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., [15]). This is motivated by the fact that the system is usually required to achieve some functionality, and not just "run" with minimal average costs.

* The authors are supported by the Czech Science Foundation, grant No P202/10/1469.

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 cannot 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 nondeterministic 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 nontrivial 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 [15] 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 [11]. 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 [14]. Energy games with capacity constraints were studied in [18]. 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 [10], 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* [12], 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 [12] 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* [18, 4], *generalized energy games* [13], *consumption games* [7], etc. We refer to [19] 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.

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 $\varrho \equiv v_0 v_1 \dots$ 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 $v_0 \dots v_i$ of ϱ of length i.

A finite path $\alpha \equiv v_0 \dots v_n$ 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 $\alpha \equiv v_0 \dots v_n$ and a finite or infinite path $\varrho \equiv u_0 u_1 \dots$ such that $v_n = u_0$, we use $\alpha \cdot \varrho$ to denote the *concatenation* of α and ϱ, i.e., the path $v_0 \dots v_n u_1 u_2 \dots$ 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 $s_{i+1}, \dots, s_{n+1} \notin R$ (intuitively, the end cost of α is the total amount of resources consumed since the last reload, or since the start if no reload happened on α).

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* (we assume that initially the battery is loaded to a full capacity). 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

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

[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.

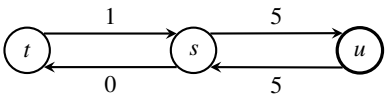

Fig. 1: An optimal controller may require memory of exponential size. Here $R = \{u\}$ and $F = S$.

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

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

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, a *cap*-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 the full version of this paper [9].

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 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 omitted. They can be found in [9].

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 (which is a simple cycle), 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 β is *reload-short* (i.e., it contains at most $|R|$ occurrences of a reload state), $MC(\beta) \leq MC(\delta)$ for every *cap*-bounded cycle δ initiated in an admissible state, and $\beta(0) \neq \beta(i)$ for all $0 < i < len(\beta)$;
- a reload-short *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 $\|cap\|$. However, we show that there is always $\hat{\beta}$ of a special structure for which we can compute (in polynomial time) a *counting* controller $\kappa[\hat{\beta}]$ of *polynomial* size such that $run(\kappa[\hat{\beta}], \hat{\beta}(0)) = \hat{\beta}^\omega$. Since α can be computed in polynomial time, it follows that we can obtain, in polynomial time, a counting controller κ of polynomial size such that $run(\kappa, s) = run(\mu, s)$, i.e., κ is *cap*-optimal in s.
- If $MC(\beta) < MC(\hat{\beta})$, then *every cap*-optimal controller for s has infinite memory. Again, we show that there is always β of a special structure, for which we can 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 a *cap*-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) of length at most $|S|^2$ 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 any 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 exists an advancing controller π optimal for s where $\|\pi\|$ is polynomial in $\|C\|$ and $\|cap\|$.*

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-short*, if β contains at most $|R|$ occurrences of a reload state. Further, we say that a cycle β is *T-visiting*, where $T \subseteq S$, if β is a *cap*-bounded reload-short cycle initiated in an admissible reload state such that β contains a state of T and $\beta(0) \neq \beta(i)$ for all $0 < i < len(\beta)$. 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-shortness 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 $Val_C^{cap}(\varrho) \geq MCS$. 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-short 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 $MC(\varrho) = MC(\varrho') \geq \inf_{i \geq 1} MC(\beta_i) \geq \inf_{i \geq 0, 1 \leq j \leq i_m} MC(\beta_{i,j}) \geq MCS$.

Now let us 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 $Val_C^{cap}(\varrho) = MC(\beta) = MCS = Val_C^{cap}(s)$. Hence, there exists an optimal advancing controller for s. It remains to show that there is no optimal finite memory controller for s. For every FMC μ we can write $run(\mu, s) \equiv \hat{\alpha} \cdot \hat{\beta}^\omega$, 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, using the same technique as in the first paragraph of this proof, into *finitely many* reload-short cycles on reloading states, whose mean cost is at least MCS. At least one of these cycles is F-visiting. Since $MC(\hat{\beta})$ is a convex combination of the mean-costs of these cycles 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-short 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 $\gamma \cdot \delta^k \cdot \gamma'$ (if $CTuple(\chi) = \bot$, 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 $\beta = CPath(\chi_0) \cdot CPath(\chi_1) \cdots CPath(\chi_j)$ 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 $0 \leq i \leq j$. 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 corresponding sequence of tuples $CTuple(\chi_0), \cdots, CTuple(\chi_j)$.

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 in $X_{r,t}$ 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:

- $V = (R' \cup \{CTuple(\chi) \mid \chi \in X\}) \times \{0, \ldots, |S|\}$.
- 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 $((r, i), (CTuple(\chi), i))$ labelled by $(c(CPath(\chi)), len(CPath(\chi)))$ and an edge $((CTuple(\chi), i), (t, i+1))$ labelled by $(0, 0)$.

– For every state $t \in R'$ and every characteristic $\chi \in X^1_{\hat{q},t}$ there is an edge $((\hat{q},0),(\mathit{CTuple}(\chi),0))$ labelled by $(c(\mathit{CPath}(\chi)), \mathit{len}(\mathit{CPath}(\chi)))$ and an edge $((\mathit{CTuple}(\chi),0),(t,1))$ 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 $\mathit{CTuple}(\chi) = (\gamma, \gamma', \delta, k)$ using Lemma 18. Then, $\mathit{len}(\mathit{CPath}(\chi)) = \mathit{len}(\gamma) + \mathit{len}(\gamma') + k \cdot \mathit{len}(\delta)$, and similarly for $c(\mathit{CPath}(\chi))$. Note that every cycle in $G_{\hat{q}}$ contains the vertex $(\hat{q},0)$. Some of the constructed graphs $G_{\hat{q}}$ may not contain a cycle (the out-degree of $(\hat{q},0)$ may be equal to 0), but, as we shall see, 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 $\mathit{rat}(\hat{\beta}) = (c_0 + c_1 + \cdots + c_{h-1})/(d_0 + d_1 + \cdots d_{h-1})$ (the denominator is positive due to the construction of $G_{\hat{q}}$). Now let $\hat{q} \in R'$ be arbitrary. Every cycle $\beta_{\hat{q}}$ in $G_{\hat{q}}$ (we can assume that it is initiated in $(\hat{q},0)$) uniquely determines a T-visiting cycle $\Psi(\beta_{\hat{q}})$ in C that is initiated in $\hat{q}$ and whose every segment has the form $\mathit{CPath}\chi$ for some χ. To see this, note that every second vertex on $\beta_{\hat{q}}$ is a 4-tuple of the form $\mathit{CTuple}(\chi)$ for some χ, so if $\mathit{CTuple}(\chi_0), \mathit{CTuple}(\chi_1), \ldots, \mathit{CTuple}(\chi_j)$ is the sequence of these 4-tuples in order in which they appear in $\beta_{\hat{q}}$, then we put $\Psi(\beta_{\hat{q}}) = \mathit{CPath}(\chi_0) \cdot \mathit{CPath}(\chi_1) \cdots \mathit{CPath}(\chi_j)$. Clearly $\mathit{MC}(\Psi(\beta_{\hat{q}})) = \mathit{rat}(\beta_{\hat{q}})$. Moreover, it is easy to see that Ψ is a bijection between the set of all cycles that appear in some $G_{\hat{q}}$ and the set of all T-visiting cycles in C whose segments are all of the form $\mathit{CPath}(\chi)$ for some χ (by Lemma 13, the latter of these sets – and thus both of them – must be non-empty). Thus, in order to find an optimal T-visiting cycle, the algorithm finds, for every $\hat{q} \in R'$, a simple cycle $\beta_{\hat{q}}$ 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., [16, 17]), then simply picks $\hat{r} \in R'$ such that the ratio of $\beta_{\hat{r}}$ is minimal and computes $\Psi(\beta_{\hat{r}})$. The fact that $\Psi(\beta_{\hat{r}})$ is an optimal T-visiting cycle follows from the above observations and from Lemma 19.

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 of s by interleaving a large number of iterations of some "cheap" cycle with visits to an accepting state. This motivates our definitions of *safe* and *strongly safe* cycles. Intuitively, a cycle is safe if, assuming unbounded battery capacity, the controller can interleave an arbitrary finite number of iterations of this cycle 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. We say that a cycle β of length at most $|S|$ and with $\beta(0)$ 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 obtain 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, D. Klaška, A. Kučera, and P. Novotný. Minimizing Running Costs in Consumption Systems. Technical report. Available at http://arxiv.org/abs/1402.4995.
10. 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.
11. K. Chatterjee and L. Doyen. Energy Parity Games. In *Proceedings of ICALP 2010, Part II* [2], pages 599–610.
12. 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.
13. 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.
14. 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.
15. K. Chatterjee, T. Henzinger, and M. Jurdziński. Mean-Payoff Parity Games. In *Proceedings of LICS 2005*, pages 178–187. IEEE Computer Society Press, 2005.
16. 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.
17. 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.
18. 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.
19. A. Kučera. Playing Games with Counter Automata. In *Reachability Problems*, volume 7550 of *Lecture Notes in Computer Science*, pages 29–41. Springer, 2012.