polynomial-sized counting controller representing some optimal T-visiting cycle β. According to Lemma 19, there is a sequence $\chi_0, \chi_1, \ldots, \chi_j$ of at most $|R|$ characteristics such that $\beta = CPath(\chi_0) \cdot CPath(\chi_1) \cdots CPath(\chi_g)$ is an optimal T-visiting cycle. To iterate the cycle β forever (starting in $\beta(0)$), a counting controller requires at most $|R| \cdot n$ basic memory elements, where n is the maximal number of basic memory elements needed to produce a compact segment $CPath(\chi_i)$, for $0 \leq i \leq g$. So, consider a compact segment $CPath(\chi_i) = \gamma \cdot \delta^k \cdot \gamma'$. Note that $k \leq cap$ since $CPath(\chi_i)$ has a characteristic and thus $c(\delta) > 0$. To produce $CPath(\chi_i)$, the controller requires at most $5|S|^3$ basic memory elements to produce the prefix γ and the suffix γ' of $CPath(\chi_i)$, and at most $|S|$ basic memory elements to iterate the cycle δ (whose length is at most $|S|$) exactly k times. The latter task also requires counting down from $k \leq cap$ to 0. Overall, the counting controller producing β^ω needs a polynomial number of basic memory elements, and requires at most $|R|$ reset actions parameterized by numbers of encoding size at most $\log(cap)$. To compute such a counting controller, it clearly suffices to compute the aforementioned sequence of tuples $CTuple(\chi_0), \cdots, CTuple(\chi_g)$.

Now we can present the algorithm promised in Proposition 15. In the following, we use X to denote the set of all possible characteristics of compact segments in C, $X_{r,t}$ to denote the set of all characteristics of the form (r, q, t, m, n, b) for some q, m, n, b, and $X^1_{r,t}$ to denote the set of all characteristics of $X_{r,t}$ where the last component is equal to 1. The algorithm first computes the set $R' \subseteq R$ of all admissible reload states (see Lemma 10). Note that R' is non-empty because there exists at least one T-visiting cycle. The idea now is to compute, for every $\hat{q} \in R'$, a polynomial-sized labelled graph $G_{\hat{q}}$ such that cycles in this graph correspond to T-visiting cycles in C that are initiated in $\hat{q}$ and that can be decomposed into segments of the form $CPath(\chi)$. An optimal T-visiting cycle is then found via a suitable analysis of the constructed graphs.

Formally, for a given $\hat{q} \in R'$ we construct a labelled graph $G_{\hat{q}} = (V, \mapsto, L, \ell)$, where $L \subset \mathbb{N}_0^2$, and where:

- $V = W \times \{0, \ldots, |S|\}$, where $W = R' \cup \{CTuple(\chi) \mid \chi \in X\}$.
- For every $0 \leq i < |S|$, every pair of states $r, t \in R'$ such that $r \neq \hat{q}$, and every characteristic $\chi \in X_{r,t}$ there is an edge $((r, i), (CPath(\chi), i))$ labelled by $(c(CPath(\chi)), len(CPath(\chi)))$ and an edge $((CPath(\chi), i), (t, i+1))$ labelled by $(0, 0)$.
- For every state $t \in R'$ and every characteristic $\chi \in X^1_{\hat{q},t}$ there is an edge $((\hat{q}, 0), (CPath(\chi), 0))$ labelled by $(c(CPath(\chi)), len(CPath(\chi)))$ and an edge $((CPath(\chi), 0), (t, 1))$ labelled by $(0, 0)$.
- For every $1 \leq i \leq |S|$ there is an edge $((\hat{q}, i), (\hat{q}, 0))$ labelled by $(0, 0)$.
- There are no other edges.

The labelling function of $G_{\hat{q}}$ can be computed in polynomial time, because given a characteristic χ, we can compute $CPath(\chi) = (\gamma, \gamma', \delta, k)$ using Lemma 18. Then, $len(CPath(\chi)) = len(\gamma) + len(\gamma') + k \cdot len(\delta)$, and similarly for $c(CPath(\chi))$. Note that every cycle in $G_{\hat{q}}$ contains the vertex $(\hat{q}, 0)$. Also note that some of the constructed graphs $G_{\hat{q}}$ may not have a cycle (the out-degree of $(\hat{q}, 0)$ may be equal to 0), but later we show that at least one of them does.

The *ratio* of a cycle $\hat{\beta} = v_0 \overset{(c_0,d_0)}{\mapsto} v_1 \overset{(c_1,d_1)}{\mapsto} v_2 \cdots \overset{(c_{h-1},d_{h-1})}{\mapsto} v_h$ in $G_{\hat{q}}$ is the value $rat(\hat{\beta}) = (c_0 + c_1 + \cdots + c_{h-1})/(d_0 + d_1 + \cdots d_{h-1})$. For every $\hat{q} \in R'$, our algorithm finds a simple cycle $\hat{\beta}_{\hat{q}}$ of minimal ratio among all cycles in $G_{\hat{q}}$. This is done using a polynomial-time algorithm for a well-studied problem of *minimum cycle ratio* (see, e.g., [15, 16]). The algorithm then picks $\hat{r} \in R'$ such that the ratio of $\hat{\beta}_{\hat{r}}$ is minimal. Clearly, $\hat{\beta}_{\hat{r}}$ has an even length and every second vertex is a 4-tuple of the form $CTuple(\chi)$ for some characteristic χ. Since all cycles in $\hat{r}$ go through $(\hat{r}, 0)$, we may assume that $\hat{\beta}_{\hat{r}}$ is initiated in this vertex. Let $CTuple(\hat{\chi}_0), CTuple(\hat{\chi}_1), \ldots, CTuple(\hat{\chi}_g)$ be the sequence of these 4-tuples, in the order they appear in $\hat{\beta}_{\hat{r}}$. From the construction of $G_{\hat{r}}$ it follows that $\beta = CPath(\hat{\chi}_0) \cdot CPath(\hat{\chi}_1) \cdots CPath(\hat{\chi}_g)$ is a reload-simple cycle initiated in an admissible state $\hat{r}$ containing a state of T (since χ_0 has the last component equal to 1), i.e., β is a T-visiting cycle. Moreover, $MP(\beta)$ is clearly equal to the ratio of $\hat{\beta}_{\hat{r}}$. Using the computed sequence of tuples $CTuple(\hat{\chi}_0), CTuple(\hat{\chi}_1), \ldots, CTuple(\hat{\chi}_g)$, the algorithm constructs the desired counting controller κ such that $run(\kappa, \beta(0)) = \beta^\omega$ (see also the discussion after Lemma 19). It is easy to check that $rat(\hat{\beta}_{\hat{r}}) = MC(\hat{\beta}_{\hat{r}})$ is equal to the mean cost of an optimal T-visiting cycle, i.e., the algorithm is correct.

3.3 Proof of Theorem 7

For the rest of this section we fix a consumption system $C = (S, \to, c, R, F)$ and an initial state $s \in S$. Intuitively, the controller can approach the limit value by interleaving a large number of iterations of some "cheap" cycle with visits to an accepting state. This motivates the following definitions of *safe* and *strongly safe* cycles. Intuitively, a cycle is safe if, assuming unbounded battery capacity, the controller can iterate the cycle for an arbitrary number of times and interleave these iterations with visits to an accepting state. A cycle is strongly safe if the same behaviour is achievable for some finite (though possibly large) capacity.

Formally, we say that two states $q, t \in S$ are *inter-reachable* if there is a path from q to t and a path from t to q (i.e., q, t are in the same strongly connected component of C). We say that a cycle β of length at most $|S|$ where $\beta(0)$ is reachable from s is *safe*, if one of the following conditions holds:

- $c(\beta) = 0$ and β contains an accepting state,
- $\beta(0)$ is inter-reachable with a reload state and an accepting state,

A cycle β reachable from s with $len(\beta) \leq |S|$ is *strongly safe*, if one of the following holds:

- $c(\beta) = 0$ and β contains an accepting state,
- $c(\beta) = 0$ and $\beta(0)$ is inter-reachable with a reload state and an accepting state,
- β contains a reload state and $\beta(0)$ is inter-reachable with an accepting state.

The following lemma characterizes the limit value of s.

Lemma 20. *$Val_C(s)$ is finite iff there is a safe cycle, in which case $Val_C(s) = \min\{MC(\beta) \mid \beta$ is a safe cycle$\}$. Further, there is a finite $cap \in \mathbb{N}_0$ such that $Val_C^{cap}(s) = Val_C(s)$ iff either $Val_C(s) = \infty$, or there is a strongly safe cycle $\hat{\beta}$ such that $MC(\hat{\beta}) = Val_C(s)$. In such a case $Val_C^{cap}(s) = Val_C(s)$ for every $cap \geq 3 \cdot |S| \cdot c_{\max}$, where $c_{\max}$ is the maximal cost of a transition in C.*

So, in order to compute the limit value and to decide whether it can be achieved with some finite capacity, we need to compute a safe and a strongly safe cycle of minimal mean cost.

Lemma 21. *The existence of a safe (or strongly safe) cycle is decidable in polynomial time. Further, if a safe (or strongly safe) cycle exists, then there is a safe (or strongly safe) cycle β computable in polynomial time such that $MC(\beta) \leq MC(\beta')$ for every safe (or strongly safe) cycle β'.*

Now we can prove the computation-related statements of Theorem 7.

To compute the limit value of s, we use the algorithm of Lemma 21 to compute a safe cycle β of minimal mean cost. If no such cycle exists, we have $Val_C(s) = \infty$, otherwise $Val_C(s) = MC(\beta)$. To decide whether $Val_C(s)$ can be achieved with some finite capacity, we again use the algorithm of Lemma 21 to compute a strongly safe cycle $\hat{\beta}$ of minimal mean cost. If such a cycle exists and $MC(\hat{\beta}) = MC(\beta)$, then $Val_C(s)$ can be achieved with some finite capacity, otherwise not. The correctness of this approach follows from Lemma 20.

It remains to bound the rate of convergence to the limit value in case when no finite capacity suffices to realize it. This is achieved in the following lemma.

Lemma 22. *Let $c_{\max}$ be the maximal cost of a transition in C. For every $cap > 4 \cdot |S| \cdot c_{\max}$ we have that*

$$Val_C^{cap}(s) - Val_C(s) \leq \frac{3 \cdot |S| \cdot c_{\max}}{cap - 4 \cdot |S| \cdot c_{\max}}.$$

4 Future work

We have shown that an optimal controller for a given consumption system always exists and can be efficiently computed. We have also exactly classified the structural complexity of optimal controllers and analyzed the limit values achievable by larger and larger battery capacity.

The concept of *cap*-bounded mean-payoff is natural and generic, and we believe it deserves a deeper study. Since mean-payoff has been widely studied (and applied) in the context of Markov decision processes, a natural question is whether our results can be extended to MDPs. Some of our methods are surely applicable, but the question appears challenging.

References

1. *Proceedings of FST&TCS 2010*, volume 8 of *Leibniz International Proceedings in Informatics*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2010.
2. *Proceedings of ICALP 2010, Part II*, volume 6199 of *Lecture Notes in Computer Science*. Springer, 2010.
3. N. Berger, N. Kapur, L.J. Schulman, and V. Vazirani. Solvency Games. In *Proceedings of FST&TCS 2008*, volume 2 of *Leibniz International Proceedings in Informatics*, pages 61–72. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2008.
4. P. Bouyer, U. Fahrenberg, K. Larsen, N. Markey, and J. Srba. Infinite Runs in Weighted Timed Automata with Energy Constraints. In *Proceedings of FORMATS 2008*, volume 5215 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2008.
5. T. Brázdil, V. Brožek, and K. Etessami. One-Counter Stochastic Games. In *Proceedings of FST&TCS 2010* [1], pages 108–119.
6. T. Brázdil, V. Brožek, K. Etessami, A. Kučera, and D. Wojtczak. One-Counter Markov Decision Processes. In *Proceedings of SODA 2010*, pages 863–874. SIAM, 2010.
7. T. Brázdil, K. Chatterjee, A. Kučera, and P. Novotný. Efficient Controller Synthesis for Consumption Games with Multiple Resource Types. In *Proceedings of CAV 2012*, volume 7358 of *Lecture Notes in Computer Science*, pages 23–38. Springer, 2012.
8. T. Brázdil, P. Jančar, and A. Kučera. Reachability Games on Extended Vector Addition Systems with States. In *Proceedings of ICALP 2010, Part II* [2], pages 478–489.
9. T. Brázdil, A. Kučera, P. Novotný, and D. Wojtczak. Minimizing Expected Termination Time in One-Counter Markov Decision Processes. In *Proceedings of ICALP 2012, Part II*, volume 7392 of *Lecture Notes in Computer Science*, pages 141–152. Springer, 2012.
10. K. Chatterjee and L. Doyen. Energy Parity Games. In *Proceedings of ICALP 2010, Part II* [2], pages 599–610.
11. K. Chatterjee and L. Doyen. Energy and Mean-Payoff Parity Markov Decision Processes. In *Proceedings of MFCS 2011*, volume 6907 of *Lecture Notes in Computer Science*, pages 206–218. Springer, 2011.
12. K. Chatterjee, L. Doyen, T. Henzinger, and J.-F. Raskin. Generalized Mean-payoff and Energy Games. In *Proceedings of FST&TCS 2010* [1], pages 505–516.
13. K. Chatterjee, M. Henzinger, S. Krinninger, and D. Nanongkai. Polynomial-Time Algorithms for Energy Games with Special Weight Structures. In *Proceedings of ESA 2012*, volume 7501 of *Lecture Notes in Computer Science*, pages 301–312. Springer, 2012.
14. K. Chatterjee, T. Henzinger, and M. Jurdziński. Mean-Payoff Parity Games. In *Proceedings of LICS 2005*, pages 178–187. IEEE Computer Society Press, 2005.
15. B. Dantzig, W. Blattner, and M. R. Rao. Finding a cycle in a graph with minimum cost to times ratio with applications to a ship routing problem. In P. Rosenstiehl, editor, *Theory of Graphs*, pages 77–84. Gordon and Breach, 1967.
16. A. Dasdan, S.S. Irani, and R.K. Gupta. Efficient algorithms for optimum cycle mean and optimum cost to time ratio problems. In *Design Automation Conference, 1999. Proceedings. 36th*, pages 37–42, 1999.
17. U. Fahrenberg, L. Juhl, K. Larsen, and J. Srba. Energy Games in Multiweighted Automata. In *Proceedings of the 8th International Colloquium on Theoretical Aspects of Computing (ICTAC'11)*, volume 6916 of *Lecture Notes in Computer Science*, pages 95–115. Springer, 2011.
18. A. Kučera. Playing Games with Counter Automata. In *Reachability Problems*, volume 7550 of *Lecture Notes in Computer Science*, pages 29–41. Springer, 2012.

Technical Appendix

A Non-trivial Behaviour of Optimal Controllers

Consider the consumption system C on Figure 2, where $R = \{s\}$ and $F = S$.

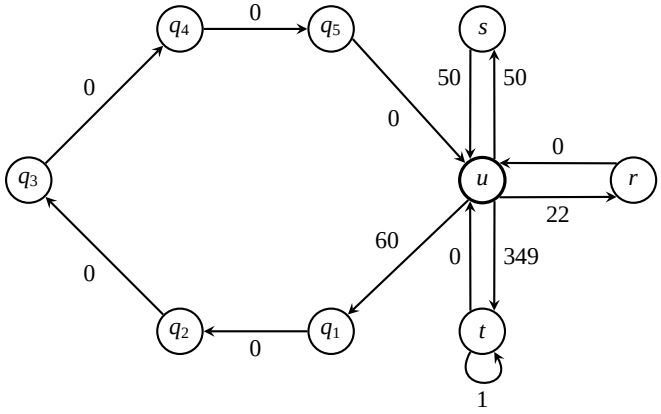

Fig. 2: A consumption system with a non-trivial behaviour of an optimal controller.

Denote by δ_1 the cycle $uq_1q_2q_3q_4q_5u$, and by δ_2 the cycle uru. Let s be the initial state and let $cap = 450$. Clearly, a controller that visits t cannot be optimal, because of the enormous mean cost of the cycle utu. Moreover it does not make sense to iterate the cycle sus, since its mean cost is much larger than the mean cost of both δ_1 and δ_2. Thus, an optimal controller goes from s to u, then iterates δ_1 for $A \in \{0, 1, \ldots, 5\}$ times, then possibly iterates δ_2 for $B \in \{0, 1, \ldots, 15\}$ times, and then returns to s. An easy computation shows that the optimal behaviour is achieved for $A = 5$ and $B = 2$, and the resulting mean cost is equal to $37/3$. This shows that the optimal controller generally has to iterate more than one simple cycle between two visits of a reload state (the controller from example on Figure 1 iterated only 1 simple cycle before returning to the reload state). Also, note that the cycle tt, which has the minimal mean cost among all simple cycles in C, is not traversed by the optimal controller at all.

Now let us again consider the example on Figure 1. Note that for any capacity $cap \geq 10$ we have $Val_C^{cap}(s) = \frac{cap}{2+2\cdot(cap-10)} = \frac{1}{2} \cdot \frac{cap}{cap-9}$. It is then clear that $Val_C(s) = 1/2$ and that this limit value cannot be achieved for any finite capacity.

Finally, consider the consumption system C on Figure 3. For every capacity cap we have $Val_C^{cap}(s) = \infty$, since every cap-bounded path C must have an infinite suffix u^ω and thus it cannot be accepting. Thus, the limit value $Val_C(s)$ is also infinite. However, if we treat the system as a one-player mean-payoff Büchi game (see, e.g., [14]), then the optimal value in s is $1/2$.

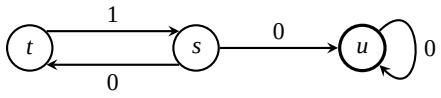

Fig. 3: Limit value is not equal to the Büchi mean-payoff value. Here $R = \{u\}$ and $F = \{t\}$.

B Proofs for Auxiliary Algorithms in Section 3.1

In this section we provide the proofs for auxiliary algorithms from Section 3.1, i.e., the proof of Lemmas 8,9,10 and 12. The problems solved by these algorithms (*cap*-reachability, existence of an acceptance witness, etc.) are variants of standard graph-theoretic problems. Our choice of algorithms is motivated by our intention to achieve as straightforward correctness proofs and proofs of polynomial running time as possible. It is not hard to see that the complexity of these algorithms (which does not dominate the overall complexity of the algorithm from Theorem 6) can be improved.

Note that the lemmas are not proved in the order in which they appear in the main text. We first prove Lemma 9, then Lemma 10, then Lemma 8 and finally Lemma 12. This is because some algorithms use as a sub-procedure an algorithm which, in the main text, appears, later than the algorithm in which this sub-procedure is used. We chose to mention the algorithms in the main text in a different order to make the flow of ideas in the main text more natural.

B.1 A Proof of Lemma 9

We denote by $ec(\alpha)$ the end cost of a finite path α. First we prove the following lemma.

Lemma 23. *If a state t is cap-reachable from s, then there is a cap-bounded path α of length at most $|S|^2$ such that $ec(\alpha) = \min\{ec(\alpha') \mid \alpha' \text{ is a cap-bounded path from } s \text{ to } t\}$.*

Proof. Let t be *cap*-reachable from s. Since the end costs are natural numbers, the value $\mathit{min\text{-}ec} = \min\{ec(\alpha') \mid \alpha' \text{ is a } \mathit{cap}\text{-bounded path from } s \text{ to } t\}$ exists. Let α be a *cap*-bounded path of minimal length among the *cap*-bounded paths from s to t that have $ec(\alpha) = \mathit{min\text{-}ec}$. Assume, for the sake of contradiction, that $\mathit{len}(\alpha) > |S|^2$. Two cases may arise: either at most $|R|$ reload states appear on α and then $\alpha = \gamma \cdot \delta \cdot \gamma'$, where either δ is a cycle not containing a reload state, or $\delta(0) = \mathit{last}(\delta) \in R$. In both cases clearly, $\gamma \cdot \gamma'$ is a *cap*-bounded path from s to t with $ec(\gamma \cdot \gamma') = ec(\alpha)$, a contradiction with the choice of α. The second case is that there are more than $|R|$ occurrences of a reload state on α. Then $\alpha = \gamma \cdot \delta \cdot \gamma'$, where $\delta(0) = \mathit{last}(\delta) \in R$. As above, we get a contradiction with the choice of α. □

We now prove a slightly more general variant of Lemma 9.

Lemma 24. *There is a polynomial-time algorithm, which for a given state s, given capacity cap and every state $t \in S$ decides, whether t is cap-reachable from s. Moreover, for every t that is cap-reachable from s the algorithm computes a cap-bounded path α of length at most $|S|^2$ such that $ec(\alpha) = \min\{ec(\alpha') \mid \alpha' \text{ is a cap-bounded path from } s \text{ to } t\}$.*

Proof. Denote $\textit{min-ec} = \min\{ec(\alpha') \mid \alpha' \text{ is a } \textit{cap}\text{-bounded path from } s \text{ to } t\}$. Moreover, for any $t \in S$ and any $i \in \mathbb{N}_0$ we denote

$$\textit{min-ec}^i(t) = \min\{ec(\alpha) \mid \alpha \text{ is a } \textit{cap}\text{-bounded path from } s \text{ to } t \text{ of length } i\}.$$

From Lemma 23 it follows that $\textit{min-ec}(t) = \min_{0 \le i \le |S|^2} \textit{min-ec}^i(t)$.

Now consider an operation $\oplus_{cap}$ on $\mathbb{N}_0 \cup \{\infty\}$ such that for any $a \oplus_{cap} b = a + b$ if $a + b \le \textit{cap}$ and ∞ otherwise (we use a standard convention that $\infty + x = x + \infty = \infty$ for any $x \in \mathbb{N}_0 \cup \{\infty\}$). Clearly $\textit{min-ec}^0(t)$ is equal to 0 if $t = s$ and equal to ∞ otherwise. For $i > 0$ a straightforward induction reveals that

$$\textit{min-ec}^i(t) = \begin{cases} \min\limits_{q \xrightarrow{a} t} (\textit{min-ec}^{i-1}(q) \oplus_{cap} a) & \text{if } t \notin R \\ 0 & \text{if } t \in R \text{ and } \min\limits_{q \xrightarrow{a} t} (\textit{min-ec}^{i-1}(q) \oplus_{cap} a) < \infty \\ \infty & \text{otherwise .} \end{cases}$$

Using these equations we can iteratively compute $\textit{min-ec}^0(q), \textit{min-ec}^1(q), \ldots, \textit{min-ec}^{|S|^2}(q)$ for all states q in polynomial time. Now by Lemma 23 we have for any state t that $\textit{min-ec}(t) = \min_{0 \le i \le |S|^2} \textit{min-ec}^i(t)$, and clearly t is *cap*-reachable from s iff $\textit{min-ec}(t) < \infty$. Moreover, let $j_t \le |S|^2$ be such that $\textit{min-ec}^{j_t}(t) = \textit{min-ec}(t) < \infty$. Then from the knowledge of $\textit{min-ec}^0, \textit{min-ec}^1, \ldots, \textit{min-ec}^j$ we can construct a finite *cap*-bounded path $\alpha = q_0 q_1 \ldots q_{j_t}$ with from s to t by putting $q_{j_t} = t$ and for every $j < j_t$ defining q_j the state that caused the $\textit{min-ec}^{j+1}(q_{j+1})$ to be set to its final value. I.e., if $q_{j+1} \notin R$, then q_j is such that $q_j \xrightarrow{a} q_{j+1}$ and $\textit{min-ec}^j(q_j) \oplus_{cap} a = \textit{min-ec}^{j+1}(q_{j+1})$, otherwise q_j is such that $q_j \xrightarrow{a} q_{j+1}$ and $\textit{min-ec}^j(q_j) \oplus_{cap} a < \infty$. The correctness of this approach is immediate. □

B.2 A Proof of Lemma 10

Before we prove Lemma 10, we prove the following simple lemma.

Lemma 25. *For a given state s it is decidable in polynomial time whether there is a cycle δ of zero cost containing s, and if such a δ exists, we can compute in polynomial time a simple cycle with this property. Moreover, for a given set of states $T \subseteq S$ it is decidable in polynomial time, whether there is a cycle of zero cost containing s and a state from T, and if such a cycle exists, we can compute such a cycle of length at most $2|S|$ in polynomial time.*

Proof. Clearly if there is a cycle of zero cost on s, there is also a simple cycle of zero cost on s. Such a cycle can be find using a simple modification of e.g., the breadth-first search algorithm – for every t such that $s \xrightarrow{0} t$ we try to find a path from t to s while ignoring the transitions of positive cost. Similarly, if there is a cycle of zero cost on s that contains a state from T, then the shortest such cycle has length at most $2|S|$ (we need to get from s to a state $t \in T$ and back via transitions of zero cost, each of these two parts requiring at most $|S|$ transitions). Again, such a cycle can be find using a suitable search algorithm: first, we compute a set $T' \subseteq T$ of states that are reachable via transitions of

zero cost from a state q s.t. $s \xrightarrow{0} q$, and then, for every $t \in T'$ we try to find (possibly empty) path from t to s. Clearly, both tasks can be implemented using a simple graph search algorithm. □

The following lemma will be also useful.

Lemma 26. *Let $r \in R$ be a reload state and $T \subseteq S$ a set of states. It is decidable in polynomial time whether there is a cap-bounded cycle δ that is initiated in r and that contains a state from T. If the answer is yes, one can compute (in polynomial time) such a cycle δ of length at most $3|S|^2$.*

Proof. Recall that we denote by $ec(\alpha)$ the end cost of a finite path α. Denote by $\mathit{min\text{-}ec}_r(t)$ the value $\min\{ec(\alpha) \mid \alpha$ is a *cap*-bounded path from r to $t\}$. We claim that a *cap*-bounded δ initiated in r and containing a state from T exists if and only if there is a $t \in T$ and $q \in R$ (possibly $q = r$) such that:

- There is a *cap*-bounded path γ_1 from r to t, and
- there is a $(\mathit{cap} - \mathit{min\text{-}ec}_r(t))$-bounded path γ_2 from t to q, and
- there is a *cap*-bounded path γ_3 from q to r.

The "if" direction can be proved as follows: if there are the aforementioned paths, then there is in particular a *cap*-bounded path γ_1 from r to t such that $ec(\gamma_1) = \mathit{min\text{-}ec}_r(t)$ (see Lemma 24). Then $\delta = \gamma_1 \cdot \gamma_2 \cdot \gamma_3$ is the required *cap*-bounded cycle containing a state from T.

Now consider the "only if" direction in the equivalence. Let $i \in \mathbb{N}_0$ be the smallest number such that $\delta(i) \in T$ and $j > i$ be the smallest number such that $\delta(j) \in R$ (such i, j must exist, since δ is a cycle). Put $\gamma_1 = \delta_{\leq i}$, $\gamma_2 = \delta(i) \ldots \delta(j)$ and $\gamma_3 = \delta(j) \ldots \delta(\mathit{len}(\delta))$. Clearly γ_3 and γ_1 are *cap*-bounded and $c(\gamma_2) \leq \mathit{cap} - ec(\gamma_1) \leq \mathit{cap} - \mathit{min\text{-}ec}_r(\delta(i))$, which proves the "only if" direction.

So in order to decide whether a desired cycle δ exists (and compute it if it does) it suffices to make three calls of the algorithm from Lemma 24 for every pair of states $t \in T$, $q \in R$. For every such pair we first use that algorithm to compute a *cap* bounded path γ_1 from r to t of minimal end cost (and thus also compute $\mathit{min\text{-}ec}_r(t)$). Then we use the algorithm to find a $(\mathit{cap} - \mathit{min\text{-}ec}_r(t))$-bounded path γ_2 from t to q, and a *cap*-bounded path γ_3 from q to r. If we find all these paths, we return $\gamma_1 \cdot \gamma_2 \cdot \gamma_3$ as the desired cycle δ (from Lemma 24 it follows that $\mathit{len}(\delta) \leq 3|S|^2$). If some of these paths does not exist, we move on to the next pair. If the algorithm fails for all pairs, the desired cycle δ does not exist. The correctness of the algorithm and its polynomial complexity are immediate.

□

Lemma 10. *The problem whether a given $q \in S$ is admissible is decidable in polynomial time. Further, if q is admissible, then there are finite paths α, γ computable in polynomial time such that $\alpha \cdot \gamma^\omega$ is a cap-bounded run initiated in s and γ is an admissibility witness for q of length at most $6|S|^2$.*

Proof. First we prove that a state $q \in S$ is admissible if and only if q is *cap*-reachable from the initial state s and one of the following conditions holds.

1. There is a cycle δ of zero cost that is initiated in q and that contains an accepting state. In this case, δ is an admissibility witness for q.
2. There is a reload state $r \in R$ and *cap*-bounded cycles θ, δ, both initiated in r, such that δ contains an accepting state and $\theta = \theta_1 \cdot \theta_2$, where $\theta_1(0) = last(\theta_2) = r$ and $last(\theta_1) = \theta_2(0) = q$. In such a case $\theta_2 \cdot \delta \cdot \theta_1$ is an admissibility witness for q.

The "if" direction is immediate, so let us consider the "only if" direction. Suppose that q is admissible, then by definition there is a *cap*-bounded run initiated in s of the form $\alpha \cdot \beta^\omega$, where β is a cycle initiated in q which contains an accepting state. In particular q is *cap* reachable from s. Now if $c(\beta) = 0$, then the case 1. above holds. So suppose that $c(\beta) \neq 0$. Then β must contain not only an accepting state, but also a reload state. Let $i \in \mathbb{N}_0$ be such that $\beta(i) \in R$, respectively. Then the paths $\theta_1 = \beta(i)\beta(i+1)\ldots\beta(len(\beta))$, $\theta_2 = \beta_{\leq i}$, and $\delta = \beta(i)\beta(i+1)\ldots\beta(len(\beta)) \cdot \beta_{\leq_i}$ have the properties stated in case 2.

So to test whether q is admissible, we have to test whether q is *cap*-reachable from s and whether 1. or 2. holds. To test the *cap*-reachability we use the polynomial algorithm of Lemma 24, which also finds the required *cap*-bounded path from s to q. To test whether 1. holds, we use the polynomial algorithm from Lemma 25. If this algorithm finds a cycle of zero cost initiated in q and containing an accepting state (by Lemma 25, the cycle returned by the algorithm has length at most $2|S|$), we can immediately output it as an admissibility witness for q. To test whether 2. holds, we use $|R|$ times the polynomial algorithm of Lemma 26: For every state $r \in R$ we test whether there are *cap*-bounded cycles θ, δ initiated in r such that θ contains q and δ contains a state from F. If the algorithm finds such cycles for some $r \in R$ (by Lemma 26 each of them will have length at most $3|S|^2$), we can use them to easily construct an admissibility witness for q of length at most $6|S|^2$ as indicated in 2. The correctness of the algorithm and its polynomial complexity are immediate. □

B.3 A Proof of Lemma 8

Lemma 8. *Let $t \in S$. The problem whether $Val_C^{cap}(t) = \infty$ is decidable in polynomial time.*

Proof. Let $t \in S$ be an arbitrary. In the following we treat t as the initial state of the system. In particular, the notion of admissibility is adapted to this choice of initial state: a state q is admissible if there is a *cap*-bounded path of the form $\alpha \cdot \gamma^\omega$ with $\alpha(0) = t$, $\gamma(0) = last(\alpha) = q$, and γ containing an accepting state.

First note that $Val_C^{cap}(t) < \infty$ if and only if there is at least one admissible state q The "if" direction is immediate, so let us consider the "only if" direction. If $Val_C^{cap}(t) < \infty$, then there is a *cap*-bounded accepting run ϱ initiated in t. We consider two cases. Either there are only finitely many transitions of a positive cost on ϱ. Then there is a simple cycle δ of zero cost containing an accepting state (since some accepting state has infinitely many occurrences on ϱ) initiated in some state $\delta(0)$ that is *cap*-reachable from t. Clearly $\alpha \cdot \delta^\omega$, where α is a *cap*-bounded path from t to q, is a *cap*-bounded accepting run, so $\delta(0)$ is admissible. The second case is that there are infinitely many transitions of positive cost on ϱ, in which case ϱ contains infinitely many occurrences of both a reload state and of an accepting state. Let r be a reload state appearing infinitely often on ϱ.

Then there is a *cap*-bounded cycle δ initiated in r and containing an accepting state. For any *cap*-bounded path α from t to r (at least one exists due to the existence of ϱ) the run $\alpha \cdot \delta^\omega$ is a *cap*-bounded accepting run, showing that r is admissible.

So to decide whether $Val_C^{cap}(t) = \infty$, it suffices to use the polynomial-time algorithm of Lemma 24 to compute the set of states that are *cap*-reachable from t, and for every such state decide, whether it is admissible, using the polynomial-time algorithm from Lemma 10. The correctness and the polynomial running time of this procedure are immediate. □

B.4 A Proof of Lemma 12

Lemma 12. *The existence of a zero-cost cycle is decidable in polynomial time, and an example of a zero-cost cycle β (if it exists) is computable in polynomial time. The same holds for zero-cost cycles containing an accepting state.*

Proof. Note that a zero-cost cycle, or zero-cost cycle containing an accepting state, is simply a simple cycle of zero cost initiated in an admissible state, or a simple cycle of zero cost containing an accepting state that is initiated in an admissible state, respectively.

So to decide whether there is a zero-cost cycle, it suffices to compute the set $A \subseteq S$ of all admissible states using the polynomial-time algorithm of Lemma 10, and then for every $q \in A$ try to find a simple cycle of zero cost initiated in q using the polynomial-time algorithm from Lemma 25. If we find such a cycle we can output it as β, otherwise we conclude that there is no such cycle.

To decide whether there is a zero-cost cycle containing an accepting state, it suffices, for every $q \in A$, to use the polynomial-time algorithm of Lemma 25 to find a cycle of zero cost initiated in q containing an accepting state. Note that if the algorithm finds such a cycle δ for some $q \in A$, this cycle does not have to be simple. However, if it is not simple, then there is a simple cycle θ of zero cost such that $\delta = \gamma \cdot \theta \cdot \gamma'$ for some finite paths γ, γ' and θ contains an accepting state. Moreover, $\delta(0)$ is an admissible state, since $\delta \cdot \gamma' \cdot \gamma$ is an admissibility witness for $\delta(0)$. So θ is a zero-cost cycle and we can return it as the desired cycle β (note that θ can be easily computed once δ is computed). If the algorithm of Lemma 25 fails to find a cycle of zero cost with an accepting state for every $q \in A$, we conclude that there is no such cycle. □

C Auxiliary Results

This section contains some auxiliary algorithms that are not mentioned in the main text and that will be useful in later proofs.

Lemma 27. *Let $C = (S, \to, c, R, F)$ be a consumption system. There is an algorithm MinPath$(s_1, s_2, m, Avoid)$ which for a given pair of states $s_1, s_2 \in S$, given number $m \in \mathbb{N}_0$ and a given set $Avoid \subseteq S$ decides, whether there is a path α satisfying the following conditions:*

- $\alpha(0) = s_1$, $last(\alpha) = s_2$, $len(\alpha) = m$, *and*

– *for all* $0 < i < len(\alpha)$ *it holds* $\alpha(i) \notin Avoid$.

If there is such a path, the algorithm computes a path α *of minimal cost among all paths satisfying the above conditions. The algorithm runs in time polynomial in* $\|C\|$ *and* m.

Proof. The algorithm constructs a labelled graph $G = (V, \mapsto, L, \ell)$, where $L \subset \mathbb{N}_0$, is defined as follows:

– $V = (S \setminus Avoid) \times \{1, 2, \ldots, m-1\} \cup \{(s_1, 0), (s_2, m)\}$.
– There is an edge $((s, i), (s', j))$ in G if and only if $j = i + 1$ and $s \xrightarrow{a} s'$ is a transition in C. In such a case $\ell((s, i), (s', j)) = a$.

Then the algorithm finds a path of minimal cost from $(s_1, 0)$ to (s_2, m) (or decides that such a path does not exist) using, e.g., the algorithm for computing shortest paths in directed acyclic graphs. The procedure then returns the corresponding path in C (it suffices to discard the second components from the computed path in G). The correctness of the procedure and its complexity analysis are straightforward.

Lemma 28. *Let* $C = (S, \to, c, R, F)$ *be a consumption system. There is an algorithm* $MinPathReach(s_1, s_2, m, Avoid, Reach)$ *which for a given pair of states* $s_1, s_2 \in S$*, given number* $m \in \mathbb{N}_0$ *and given sets* $Reach, Avoid \subseteq S$ *decides, whether there is a path* α *satisfying the following conditions:*

– $\alpha(0) = s_1$, $last(\alpha) = s_2$, $len(\alpha) = m$, *and*
– *for all* $0 < i < len(\alpha)$ *it holds* $\alpha(i) \notin Avoid$, *and*
– *there is* $0 \leq j \leq len(\alpha)$ *such that* $\alpha(j) \in Reach$.

If there is such a path, the algorithm computes a path α *of minimal cost among all paths satisfying the above conditions. The algorithm runs in time polynomial in* $\|C\|$ *and* m.

Proof. If $s_1 \in Reach$ or $s_2 \in Reach$, then we just call the algorithm $MinPath(s_1, s_2, m, Avoid)$ from Lemma 27.

Otherwise for every $q \in Reach$ and every $0 < i < m$ the algorithm constructs a labelled graph $G_{q,i} = (V, \mapsto, L, \ell)$, where $L \subset \mathbb{N}_0$, is defined as follows:

– $V = (S \setminus Avoid) \times \{1, 2, \ldots i-1, i+1, \ldots, m-1\} \cup \{(s_1, 0), (s_2, m), (q, i)\}$.
– There is an edge $((s, i), (s', j))$ in G if and only if $j = i + 1$ and $s \xrightarrow{a} s'$ is a transition in C. In such a case $\ell((s, i), (s', j)) = a$.

Then, for every $q \in Reach$ and every $0 < i < m$ the algorithm finds a path $\alpha_{q,i}$ of minimal cost from $(s_1, 0)$ to (s_2, m) in $G_{q,i}$ (or decides that such a path does not exist) using, again the algorithm for computing shortest paths in directed acyclic graphs. If $\alpha_{q,i}$ exists for at least for one pair q, i, the algorithm returns $\alpha_{q,i}$ of minimal cost (the minimum is taken among all $q \in Reach$, $0 < i < m$), otherwise the path satisfying the required conditions does not exist. Again, the correctness of the algorithm and its complexity are straightforward, since every path α satisfying the required conditions induces, in a natural way, a corresponding path α' of the same cost in $G_{q,j}$, where q, j are such that $q = \alpha(j) \in Reach$. Conversely, every α' in some $G_{q,i}$ induces a path α of the same cost in C that satisfies the required conditions. □

D Proofs of Section 3.2

D.1 A Proof of Lemma 16

Lemma 16. *If there is at least one compact segment with a given characteristic χ, then there is also an optimal compact segment for χ. Moreover, all compact segments optimal for a given characteristic have the same total cost and length.*

Proof. Fix a characteristic $\chi = (r, q, t, m, n, b)$. If there is at least one compact segment $\eta = \xi \cdot \theta^j \cdot \xi'$ having characteristic χ, there is also at least one segment of a characteristic $(r, q, t, m, 0, b)$ (namely $\xi \cdot \xi'$); and (provided that $n > 0$) at least one cycle of length n initiated in q which is either a segment or does not contain any reload state (namely θ). So there also is a segment $\gamma \cdot \gamma'$ and (provided that $n > 0$) a cycle δ satisfying the above conditions whose costs are minimal among all segments and cycles that satisfy these conditions, respectively. Let k be either 1 (if $n = 0$) or the maximal number such that $\gamma \cdot \delta^k \cdot \gamma'$ is a *cap*-bounded path (if $n > 0$ – then such a k must exist, because $c(\gamma \cdot \delta \cdot \gamma') \leq c(\eta)$). Clearly $\gamma \cdot \delta^k \cdot \gamma'$ is a compact segment optimal for χ.

For the second part, let $\eta = \xi \cdot \theta^j \cdot \xi'$ and $\eta' = \gamma \cdot \delta^k \cdot \gamma'$ be two segments optimal for the same characteristic $\chi = (r, q, t, m, n, b)$. If $n = 0$, then clearly $len(\eta) = len(\eta') = m$ and from the optimality of both segments we get the equality of their costs. Otherwise, by definition of optimal segments we have $c(\gamma \cdot \gamma') = c(\xi \cdot \xi')$ and $c(\theta) = c(\delta)$. To prove the lemma it suffices to show that $j = k$. Suppose that, e.g., $j < k$, the other case is symmetrical. Then $\xi \cdot \theta^{j+1} \cdot \xi'$ is a *cap*-bounded path (since its cost is at most the cost of η'), a contradiction with the fact that η has a characteristic. □

D.2 A Proof of Lemma 17

Lemma 17. *There is an optimal T-visiting cycle β whose every segment is a compact segment optimal for some characteristic.*

Proof. We say that a segment is *bad* if it is not a compact segment optimal for some characteristic χ. Given an optimal T-visiting cycle β containing $g > 0$ bad segments, we show how to construct an optimal T-visiting cycle β' containing $g - 1$ bad segments. Combined with the existence of at least one optimal T-visiting cycle (which follows from Lemma 13), this proves the lemma.

So let β be an optimal T-visiting cycle and η its bad segment (i.e., $\beta = \xi \cdot \eta \cdot \xi'$ for some finite paths ξ, ξ'). We denote $t = \eta(0)$ and $r = last(\eta)$. In the following we call every segment initiated in t and ending in r an r-t-segment. We also say that two paths are T-equivalent, if both of them contain a state from T or none of them does. We will construct an r-t-segment η' such that η' is not bad, η' is T-equivalent ot η and $MC(\beta) = MC(\beta')$, where $\beta' = \xi \cdot \eta' \cdot \xi'$. Then clearly β' is an optimal T-visiting cycle having at most $g - 1$ bad segments.

The construction proceeds in two steps. First we construct a compact r-t-segment $\hat{\eta}$ of cost and length equal to $c(\eta)$ and $len(\eta)$, respectively. Then we construct an r-t-segment η' with $MC(\xi \cdot \eta' \cdot \xi') = MC(\xi \cdot \hat{\eta} \cdot \xi') = MC(\beta)$ such that η' is a compact segment

with a characteristic, and we show that η' must be optimal for all of its characteristics. During the construction we ensure that $\hat{\eta}$ and η' are T-equivalent to η.

Note that from the optimality of β it follows that every r-t-segment of length equal to η which is T-equivalent to η must have a cost greater or equal to $c(\eta)$. We will often use this fact in the proof.

Constructing $\hat{\eta}$ from η: We employ a technique similar to the technique of decomposition into simple cycles. An *η-decomposition* is a sequence $dc = \alpha_0, \delta_0, k_0, \alpha_1, \delta_1, k_1, \ldots, \alpha_{h-1}, \delta_{h-1}, k_{h-1}, \alpha_h$ such that

- For every i the α_i is a finite path, δ_i is a simple cycle, and k_i is a positive integer, and
- $\alpha_0\delta_0^{k_0}\alpha_1\delta_1^{k_1}\cdots\alpha_{h-1}\delta_{h-1}^{k_{h-1}}\alpha_h$ is an r-t-segment that is T-equivalent to η and whose cost and length are equal to $c(\eta)$ and $len(\eta)$, respectively.

A *rank* of such an η-decomposition is the vector of natural numbers

$$rank(dc) \quad = \quad \Big(\sum_{i=0}^{h} len(\alpha_i),\ h,\ |\{i \mid k_i > |S|\}| \Big).$$

Now let $dc = \alpha_0, \delta_0, k_0, \alpha_1, \delta_1, k_1, \ldots, \alpha_{h-1}, \delta_{h-1}, k_{h-1}, \alpha_h$ be an η-decomposition with rank minimal w.r.t. the lexicographic ordering (such an η-decomposition exists, since ranks are vectors of natural numbers), and let $\hat{\eta} = \alpha_0\delta_0^{k_0}\alpha_1\delta_1^{k_1}\cdots\alpha_{h-1}\delta_{h-1}^{k_{h-1}}\alpha_h$ be the corresponding r-t-segment T-equivalent to η, whose length and cost are equal to $len(\eta)$ and $c(\eta)$, respectively. We claim that the following holds: for every $0 \leq i \leq h$ we have $len(\alpha_i) < |S|$, $h \leq 2 \cdot |S|$ and $|\{i \mid k_i > |S|\}| \leq 1$. From this it immediately follows that $\hat{\eta} = \gamma \cdot \delta^k \cdot \gamma'$, where δ is a cycle of length at most $|S|$, and γ, γ' are such that $len(\gamma \cdot \gamma') \leq 4|S|^3$. In particular, $\hat{\eta}$ is a compact segment (the fact that the paths γ, γ' are shorter than required for the compactness will be used in the second part of the proof).

First let us assume, for the sake of contradiction, that for some $0 \leq i \leq h$ it holds $len(\alpha_i) \geq |S|$. Then $\alpha_i = \alpha'\delta'\alpha''$ for some simple cycle δ' of positive length and some (possibly empty) finite paths α', α''. Then the sequence $dc' = \alpha_0, \delta_0, k_0, \ldots, \delta_{i-1}, k_{i-1}, \alpha', \delta', 1, \alpha'', \delta_i, k_i, \ldots \alpha_h$ is an η-decomposition such that $rank(dc')[1] = rank(dc)[1] - len(\delta') < rank(dc)[1]$, a contradiction with the choice of dc.

Now let us assume that $h > 2 \cdot |S|$. Then there are $0 \leq i < j < h$ such that $len(\delta_i) = len(\delta_j)$ and δ_i is T-equivalent to δ_j. It must be the case that $c(\delta_i) = c(\delta_j)$, otherwise, if e.g. $c(\delta_i) < c(\delta_j)$, then $\alpha_0\delta_0^{k_0}\cdots\delta_{i-1}^{k_{i-1}}\alpha_i\alpha_{i+1}\cdots\alpha_j\delta_j^{k_j+k_i}\alpha_{j+1}\cdots\alpha_h$ would be an r-t-segment T-equivalent to θ whose length equals $len(\eta)$ and whose cost is smaller than $c(\eta)$, a contradiction with the optimality of β. So the sequence $dc' = \alpha_0, \ldots, \delta_{i-1}, k_{i-1}, (\alpha_i \cdot \alpha_{i+1}), \delta_{i+1}, \ldots, \alpha_j, \delta_j, k_j + k_i, \alpha_{j+1}, \ldots, \alpha_h$ is an η-decomposition with $rank(dc')[1] = rank(dc)[1]$ and $rank(dc')[2] = rank(dc)[2] - 1$, a contradiction with the choice of dc.

Finally, let us assume that there are $0 \leq i < j < h$ such that $k_i > |S|$ and $k_j > |S|$. We distinguish three cases: $MC(\delta_i) > MC(\delta_j)$, $MC(\delta_i) < MC(\delta_j)$ and $MC(\delta_i) = MC(\delta_j)$.

First assume that $MC(\delta_i) > MC(\delta_j)$ and let a be the greatest natural number such that $|S| \geq k_i - a \cdot len(\delta_j) \geq 1$ (clearly $a \geq 1$). We have

$$MC(\delta_i) - MC(\delta_j) = \frac{c(\delta_i) \cdot len(\delta_j) - c(\delta_j) \cdot len(\delta_i)}{len(\delta_i) \cdot len(\delta_j)} > 0,$$

from which it follows that $c(\delta_i) \cdot len(\delta_j) - c(\delta_j) \cdot len(\delta_i) > 0$. Now consider the path $\pi = \alpha_0 \delta_0^{k_0} \cdots \alpha_i \delta_i^{k_i - a \cdot len(\delta_j)} \alpha_{i+1} \cdots \alpha_j \delta_j^{k_j + a \cdot len(\delta_i)} \alpha_{j+1} \cdots \alpha_h$. Clearly $len(\pi) = len(\hat{\eta}) = len(\eta)$ and π is an r-t-segment T-equivalent to η. Moreover,

$$c(\pi) = c(\hat{\eta}) - a \cdot (c(\delta_i) \cdot len(\delta_j) - c(\delta_j) \cdot len(\delta_i)) < c(\hat{\eta}) = c(\eta),$$

a contradiction with the optimality of β.

The case $MC(\delta_i) < MC(\delta_j)$ is handled symmetrically, so it remains to consider the case $MC(\delta_i) = MC(\delta_j)$. In this case we clearly have $c(\delta_i) \cdot len(\delta_j) - c(\delta_j) \cdot len(\delta_i) = 0$ and thus the aforementioned path π is an r-t-segment T-equivalent to η such that not only $len(\pi) = len(\eta)$, but also $c(\pi) = c(\eta)$. It follows that the corresponding sequence $dc' = \alpha_0, \delta_0, k_0, \cdots \alpha_i, \delta_i, k_i - a \cdot len(\delta_j), \alpha_{i+1}, \cdots, \alpha_j, \delta_j, k_j + a \cdot len(\delta_i), \alpha_{j+1}, \cdots, \alpha_h$ is an η-decomposition such that $rank(dc')[1] = rank(dc)[1]$, $rank(dc')[2] = rank(dc)[2]$ and $rank(dc')[3] < rank(dc)[3]$, a contradiction with the choice of dc.

Constructing η' from $\hat{\eta}$: We now have an r-t-segment $\hat{\eta}$ such that $\hat{\eta}$ is T-equivalent to η, $len(\hat{\eta}) = len(\eta)$ and $c(\hat{\eta}) = c(\eta)$ (and thus also $MC(\hat{\eta}) = MC(\eta)$), and moreover $\hat{\eta} = \gamma \cdot \delta^k \cdot \gamma'$ for some finite paths γ, γ' of combined length at most $4|S|^3$ and δ either a single vertex or a simple cycle. The compact segment $\hat{\eta}$ may not have a characteristic for three reasons:

- $c(\delta) = 0$;
- δ contains a state from T and $\gamma \cdot \gamma'$ not;
- δ is a cycle and $\gamma \cdot \delta^{k+1} \cdot \gamma'$ is also a *cap*-bounded path.

The first two cases actually cannot happen. Indeed, if $c(\delta) = 0$, then δ is a zero-cost cycle (recall that every state on a T-visiting cycle is admissible, and $\delta(0)$ lies on a T-visiting cycle $\xi \cdot \hat{\eta} \cdot \xi'$), a contradiction with the assumptions of Proposition 15. In the second case clearly $k \geq 1$ and $\hat{\eta} = \gamma \cdot \delta \cdot \delta^{k-1} \cdot \gamma'$ is a compact segment of a characteristic $(r, \delta(0), t, m, len(\delta), 1)$, where $m = len(\gamma \cdot \delta \cdot \gamma') \leq 4|S|^3 + |S| \leq 5|S|^3$, a contradiction with η' not having a characteristic.

Now suppose that the first two cases do not occur and the third does. Then we have $k \geq 2$, since otherwise we would have $len(\hat{\eta}) \leq 4|S|^3 + |S| \leq 5|S|^3$ and $\hat{\eta}$ would have a characteristic $(r, r, t, len(\hat{\eta}), 0, 0)$. Now let $z \geq 1$ be the maximal number such that $\eta' = \gamma \cdot \delta^z \cdot \gamma'$ is a *cap*-bounded path. Clearly, η' is a compact r-t-segment T-equivalent to $\hat{\eta}$ with a characteristic. We need to show that $MC(\xi \cdot \eta' \cdot \xi') = MC(\xi \cdot \hat{\eta} \cdot \xi')$. From the optimality of β it follows that it suffices to show $MC(\xi \cdot \eta' \cdot \xi') \leq MC(\xi \cdot \hat{\eta} \cdot \xi')$.

Assume, for the sake of contradiction, that $MC(\xi \cdot \eta' \cdot \xi') > MC(\xi \cdot \hat{\eta} \cdot \xi')$. Denote $C = c(\xi \cdot \gamma \cdot \gamma' \cdot \xi')$ and $D = len(\xi \cdot \gamma \cdot \gamma' \cdot \xi')$. Clearly

$$MC(\xi \cdot \eta' \cdot \xi') = \frac{C + z \cdot c(\delta)}{D + z \cdot len(\delta)} > \frac{C + k \cdot c(\delta)}{D + k \cdot len(\delta)} = MC(\xi \cdot \hat{\eta} \cdot \xi'). \quad (1)$$

Denote $z' \geq 1$ the number such that $z = k + z'$. From (1) we gradually get

$$\begin{aligned} k \cdot C \cdot len(\delta) + z \cdot D \cdot c(\delta) &> k \cdot D \cdot c(\delta) + z \cdot C \cdot len(\delta) \\ z' \cdot D \cdot c(\delta) &> z' \cdot C \cdot len(\delta) \\ D \cdot c(\delta) &> C \cdot len(\delta) \\ (k-1) \cdot D \cdot c(\delta) &> (k-1) \cdot C \cdot len(\delta) \quad (\text{since } k \geq 2) \\ C \cdot len(\delta) + k \cdot D \cdot c(\delta) &> k \cdot C \cdot len(\delta) + D \cdot c(\delta) \\ \frac{C + k \cdot c(\delta)}{D + k \cdot len(\delta)} &> \frac{C + c(\delta)}{D + len(\delta)}. \end{aligned}$$

But then $(\xi \cdot \gamma \cdot \delta \cdot \gamma' \cdot \xi')$ is a T-visiting cycle with $MC(\xi \cdot \gamma \cdot \delta \cdot \gamma' \cdot \xi') = (C + c(\delta))/(D + len(\delta)) < MC(\xi \cdot \hat{\eta} \cdot \xi') = MC(\beta)$, a contradiction with the optimality of β.

Now let $\chi = (r, q, t, m, n, b)$ be any characteristic of η'. We show that η' is optimal for this characteristic. Assume, for the sake of contradiction, that it is not optimal for χ. Two cases may happen:

- There is an r-t-segment $\alpha_0 \cdot \alpha_1$ of a characteristic $(r, q, t, m, 0, b)$ such that $c(\alpha_0 \cdot \alpha_1) < c(\gamma \cdot \gamma')$. Then $\xi \cdot \alpha_0 \cdot \delta^z \cdot \alpha_1 \cdot \xi'$ is a T-visiting cycle with $MC(\xi \cdot \alpha_0 \cdot \delta^z \cdot \alpha_1 \cdot \xi') < MC(\xi \cdot \eta' \cdot \xi') = MC(\beta)$, a contradiction with the optimality of β.
- There is a cycle θ of length n initiated in q such that θ is either a segment or does not contain any reload state and $c(\theta) < c(\delta)$. Then $\xi \cdot \gamma \cdot \theta^x \cdot \gamma' \cdot \xi$ is again a T-visiting cycle whose mean cost is smaller than $MC(\beta)$, a contradiction.

□

D.3 A Proof of Lemma 18

Lemma 18. *There is an algorithm which decides, for a given characteristic χ, whether the set of all compact segments that have a characteristic χ is non-empty, and if the answer is* yes*, it computes a tuple $(\gamma, \gamma', \delta, k)$ such that $\gamma \cdot \delta^k \cdot \gamma'$ is a compact segment optimal for χ. The algorithm runs in polynomial time.*

Proof. Let $\chi = (r, q, t, m, n, b)$ be the input characteristic. From the definition of an optimal compact segment for χ it follows that we have to compute the following:

- If $n > 0$ we have to compute a cycle δ^* of minimal cost among all cycles δ satisfying the following: $len(\delta) = n$, $\delta(0) = q$, and δ is either a segment (if $q \in R$), or δ does not contain any reload state (if $q \notin R$). In both cases we can use the algorithm of Lemma 27, namely return the result of $MinPath(q, q, n, R)$ as the desired cycle δ^*. If $MinPath(q, q, n, R)$ answers that the required path does not exist, we can immediately say that no compact segment has χ as its characteristic, i.e., we return "no".
- If $b = 0$, we have to compute a compact segment α^* of minimal cost among all compact segments that have a characteristic $(r, q, t, m, 0, 0)$. Path α^* can be computed using the algorithm *MinPathReach* of Lemma 28, namely by making a call $MinPathReach(r, t, m, R, \{q\})$. If the result of this call is a non-existence of the required path, we again immediately return "no".

– If $b = 1$, we have to compute a compact segment α^* of minimal cost among all compact segments that have a characteristic $(r, q, t, m, 0, 1)$. If r, q or $t \in T$, we can proceed as in the previous case. Otherwise the path α^* can be computed as follows. For every $0 \leq m' \leq m$ we compute these paths:
  - $\alpha_{m',1}$ by calling *MinPathReach*(r, q, m', R, T),
  - $\alpha_{m-m',1}$ by calling *MinPath*$(q, t, m - m', R)$,
  - $\alpha_{m',2}$ by calling *MinPath*(r, q, m', R),
  - $\alpha_{m-m',2}$ by calling *MinPathReach*$(q, t, m - m', R, T)$.

  If for all such m' and all $i \in \{1, 2\}$ one of the paths $\alpha_{m',i}$, $\alpha_{m-m',i}$ does not exist, we immediately return "no".Then we select $0 \leq m' \leq m$ and $i \in \{1, 2\}$ that minimizes $c(\alpha_{m',i} \cdot \alpha_{m-m',i})$ and we return $\alpha^* := \alpha_{m',i} \cdot \alpha_{m-m',i}$. The correctness of this is clear, since every compact segment $\gamma \cdot \gamma'$ of a characteristic $(r, q, t, m, 0, 1)$ and of minimal cost has the property that γ or γ' satisfies the conditions stated in Lemma 28 (and it is of minimal cost among all paths satisfying these conditions), and both γ and γ' satisfy the conditions stated in Lemma 27, with one of them having minimal cost among all paths satisfying this condition.

Now having the paths α^* (and δ^* if $n > 0$) we write $\alpha^* = \gamma \cdot \gamma'$, where $last(\gamma) = q$, and check whether $c(\alpha^*) \leq cap$ or $c(\gamma \cdot \delta * \cdot \gamma') \leq cap$, depending on whether $n = 0$ or not. If this check fails, we return "no." Otherwise, if $n = 0$ we set $\delta^* = q$ and $k = 1$, else we set $k = \lfloor (cap - c(\alpha^*))/c(\delta^*) \rfloor$ (this number exists since $c(\delta*) > 0$ – otherwise there would be a zero-cost cycle on an admissible state q, a contradiction with the assumptions of Proposition 15). Clearly $\gamma \cdot (\delta^*)^k \gamma'$ is a compact segment optimal for χ, so we return the tuple $(\gamma, \gamma', \delta^*, k)$ as the desired result. □

D.4 A Proof of Lemma 19

Lemma 19. *There is an optimal T-visiting cycle β such that every segment of β is of the form CPath(χ) for some characteristic χ.*

Proof. First note that if a compact segment η contains a state from T and at the same time it is optimal for some characteristic $\chi = (r, q, t, m, n, 0)$, then η also has a characteristic $\chi' = (r, q, t, m, n, 1)$ (recall the last condition from the definition of a characteristic). Since every compact segment with a characteristic χ' also has a characteristic χ, η is optimal also for χ'.

Now among all optimal T-visiting cycles whose all segments are compact and optimal for some characteristic (at least one exists due to Lemma 17) let β be the one minimizing the number of segments that are not of the form *CPath*(χ) for some χ. Suppose that β contains such a segment η and denote χ a characteristic of η for which η is optimal. As mentioned above, if η contains a state from T, then we may assume that the last component of χ is 1. Write $\beta = \xi \cdot \eta \cdot \xi'$. By Lemma 16 we have $len(\eta) = len(CPath(\chi))$ and $c(\eta) = c(CPath(\chi))$ and thus $MC(\beta) = MC(\beta')$ where $\beta' = \xi \cdot CPath(\chi) \cdot \xi'$. Note that if η contains a state from T then, by our assumption, the last component of χ is 1, so *CPath*(χ) also contains a state from T. Thus, β' is an optimal T-visiting cycle containing smaller number of undesirable segments than β, a contradiction with the choice of β. □

E Proofs of Section 3.3

E.1 A Proof of Lemma 20

Lemma 20. *$Val_C(s)$ is finite iff there is a safe cycle, in which case $Val_C(s) = \min\{MC(\beta) \mid \beta$ is a safe cycle$\}$. Further, there is a finite $cap \in \mathbb{N}_0$ such that $Val_C^{cap}(s) = Val_C(s)$ iff either $Val_C(s) = \infty$, or there is a strongly safe cycle $\hat{\beta}$ such that $MC(\hat{\beta}) = Val_C(s)$. In such a case $Val_C^{cap}(s) = Val_C(s)$ for every $cap \geq 3 \cdot |S| \cdot c_{\max}$, where $c_{\max}$ is the maximal cost of a transition in C.*

Proof. Denote $MS = \min\{MC(\beta) \mid \beta$ is a safe cycle$\}$. We say that a simple cycle δ is a *simple sub-cycle* of β if $\beta = \xi \cdot \delta \cdot \xi'$ for some ξ, ξ'. We say that a set of simple cycles D is a *decomposition of* β (into simple cycles) if for every two distinct $\delta, \delta' \in D$ it holds that δ' is a simple sub-cycle of $\xi \cdot \xi'$, where $\beta = \xi \cdot \delta \cdot \xi'$. Note that for any decomposition D of β it holds $MC(\beta) \geq \min_{\delta \in D} MC(\delta)$.

Suppose that $Val_C(s) < \infty$. We show that a safe cycle exists and moreover, for any capacity cap we have $Val_C^{cap}(s) \geq MS$, from which it follows that $Val_C(s) \geq MS$. If $Val_C^{cap}(s) = \infty$, the inequality is trivial. Otherwise the inequality follows from the first part of the following claim (its second part will be used later in the proof):

Claim. If $Val_C^{cap}(s) < \infty$, then there is a safe cycle δ such that $Val_C^{cap}(s) \geq MC(\delta)$. Moreover, if $Val_C^{cap}(s) = Val_C(s)$, then δ is strongly safe.

Indeed, from Lemma 11 and 14 it follows that there is a path α and cycles β, γ such that γ contains an accepting state and either $c(\beta) = 0$ or $\beta(0) \in R$, and for a run $\varrho = \alpha \cdot \beta \cdot \gamma \cdot \beta^2 \cdot \gamma \cdot \beta^4 \cdots$ it holds $MC(\beta) = Val_C^{cap}(\varrho) = Val_C^{cap}(s)$. Note that δ and γ must be in the same strongly connected component C of C. In particular, all states of β are inter-reachable with an accepting state.

Now we consider two cases. Either $c(\beta \cdot \gamma) = 0$, in which case $\beta \cdot \gamma$ has a sub-path δ which is a simple cycle of zero cost containing an accepting state – i.e., δ is a strongly safe cycle and $Val_C^{cap}(s) = 0 = MC(\delta) = MS$.

If $c(\beta \cdot \gamma) \neq 0$, we distinguish two sub-cases. Either $c(\beta) = 0$. Then $c(\gamma) \neq 0$ and it follows that $\beta \cdot \gamma$ contains a reload state. Then every simple sub-cycle δ of β has $\delta(0)$ inter-reachable not only with an accepting state (as shown above), but also with a reload state and hence it is a strongly safe cycle (of zero cost). Here again $Val_C^{cap}(s) = 0 = MC(\delta) = MS$.

The second sub-case is $c(\beta) \neq 0$. Then $\beta(0) \in R$ and thus all states on β are inter-reachable with a reload state. Let D be a decomposition of β into simple cycles and $\delta \in D$ be the simple cycle of minimal mean cost. Clearly δ is a safe cycle and $MC(\beta) \geq MC(\delta)$, which finishes the proof of the first part of the claim. Now pick such a simple $\delta \in D$ which contains a reload state contained in β. Clearly this δ is strongly safe. We claim that either $MC(\delta) \leq Val_C^{cap}(s)$ or $Val_C^{cap}(s) < Val_C(s)$, which finishes the proof of the second part of the claim. So suppose that $MC(\delta) > Val_C^{cap}(s) = MC(\beta)$ and write $\beta = \xi \cdot \delta \cdot \xi'$. Clearly $MC(\xi \cdot \xi') < MC(\delta)$ and thus for the cycle $\beta' = \xi \cdot \xi' \cdot \xi \cdot \delta \cdot \xi'$ it holds $MC(\beta') < MC(\beta)$. Moreover, the run $\varrho' = \alpha \cdot \beta' \cdot \gamma \cdot \beta'^2 \cdot \gamma \cdot \beta'^4 \cdots$ is accepting and cap'-bounded for $cap' = \max\{cap, c(\beta' \cdot \gamma)\}$. Also note that $Val_C^{cap'}(\varrho) = MC(\beta')$ (this

can be established via a straightforward computation identical to the one from the proof of Proposition 14). Thus, $Val_C^{cap}(s) = MC(\beta) > MC(\beta') \geq Val_C^{cap'}(s) \geq Val_C(s)$.

Conversely, suppose that there is a safe cycle and let β be the one of minimal mean cost. We show that for every $1 \geq \varepsilon > 0$ and every capacity $cap \geq \lceil (6|S|^2 c_{\max}^2)/\varepsilon \rceil$ there is a run ϱ_ε such that $Val_C^{cap}(\varrho_\varepsilon) \leq MC(\beta) + \varepsilon$. From this it immediately follows that $Val_C(s)$ is finite, and in combination with the previous paragraph we get $Val_C(s) = MS$.

Let α be a shortest (w.r.t. the number of transitions) path from s to $\beta(0)$. Note that $c(\alpha) \leq |S| \cdot c_{\max}$. If $c(\beta) = 0$ and β contains an accepting state, then we can take $\varrho_\varepsilon := \alpha \cdot \beta^\omega$, since for every $cap \geq |S| c_{\max}$ this is a cap-bounded accepting run with $Val_C^{cap}(\varrho_\varepsilon) = MC(\beta)$. Otherwise let γ_1 be a shortest path from $\beta(0)$ to some accepting state f, γ_2 a shortest path from f to some reload state r, and γ_3 a shortest path from r to β_0, and put $\gamma = \gamma_1 \cdot \gamma_2 \cdot \gamma_3$. Note that $c(\gamma) \leq 3|S| c_{\max}$. Set $k = \lceil (3|S| c_{\max})/\varepsilon \rceil$. It easily follows that $\varrho_\varepsilon := \alpha \cdot (\gamma \cdot \beta^k)^\omega$ is a cap-bounded accepting path for any $cap \geq \lceil (6|S|^2 c_{\max}^2)/\varepsilon \rceil$, since the consumption between two visits of the reload state r on γ is bounded by $c(\gamma) + k \cdot c(\beta) \leq 3|S| c_{\max} + 3|S|^2 c_{\max}^2/\varepsilon$. Let us compute $MC(\varrho_\varepsilon)$. We have

$$MC(\varrho_\varepsilon) = MC(\gamma \cdot \beta^k) = \frac{c(\gamma) + k \cdot c(\beta)}{len(\gamma) + k \cdot len(\beta)} \leq \frac{c(\gamma)}{k \cdot len(\beta)} + MC(\beta).$$

Now $c(\gamma)/(k \cdot len(\beta)) \leq c(\gamma)/k \leq 3|S| \cdot c_{\max}/k \leq \varepsilon$ as required.

Now suppose that there is a finite capacity cap such that $Val_C^{cap}(s) = Val_C(s)$ (= MS, as shown above). From the above claim it immediately follows that there is a strongly safe cycle δ such that $MC(\delta) = MS$.

Conversely, suppose that $\hat{\beta}$ is a strongly safe cycle with $MC(\hat{\beta}) = Val_C(s)$. Let α be again a shortest path from s to $\beta(0)$. If $c(\hat{\beta}) = 0$ and $\hat{\beta}$ contains an accepting state, then we again take $\varrho = \alpha \cdot \hat{\beta}^\omega$ – this is clearly an $(|S| \cdot c_{\max})$-bounded run and $Val_C^{cap}(\varrho) = 0 = Val_C(s)$ for any $cap \geq |S| \cdot c_{\max}$. Now suppose that $c(\hat{\beta}) = 0$ and $\hat{\beta}(0)$ is inter-reachable with accepting state f and with an accepting state r. Then there exists a cycle γ initiated in $\hat{\beta}(0)$ that contains both f and r. Then $\varrho := \alpha \cdot \hat{\beta} \cdot \gamma \cdot \hat{\beta}^2 \cdot \gamma \cdot \hat{\beta}^4$ is a $(3|S| c_{\max})$-bounded accepting run with $Val_C^{3|S| c_{\max}} = MC(\hat{\beta}) = 0 = Val_C(s)$.

Finally, suppose that $\hat{\beta}$ contains a reload state and $\hat{\beta}(0)$ is inter-reachable with an accepting state. Let γ be the cycle of minimal length among those initiated in $\hat{\beta}(0)$ that contain an accepting state. Then $\varrho := \alpha \cdot \hat{\beta} \cdot \gamma \cdot \hat{\beta}^2 \cdot \gamma \cdot \hat{\beta}^4$ is again a $(3|S| c_{\max})$-bounded accepting run, where the boundedness now comes from the fact that $\hat{\beta}$ contains a reload state and thus the end cost of $\hat{\beta}^k$ is at most the end cost of $\hat{\beta}$, which is at most $|S| \cdot c_{\max}$. Clearly, $Val_C^{3|S| c_{\max}}(\varrho) = MC(\beta) = Val_C(s)$.

□

E.2 A Proof of Lemma 21

Lemma 21. *The existence of a safe (or strongly safe) cycle is decidable in polynomial time. Further, if a safe (or strongly safe) cycle exists, then there is a safe (or strongly safe) cycle β computable in polynomial time such that $MC(\beta) \leq MC(\beta')$ for every safe (or strongly safe) cycle β'.*

Proof. To find a safe cycle β of minimal mean cost (or to decide that no such cycle exists), we proceed as follows. First we compute the set A of all states reachable from s. Then for every state $f \in F \cap A$ we use the polynomial-time algorithm of Lemma 25 to find a cycle of zero cost initiated in f. If we find such a cycle δ for some $f \in F \cap A$, then clearly δ is a safe cycle of minimal (i.e., zero) mean cost. If we do not find such a cycle, we decompose C into its strongly connected components (SCCs), using, e.g., the Tarjan's algorithm. For every SCC C that is reachable from s we check, whether C contains both a reload state and an accepting state. If no such component exists, we conclude that there is no safe cycle. Otherwise, for every SCC C that contains both a reload state and an accepting state we compute a cycle δ_C of length at most $|S|$ of minimal mean cost in C, using standard polynomial time-algorithm for finding a cycle of the minimal mean cost (see, e.g., [15, 16]). [2] Then clearly the cycle δ_{C^*} such that $MC(\delta_{C^*}) = \min\{MC(\delta_C) \mid C \text{ is a SCC containing a reload and an accepting state}\}$ is a safe cycle of minimal cost among all safe cycles in C.

For strongly safe cycles we proceed in a similar way. First we check whether there is a cycle of zero cost containing a reachable accepting state using the same approach as in the previous paragraph. If we find such a cycle, than it is a strongly safe cycle of minimal mean cost. Otherwise, we again decompose C into SCCs and identify those SCCs reachable from s that contain both a reload state and an accepting state. Let X be the set of all such SCCs. If $X = \emptyset$, we immediately get that no strongly safe cycles exist. Otherwise for every SCC $C \in X$ we check, whether there is a cycle of zero cost in C, using again the algorithm from Lemma 25 (we can also use the aforementioned algorithms for finding a cycle of minimal mean cost). If such a cycle exists for some $C \in X$, it is clearly a strongly safe cycle of minimal mean cost. Otherwise, we have to find a cycle δ of length at most $|S|$ in some $C \in X$ such that δ contains a reload state (and we of course need to find a cycle of minimal mean cost among all such cycles). To this end, for every $C \in X$ and every reload state $r \in C \cap R$ we construct a labelled graph $G_r = (V, \mapsto, L, \ell)$, where $L \subset \mathbb{N}_0$ defined as follows:

- $V = C \times \{0, \ldots, |S|\}$,
- there is an edge $(q, i) \overset{a}{\mapsto} (q', j)$ in G_r, here $0 \leq i, j \leq |S|$, whenever $i = j + 1$ and $q \xrightarrow{a} q'$ is a transition in C,
- for every $1 \leq i \leq |S|$ there is an edge $(r, i) \overset{0}{\mapsto} (r, 0)$,
- there are no other edges.

Note that G_r does not have to contain a cycle if C does not contain a cycle, which may happen if C contains a single state without a self-loop. If this is the case for all $C \in X$, we get that there are no strongly safe cycles. Otherwise note that every cycle in some G_r contains the state $(r, 0)$. Moreover, there is a natural many-to-one correspondence between the simple cycles in G_r and cycles of length at most $|S|$ that are initiated in r in C, and this correspondence preserves the mean cost of the cycles. So in order to compute a cycle δ of minimal mean cost among all cycles that are initiated in a reload state of some $C \in X$, it suffices to compute, for every $C \in X$ and every $r \in C \cap R$ a simple cycle δ_r of minimal mean cost in G_r, using the standard algorithms mentioned above. If we

[2] Note that given any cycle θ of minimal mean cost, we can easily extract from θ a simple cycle δ such that $MC(\delta) = MC(\theta)$.

then select r^* such that $MC(\delta_{r^*}) = \min\{MC(\delta_r) \mid C \in X, r \in C \cap R\}$, then from δ_{r^*} we can easily compute the corresponding cycle δ'_{r^*} in C that is a strongly safe cycle of minimal mean cost among all strongly safe cycles.

The correctness of the algorithm and its polynomial running time are immediate. □

E.3 A Proof of Lemma 22

Lemma 22. *Let $c_{\max}$ be the maximal cost of a transition in C. For every $cap > 4 \cdot |S| \cdot c_{\max}$ we have that*

$$Val_C^{cap}(s) - Val_C(s) \leq \frac{3 \cdot |S| \cdot c_{\max}}{cap - 4 \cdot |S| \cdot c_{\max}}.$$

Proof. The proof employs techniques very similar to those used in the proof of Lemma 20. If $Val_C(s) = \infty$, then the lemma is immediate. Otherwise by Lemma 20 there is a safe cycle β such that $MC(\beta) = Val_C(s)$. Let α be the path from s to $\beta(0)$ of minimal length. If $c(\beta) = 0$ and β contains an accepting state, then for every $cap \geq |S| \cdot c_{\max}$ we have $Val_C^{cap}(s) = Val_C^{cap}(\alpha \cdot \beta^\omega) = MC(\beta) = 0$, and the lemma holds. It remains to consider the case when $\beta(0)$ is inter-reachable with both a reload state and an accepting state. Then let γ_1 be a shortest (w.r.t. the number of transitions) path from $\beta(0)$ to some accepting state f, γ_2 a shortest path from f to some reload state r, and γ_3 a shortest path from r to β_0, and put $\gamma = \gamma_1 \cdot \gamma_2 \cdot \gamma_3$. Finally, put $k = \lfloor (cap - 3 \cdot |S| \cdot c_{\max})/(c(\beta)) \rfloor$. Note that $k \geq 1$ since $cap \geq 4|S|c_{\max}$. Then $\varrho := \alpha \cdot (\gamma \cdot \beta^k)^\omega$ is a cap-bounded accepting run, since the consumption between two visits of the reload state r on γ is bounded by $3|S|c_{\max} + k \cdot c(\beta) \leq 3|S|c_{\max} + cap - 3|S|c_{\max} = cap$. Moreover,

$$Val_C^{cap}(\varrho) = MC(\gamma \cdot \beta^\omega) = \frac{c(\gamma) + k \cdot c(\beta)}{len(\gamma) + k \cdot len(\beta)} \leq \frac{c(\gamma)}{k \cdot len(\beta)} + MC(\beta).$$

Now

$$\frac{c(\gamma)}{k \cdot len(\gamma)} \leq \frac{c(\gamma)}{\left(\frac{cap - 3|S|c_{\max}}{c(\beta)} - 1\right) \cdot c(\beta)} \leq \frac{3|S|c_{\max}}{cap - 3|S|c_{\max} - c(\beta)} \leq \frac{3|S|c_{\max}}{cap - 4|S|c_{\max}}$$

as required. □