Efficient Controller Synthesis for Consumption Games with Multiple Resource Types

Tomáš Brázdil[1⋆], Krishnendu Chatterjee[2⋆], Antonín Kučera[1⋆], and Petr Novotný[1⋆]

[1] Faculty of Informatics, Masaryk University, Brno, Czech Republic.
{brazdil,kucera,xnovot18}@fi.muni.cz
[2] IST Austria, Klosterneuburg, Austria.
krish.chat@gmail.com

Abstract. We introduce *consumption games*, a model for discrete interactive system with multiple resources that are consumed or reloaded independently. More precisely, a consumption game is a finite-state graph where each transition is labeled by a vector of resource updates, where every update is a non-positive number or ω. The ω updates model the reloading of a given resource. Each vertex belongs either to player $\Box$ or player $\Diamond$, where the aim of player $\Box$ is to play so that the resources are never exhausted. We consider several natural algorithmic problems about consumption games, and show that although these problems are computationally hard in general, they are solvable in polynomial time for every fixed number of resource types (i.e., the dimension of the update vectors) and bounded resource updates.

1 Introduction

In this paper we introduce *consumption games*, a model for discrete interactive systems with multiple resources that can be consumed and reloaded independently. We show that consumption games, despite their rich modeling power, still admit efficient algorithmic analysis for a "small" number of resource types. This property distinguishes consumption games from other related models, such as games over vector addition systems or multi-energy games (see below), that are notoriously intractable.

Roughly speaking, a consumption game is a finite-state directed graph where each state belongs either to player $\Box$ (controller) or player $\Diamond$ (environment). Every transition $s \to t$ is labeled by a d-dimensional vector δ such that each component $\delta(i)$ is a non-positive integer (encoded in *binary*) or ω. Intuitively, if $\delta(i) = -n$, then the current load of the i-th resource is decreased by n while performing $s \to t$, and if $\delta(i) = \omega$, then the i-th resource can be "reloaded" to an arbitrarily high value greater than or equal to the current load. A *configuration* of a consumption game is determined by the current control state and the current load of all resources, which is a d-dimensional vector of positive integers. A play of a consumption game is initiated in some state and some

⋆ Tomáš Brázdil, Antonín Kučera, and Petr Novotný are supported by the Czech Science Foundation, grant No. P202/10/1469. Krishnendu Chatterjee is supported by the FWF (Austrian Science Fund) NFN Grant No S11407-N23 (RiSE) and ERC Start grant (279307: Graph Games).

initial load of resources. The aim of player $\Box$ is to play *safely*, i.e., select transitions in his states so that the vector of current resource loads stays positive in every component (i.e., the resources are never exhausted). Player $\Diamond$ aims at the opposite.

The resources may correspond to fuel, electricity, money, or even more abstract entities such as time or patience. To get a better intuition behind consumption games and the abstract problems studied in this paper, let us discuss one particular example in greater detail.

The public transport company of Brno city[4] maintains the network of public trams, buses, trolleybuses, and boats. Due to the frequent failures and breakdowns in electrical wiring, rails, railroad switches, and the transport vehicles, the company has several emergency teams which travel from one accident to another according to the directives received from the central supervisory office. Recently, the company was considering the possibility of replacing their old diesel vans by new cars equipped with more ecological natural gas engines. The problem is that these cars have smaller range and can be tanked only at selected gas stations. So, it is not clear whether the cars are usable at all, i.e., whether they can always visit a gas station on time regardless where and when an accident happens, and what are the time delays caused by detours to gas stations. Now we indicate how to construct the associated consumption game model and how to rephrase the above questions formally.

We start with a standard graph G representing the city road network, i.e., the nodes of G correspond to distinguished locations (such as crossings) and the edges correspond to the connecting roads. Then we identify the nodes corresponding to gas stations that sell natural gas, and to each edge (road) we assign two negative numbers corresponding to the expected time and fuel needed to pass the road. Every morning, a car leaves a central garage (where it is fully tanked) and returns to the same place in the evening. The maximal number of accidents serviced per day can be safely overestimated by 12. Our consumption game C has two resource types modeling the fuel and time in the expected way. The fuel is consumed by passing a transition (road), and can be reloaded by the outgoing transitions of gas stations. The time is also consumed by passing the roads, and the only node where it can be reloaded is the central garage, but only after completing the 12 jobs. In the states of C we remember the current job number (from 1 to 12) and the current target node. At the beginning, and also after visiting the current target node, the next target node is selected by player $\Diamond$. Technically, the current target node belongs to player $\Diamond$, and there is a transition for every (potential) next target node. Performing such a transition does not consume the resources, but the information about the next target node is stored in the chosen state, job index is increased, and the control over the play is given back to player $\Box$ who models the driver. This goes on until the job index reaches 12. Then, player $\Diamond$ makes no further choice, but it is possible to reload the time resource at the node corresponding to the central garage, and hence player $\Box$ aims at returning to this place as quickly as possible (without running out of gas). Note that C has about $12 \cdot n^2$ states, where n is the number of states of G.

The question whether the new cars are usable at all can now be formalized as follows: *Is there is* safe *strategy for player* $\Box$ *in the initial configuration such that the fuel resource is never reloaded to a value which is higher than the tank capacity of the car?*

[4] DPMB, Dopravní Podnik Města Brna.

In the initial configuration, the fuel resource is initialized to 1 because it can be immediately reloaded in the central garage, and the time resource is initialized to a "sufficiently high value" which is efficiently computable due to the *finite reload property* formulated in Corollary 7. Similarly, the extra time delays caused by detours to gas stations can be estimated by computing the *minimal initial credit* for the time resource, i.e., the minimal initial value sufficient for performing a safe strategy, and comparing this number with the minimal initial credit for the time resource in a simplified consumption game where the fuel is not consumed at all (this corresponds to an ideal "infinite tank capacity"). Similarly, one could also analyze the extra fuel costs, or model the consumption of the material needed to perform the repairs, and many other aspects.

An important point of the above example is that the number of resources is relatively small, but the number of states is large. This motivates the study of *parameterized complexity* of basic decision/optimization problems for consumption games, where the parameters are the following:

- d, the number of resources (or *dimension*);
- ℓ, the maximal finite $|\delta(i)|$ such that $1 \leq i \leq d$ and δ is a label of some transition.

Main Results. For every state s of a consumption game C, we consider the following sets of vectors (see Section 2 for precise definitions):

- $\mathit{Safe}(s)$ consists of all vectors α of positive integers such that player $\Box$ has a safe strategy in the configuration (s, α). That is, $\mathit{Safe}(s)$ consists of all vectors describing a sufficient *initial* load of all resources needed to perform a safe strategy.
- $\mathit{Cover}(s)$ consists of all vectors α of positive integers such that player $\Box$ has a safe strategy σ in the configuration (s, α) such that for every strategy π for player $\Diamond$ and every configuration (t, β) visited during the play determined by σ and π we have that $\beta \leq \alpha$. Note that physical resources (such as fuel, water, electricity, etc.) are stored in devices with finite capacity (tanks, batteries, etc.), and hence it is important to know what capacities of these devices are sufficient for performing a safe strategy. These sufficient capacities correspond to the vectors of $\mathit{Cover}(s)$.

Clearly, both $\mathit{Safe}(s)$ and $\mathit{Cover}(s)$ are upwards closed with respect to component-wise ordering. Hence, these sets are fully determined by their *finite* sets of minimal elements. In this paper we aim at answering the very basic algorithmic problems about $\mathit{Safe}(s)$ and $\mathit{Cover}(s)$, which are the following:

(A) *Emptiness*. For a given state s, decide whether $\mathit{Safe}(s) = \emptyset$ (or $\mathit{Cover}(s) = \emptyset$).
(B) *Membership*. For a given state s and a vector α, decide whether $\alpha \in \mathit{Safe}(s)$ (or $\alpha \in \mathit{Cover}(s)$). Further, decide whether α is a *minimal* vector of $\mathit{Safe}(s)$ (or $\mathit{Cover}(s)$).
(C) *Compute the set of minimal vectors* of $\mathit{Safe}(s)$ (or $\mathit{Cover}(s)$).

Note that these problems subsume the questions of our motivating example. We show that *all of these problems are computationally hard*, but solvable in *polynomial time* for every fixed choice of the parameters d and ℓ introduced above. Since the degree of the bounding polynomial increases with the size of the parameters, we do *not* provide fixed-parameter tractability results in the usual sense of parameterized complexity (as it is mentioned in Section 3, this would imply a solution to a long-standing open problem in study of graph games). Still, these results clearly show that for "small" parameter

values, the above problems *are* practically solvable even if the underlying graph of C is very large. More precisely, we show the following for game graphs with n states:

- The emptiness problems for $\mathit{Safe}(s)$ and $\mathit{Cover}(s)$ are **coNP**-complete, and solvable in $O(d! \cdot n^{d+1})$ time.
- The membership problems for $\mathit{Safe}(s)$ and $\mathit{Cover}(s)$ are **PSPACE**-hard and solvable in time $|\alpha| \cdot (d \cdot \ell \cdot n)^{O(d)}$ and $O(\Lambda^2 \cdot n^2)$, respectively, where $|\alpha|$ is the encoding size of α and $\Lambda = \Pi_{i=1}^{d} \alpha(i)$.
- The set of minimal elements of $\mathit{Safe}(s)$ and $\mathit{Cover}(s)$ is computable in time $(d \cdot \ell \cdot n)^{O(d)}$ and $(d \cdot \ell \cdot n)^{O(d \cdot d!)}$, respectively.

Then, in Section 4, we show that the complexity of some of the above problems can be substantially improved for two natural subclasses of *one-player* and *decreasing* consumption games by employing special methods. A consumption game is *one-player* if all states are controlled by player $\Box$, and *decreasing* if every resource is either reloaded or decreased along every cycle in the graph of C. For example, the game constructed in our motivating example is decreasing, and we give a motivating example for one-player consumption games in Section 4. In particular, we prove that

- the emptiness problem for $\mathit{Safe}(s)$ and $\mathit{Cover}(s)$ is solvable in polynomial time both for one-player and decreasing consumption games;
- the membership problem for $\mathit{Safe}(s)$ is **PSPACE**-complete (resp. **NP**-complete) for decreasing consumption games (resp. one-player consumption games).
- Furthermore, for both these subclasses we present algorithms to compute the minimal elements of $\mathit{Safe}(s)$ by a reduction to *minimum multi-distance reachability* problem, and solving the minimum multi-distance reachability problem on game graphs. Though these algorithms do not improve the worst case complexity over general consumption games, they are iterative and potentially terminate much earlier (we refer to Section 4.3 and Section 4.4 for details).

Related Work. Our model of consumption games is related but incomparable to *energy games* studied in the literature. In energy games both positive and non-positive weights are allowed, but in contrast to consumption games there are no ω-weights. Energy games with single resource type were introduced in [5], and it was shown that the minimal initial credit problem (and also the membership problem for $\mathit{Safe}(s)$) can be solved in exponential time. Further, it follows from the results of [5] that the emptiness problem for $\mathit{Safe}(s)$, which was shown to be equivalent to two-player mean-payoff games [2], lies in $\mathbf{NP} \cap \mathbf{coNP}$.

Games over extended vector addition systems with states (eVASS games), where the weights in transition labels are in $\{-1, 0, 1, \omega\}$, were introduced and studied in [4]. In [4], it was shown that the question whether player $\Box$ has a safe strategy in a given configuration is decidable, and the winning region of player $\Box$ is computable in $(d-1)$-**EXPTIME**, where d is the eVASS dimension, and hence the provided solution is impractical even for very small d's. A closely related model of energy games with multiple resource types (or multi-energy games) was considered in [7]. The minimal initial credit problem (and also the membership problem for $\mathit{Safe}(s)$) for multi-energy games can be reduced to the corresponding problem over eVASS games with an exponential reduction to encode the integer weights into weights $\{-1, 0, 1\}$. Thus the minimal initial credit problem can be solved in d-**EXPTIME**, and the membership problem is

EXPSPACE-hard (the hardness follows from the classical result of Lipton [12]). The emptiness problem for *Safe*(s) is **coNP**-complete for multi-energy games [7]. Thus the complexity of the membership and the minimal initial credit problem for consumption games is much better (it is in **EXPTIME** and **PSPACE**-hard and can be solved in polynomial time for every fixed choice of the parameters) as compared to eVASS games or multi-energy games (**EXPSPACE**-hard and can be solved in d-**EXPTIME**). For eVASS games with fixed dimensions, the problem can be solved in polynomial time for $d = 2$ (see [6]), and it is open whether the complexity can be improved for other constants. Moreover, for the important subclasses of one-player and decreasing consumption games we show much better bounds (polynomial time algorithms for emptiness and optimal complexity bounds for membership in *Safe*(s)).

The paper is organized as follows. After presenting necessary definitions in Section 2, we present our solution to the three algorithmic problems (A)-(C) for general consumption games in Section 3. In Section 4, we concentrate on the two subclasses of decreasing and one-player consumption games and give optimized solutions to some of these problems. Finally, in Section 5 we give a short list of open problems which, in our opinion, address some of the fundamental properties of consumption games that deserve further attention. Due to the lack of space, the proofs are ommited. They can be found in the full version of this paper [3].

2 Definitions

In this paper, the set of all integers is denoted by $\mathbb{Z}$. For a given operator $\bowtie \in \{>, <, \leq, \geq\}$, we use $\mathbb{Z}_{\bowtie 0}$ to denote the set $\{i \in \mathbb{Z} \mid i \bowtie 0\}$, and $\mathbb{Z}^{\omega}_{\bowtie 0}$ to denote the set $\mathbb{Z}_{\bowtie 0} \cup \{\omega\}$, where $\omega \notin \mathbb{Z}$ is a special symbol representing an "infinite amount" with the usual conventions (in particular, $c + \omega = \omega + c = \omega$ and $c < \omega$ for every $c \in \mathbb{Z}$). For example, $\mathbb{Z}_{<0}$ is the set of all negative integers, and $\mathbb{Z}^{\omega}_{<0}$ is the set $\mathbb{Z}_{<0} \cup \{\omega\}$. We use Greek letters $\alpha, \beta, \ldots$ to denote vectors over $\mathbb{Z}_{\bowtie 0}$ or $\mathbb{Z}^{\omega}_{\bowtie 0}$, and $\mathbf{0}$ to denote the vector of zeros. The i-th component of a given α is denoted by $\alpha(i)$. The standard component-wise ordering over vectors is denoted by $\leq$, and we also write $\alpha < \beta$ to indicate that $\alpha(i) < \beta(i)$ for every i.

Let M be a finite or countably infinite alphabet. A *word* over M is a finite or infinite sequence of elements of M. The empty word is denoted by ε, and the set of all finite words over M is denoted by M^*. Sometimes we also use M^+ to denote the set $M^* \setminus \{\varepsilon\}$. The length of a given word w is denoted by $len(w)$, where $len(\varepsilon) = 0$ and the length of an infinite word is ∞. The individual letters in a word w are denoted by $w(0), w(1), \ldots$, and for every infinite word w and every $i \geq 0$ we use w_i to denote the infinite word $w(i), w(i{+}1), \ldots$.

A *transition system* is a pair $\mathcal{T} = (V, \rightarrow)$, where V is a finite or countably infinite set of *vertices* and $\rightarrow \subseteq V \times V$ a *transition relation* such that for every $v \in V$ there is at least one outgoing transition (i.e., a transition of the form $v \rightarrow u$). A *path* in $\mathcal{T}$ is a finite or infinite word w over V such that $w(i) \rightarrow w(i{+}1)$ for every $0 \leq i < len(w)$. We call a finite path a *history* and infinite path a *run*. The sets of all finite paths and all runs in $\mathcal{T}$ are denoted by $FPath(\mathcal{T})$ and $Run(\mathcal{T})$, respectively.

Definition 1. *A* (2-player) game *is a triple* $G = (V, \mapsto, (V_{\Box}, V_{\Diamond}))$ *where* $(V, \mapsto)$ *is a transition system and* $(V_{\Box}, V_{\Diamond})$ *is a partition of* V. *If* $V_{\Diamond} = \emptyset$, *then* G *is a* 1-player game.

A game G is played by two players, $\Box$ and $\Diamond$, who select transitions in the vertices of $V_\Box$ and $V_\Diamond$, respectively. Let $\odot \in \{\Box, \Diamond\}$. A *strategy* for player $\odot$ is a function which to each $wv \in V^*V_\odot$ assigns a state $v' \in V$ such that $v \mapsto v'$. The sets of all strategies for player $\Box$ and player $\Diamond$ are denoted by Σ_G and Π_G (or just by Σ and Π if G is understood), respectively. We say that a strategy τ is *memoryless* if $\tau(wv)$ depends just on the last state v, for every $w \in V^*$. Strategies that are not necessarily memoryless are called *history-dependent*. Note that every initial vertex v and every pair of strategies $(\sigma, \pi) \in \Sigma \times \Pi$ determine a unique infinite path in G initiated in v, which is called a *play* and denoted by $Play_{\sigma,\pi}(v)$.

Definition 2. *Let $d \geq 1$. A* consumption game of dimension d *is a tuple $C = (S, E, (S_\Box, S_\Diamond), L)$ where S is a finite set of* states*, (S, E) is a transition system, $(S_\Box, S_\Diamond)$ is a partition of S, and L is* labelling *which to every $(s,t) \in E$ assigns a vector $\delta = (\delta(1), \ldots, \delta(d))$ such that $\delta(i) \in \mathbb{Z}^\omega_{\leq 0}$ for every $1 \leq i \leq d$. If $s \in S_\Diamond$, we require that $\delta(i) \neq \omega$ for all $1 \leq i \leq d$. We write $s \xrightarrow{\delta} t$ to indicate that $(s,t) \in E$ and $L(s,t) = \delta$.*

We say that C is one-player *if $S_\Diamond = \emptyset$, and* decreasing *if for every $n \geq 1$, every $1 \leq i \leq d$, and every path $s_0 \xrightarrow{\delta_1} s_1 \xrightarrow{\delta_2} \cdots \xrightarrow{\delta_n} s_n$ such that $s_0 = s_n$, there is some $j \leq n$ where $\delta_j(i) \neq 0$.*

Intuitively, if $s \xrightarrow{\delta} t$, then the system modeled by C can move from the state s to the state t so that its resources are consumed/reloaded according to δ. More precisely, if $\delta(i) \leq 0$, then the current load of resource i is decreased by $|\delta(i)|$, and if $\delta(i) = \omega$, then the resource i can be reloaded to an arbitrarily high positive value larger than or equal to the current load. The aim of player $\Box$ is to play so that the resources are never exhausted, i.e., the vector of current loads stays positive in every component. The aim of player $\Diamond$ is to achieve the opposite.

The above intuition is formally captured by defining the associated infinite-state game G_C for C. The vertices of G_C are *configurations* of C, i.e., the elements of $S \times \mathbb{Z}^d_{>0}$ together with a special configuration F (which stands for "fail"). The transition relation $\mapsto$ of G_C is determined as follows:

- $F \mapsto F$.
- For every configuration (s, α) and every transition $s \xrightarrow{\delta} t$ of C such that $\alpha(i) + \delta(i) > 0$ for all $1 \leq i \leq d$, there is a transition $(s, \alpha) \mapsto (t, \alpha+\gamma)$ for every $\gamma \in \mathbb{Z}^d$ such that
  - $\gamma(i) = \delta(i)$ for every $1 \leq i \leq d$ where $\delta(i) \neq \omega$;
  - $\gamma(i) \geq 0$ for every $1 \leq i \leq d$ where $\delta(i) = \omega$.
- If (s, α) is a configuration and $s \xrightarrow{\delta} t$ a transition of C such that $\alpha(i) + \delta(i) \leq 0$ for some $1 \leq i \leq d$, then there is a transition $(s, \alpha) \mapsto F$.
- There are no other transitions.

A strategy σ for player $\Box$ in G_C is *safe* in a configuration (s, α) iff for every strategy π for player $\Diamond$ we have that $Play_{\sigma,\pi}(s, \alpha)$ does *not* visit the configuration F. For every $s \in S$, we use

- $Safe(s)$ to denote the set of all $\alpha \in \mathbb{Z}^d_{>0}$ such that player $\Box$ has a safe strategy in (s, α);
- $Cover(s)$ to denote the set of all $\alpha \in \mathbb{Z}^d_{>0}$ such that player $\Box$ has a safe strategy σ in (s, α) such that for every strategy π for player $\Diamond$ and every configuration (t, β) visited by $Play_{\sigma,\pi}(s, \alpha)$ we have that $\beta \leq \alpha$.

If $\alpha \in \mathit{Safe}(s)$, we say that α is *safe in* s, and if $\alpha \in \mathit{Cover}(s)$, we say that α *covers* s. Obviously, $\mathit{Cover}(s) \subseteq \mathit{Safe}(s)$, and both $\mathit{Safe}(s)$ and $\mathit{Cover}(s)$ are upwards closed w.r.t. component-wise ordering (i.e., if $\alpha \in \mathit{Safe}(s)$ and $\alpha \leq \alpha'$, then $\alpha' \in \mathit{Safe}(s)$). This means that $\mathit{Safe}(s)$ and $\mathit{Cover}(s)$ are fully described by its finitely many minimal elements.

Intuitively, $\mathit{Safe}(s)$ consists of all vectors describing a sufficiently large *initial* amount of all resources needed to perform a safe strategy. Note that during a play, the resources can be reloaded to values that are larger than the initial one. Since physical resources are stored in "tanks" with finite capacity, we need to know what *capacities* of these tanks are sufficient for performing a safe strategy. These sufficient capacities are encoded by the vectors of $\mathit{Cover}(s)$.

3 Algorithms for General Consumption Games

In this section we present a general solution for the three algorithmic problems (A)-(C) given in Section 1.

We start by a simple observation that connects the study of consumption games to a more mature theory of Streett games. A Streett game is a tuple $\mathcal{S} = (V, \mapsto, (V_\Box, V_\Diamond), \mathcal{A})$, where $(V, \mapsto, (V_\Box, V_\Diamond))$ is a 2-player game with finitely many vertices, and $\mathcal{A} = \{(G_1, R_1), \ldots, (G_m, R_m)\}$, where $m \geq 1$ and $G_i, R_i \subseteq \mapsto$ for all $1 \leq i \leq m$, is a Streett (or strong fairness) winning condition (for technical convenience, we consider G_i, R_i as subsets of edges rather than vertices). For an infinite path w in $\mathcal{S}$, let $\inf(w)$ be the set of all edges that are executed infinitely often along w. We say that w *satisfies* $\mathcal{A}$ iff $\inf(w) \cap G_i \neq \emptyset$ implies $\inf(w) \cap R_i \neq \emptyset$ for every $1 \leq i \leq m$. A strategy $\sigma \in \Sigma_\mathcal{S}$ is *winning* in $v \in V$ if for every $\pi \in \Pi_\mathcal{S}$ we have that $\mathit{Play}_{\sigma,\pi}(v)$ satisfies $\mathcal{A}$. The problem whether player $\Box$ has a winning strategy in a vertex $v \in V$ is **coNP**-complete [9], and the problem can be solved in $O(m! \cdot |V|^{m+1})$ time [13].

For the rest of this section, we fix a consumption game $C = (S, E, (S_\Box, S_\Diamond), L)$ of dimension d, and we use ℓ to denote the maximal finite $|\delta(i)|$ such that $1 \leq i \leq d$ and δ is a label of some transition.

Lemma 3. *Let* $\mathcal{S}_C = (S, E, (S_\Box, S_\Diamond), \mathcal{A})$ *be a Streett game where* $\mathcal{A} = \{(G_1, R_1), \ldots, (G_d, R_d)\}$, $G_i = \{(s,t) \in E \mid L(s,t)(i) < 0\}$, *and* $R_i = \{(s,t) \in E \mid L(s,t)(i) = \omega\}$ *for every* $1 \leq i \leq d$. *Then for every* $s \in S$ *the following assertions hold:*

1. *If* $\mathit{Safe}(s) \neq \emptyset$, *then player* $\Box$ *has a winning strategy in* s *in the Streett game* $\mathcal{S}_C$.
2. *If player* $\Box$ *has a winning strategy in* s *in the Streett game* $\mathcal{S}_C$, *then* $(d! \cdot |S| \cdot \ell + 1, \ldots, d! \cdot |S| \cdot \ell + 1) \in \mathit{Safe}(s) \cap \mathit{Cover}(s)$.

An immediate consequence of Lemma 3 is that $\mathit{Safe}(s) = \emptyset$ iff $\mathit{Cover}(s) = \emptyset$. Our next lemma shows that the existence of a winning strategy in Streett games is polynomially reducible to the problem whether $\mathit{Safe}(s) = \emptyset$ in consumption games.

Lemma 4. *Let* $\mathcal{S} = (V, \mapsto, (V_\Box, V_\Diamond), \mathcal{A})$ *be a Streett game where* $\mathcal{A} = \{(G_1, R_1), \ldots, (G_m, R_m)\}$. *Let* $C_\mathcal{S} = (V, \mapsto, (V_\Box, V_\Diamond), L)$ *be a consumption game of dimension* m *where* $L(u,v)(i)$ *is either* -1, ω, *or* 0, *depending on whether* $(u,v) \in G_i$, $(u,v) \in R_i$, *or* $(u,v) \notin G_i \cup R_i$, *respectively. Then for every* $v \in V$ *we have that player* $\Box$ *has a winning strategy in* v *(in* $\mathcal{S}$*) iff* $\mathit{Safe}(v) \neq \emptyset$ *(in* $C_\mathcal{S}$*).*

A direct consequence of Lemma 3 and Lemma 4 is the following:

Theorem 5. *The emptiness problems for Safe(s) and Cover(s) are* **coNP***-complete and solvable in $O(d! \cdot |S|^{d+1})$ time.*

Also observe that if managed to prove that the emptiness problem for *Safe*(s) or *Cover*(s) is fixed-parameter tractable in d for consumption games where ℓ is equal to one (i.e., if we proved that the problem is solvable in time $F(d) \cdot n^{O(1)}$ where n is the size of the game and F a computable function), then due to Lemma 4 we would immediately obtain that the problem whether player $\Box$ has a winning strategy in a given Streett game is also fixed-parameter tractable. That is, we would obtain a solution to one of the long-standing open problems of algorithmic study of graph games.

Now we show how to compute the set of minimal elements of *Safe*(s). A key observation is the following lemma whose proof is non-trivial.

Lemma 6. *For every $s \in S$ and every minimal $\alpha \in Safe(s)$ we have that $\alpha(i) \leq d \cdot \ell \cdot |S|$ for every $1 \leq i \leq d$.*

Observe that Lemma 6 does *not* follow from Lemma 3 (2.). Apart from Lemma 6 providing better bound, Lemma 3 (2.) only says that if *all* resources are loaded enough, then there is a safe strategy. However, we aim at proving a substantially stronger result saying that *no* resource needs to be reloaded to more than $d \cdot \ell \cdot |S|$ *regardless* how large is the current load of other resources.

Intuitively, Lemma 6 is obtained by a somewhat tricky inductive argument where we first consider all resources as being "sufficiently large" and then bound the components one by one. Since a similar technique is also used to compute the minimal elements of *Cover*(s), we briefly introduce the main underlying notions and ideas.

An *abstract load vector* μ is an element of $(\mathbb{Z}^{\omega}_{>0})^d$. The *precision* of μ is the number of components different from ω. The standard componentwise ordering is extended also to abstract load vectors by stipulating that $c < \omega$ for every $c \in \mathbb{Z}$. Given an abstract load vector μ and a vector $\alpha \in (\mathbb{Z}_{>0})^d$, we say that α *matches* μ if $\alpha(j) = \mu(j)$ for all $1 \leq j \leq d$ such that $\mu(j) \neq \omega$. Finally, we say that μ is *compatible* with *Safe*(s) (or *Cover*(s)) if there is some $\alpha \in Safe(s)$ (or $\alpha \in Cover(s)$) that matches μ.

The proof of Lemma 6 is obtained by showing that for every *minimal* abstract load vector μ with precision i compatible with *Safe*(s) we have that $\mu(j) \leq i \cdot \ell \cdot |S|$ for every $1 \leq j \leq d$ such that $\mu(j) \neq \omega$. Since the minimal elements of *Safe*(s) are exactly the minimal abstract vectors of precision d compatible with *Safe*(s), we obtain the desired result. The claim is proven by induction on i. In the induction step, we pick a minimal abstract vector μ with precision i compatible with s, and choose a component j such that $\mu(j) = \omega$. Then we show that if we replace $\mu(j)$ with some k whose value is bounded by $(i + 1) \cdot \ell \cdot |S|$, we yield a minimal compatible abstract vector with precision $i + 1$. The proof of this claim is the very core of the whole argument, and it involves several subtle observations about the structure of minimal abstract load vectors. The details are given in [3].

An important consequence of Lemma 6 is the following:

Corollary 7 (Finite reload property). *If $\alpha \in Safe(s)$ and $\beta(i) = \min\{\alpha(i), d \cdot \ell \cdot |S|\}$ for every $1 \leq i \leq d$, then $\beta \in Safe(s)$.*

Due to Corollary 7, for every minimal $\alpha \in \mathit{Safe}(s)$ there is a safe strategy which never reloads any resource to more than $d \cdot \ell \cdot |S|$. Thus, we can significantly improve the bound of Lemma 3 (2.).

Corollary 8. *If $\mathit{Safe}(s) \neq \emptyset$, then $(d \cdot \ell \cdot |S|, \ldots, d \cdot \ell \cdot |S|) \in \mathit{Safe}(s) \cap \mathit{Cover}(s)$.*

Another consequence of Corollary 7 is that one can reduce the problem of computing the minimal elements of $\mathit{Safe}(s)$ to the problem of determining a winning set in a finite-state 2-player safety game with at most $|S| \cdot d^d \cdot \ell^d \cdot |S|^d + 1$ vertices, which is obtained from C by storing the vector of current resource loads explicitly in the states. Whenever we need to reload some resource, it can be safely reloaded to $d \cdot \ell \cdot |S|$, and we simulate this reload be the corresponding transition. Since the winning set in a safety game with n states and m edges can be computed in time linear in $n + m$ [10, 1], we obtain the following:

Corollary 9. *The sets of all minimal elements of all $\mathit{Safe}(s)$ are computable in time $(d \cdot \ell \cdot |S|)^{O(d)}$.*

The complexity bounds for the algorithmic problems (B) and (C) for $\mathit{Safe}(s)$ are given in our next theorem. The proofs of the presented lower bounds are given in [3].

Theorem 10. *Let $\alpha \in \mathbb{Z}^d_{>0}$ and $s \in S$.*

- *The problem whether $\alpha \in \mathit{Safe}(s)$ is* **PSPACE***-hard and solvable in time $|\alpha| \cdot (d \cdot \ell \cdot |S|)^{O(d)}$, where $|\alpha|$ is the encoding size of α.*
- *The problem whether α is a minimal vector of $\mathit{Safe}(s)$ is* **PSPACE***-hard and solvable in time $|\alpha| \cdot (d \cdot \ell \cdot |S|)^{O(d)}$, where $|\alpha|$ is the encoding size of α.*
- *The set of all minimal vectors of $\mathit{Safe}(s)$ is computable in time $(d \cdot \ell \cdot |S|)^{O(d)}$.*

Now we provide analogous results for $\mathit{Cover}(s)$. Note that deciding the membership to $\mathit{Cover}(s)$ is trivially reducible to the problem of computing the winning region in a finite-state game obtained from C by constraining the vectors of current resource loads by α. Computing the minimal elements of $\mathit{Cover}(s)$ is more problematic. One is tempted to conclude that all components of the minimal vectors for each $\mathit{Cover}(s)$ are bounded by a "small" number, analogously to Lemma 6. In this case, we obtained only the following bound, which is still polynomial for every fixed d and ℓ, but grows *double-exponentially* in d. The question whether this bound can be lowered is left open, and seems to require a deeper insight into the structure of covering vectors.

Lemma 11. *For every $s \in S$ and every minimal $\alpha \in \mathit{Cover}(s)$ we have that $\alpha(i) \leq (d \cdot \ell \cdot |S|)^{d!}$ for every $1 \leq i \leq d$.*

The proof of Lemma 11 is given in [3]. It is based on re-using and modifying some ideas introduced in [4] for general eVASS games. The following theorem sums up the complexity bounds for problems (B) and (C) for $\mathit{Cover}(s)$.

Theorem 12. *Let $\alpha \in \mathbb{Z}^d_{>0}$ and $s \in S$.*

- *The problem whether $\alpha \in \mathit{Cover}(s)$ is* **PSPACE***-hard and solvable in $O(\Lambda^2 \cdot |S|^2)$ time, where $\Lambda = \Pi_{i=1}^{d} \alpha(i)$.*
- *The problem whether α is a minimal element of $\mathit{Cover}(s)$ is* **PSPACE***-hard and solvable in $O(d \cdot \Lambda^2 \cdot |S|^2)$ time, where $\Lambda = \Pi_{i=1}^{d} \alpha(i)$.*
- *The set of all minimal vectors of $\mathit{Cover}(s)$ is computable in $(d \cdot \ell \cdot |S|)^{O(d \cdot d!)}$ time.*

4 Algorithms for One-Player and Decreasing Consumption Games

In this section we present more efficient algorithms for the two subclasses of decreasing and one-player consumption games. Observe that these special classes of games can still retain a rich modeling power. In particular, the decreasing subclass is quite natural as systems that do not decrease some of the resources for a long time most probably stopped working completely (also recall that the game considered in Section 1 is decreasing). One-player consumption games are useful for modeling a large variety of scheduling problems, as it is illustrated in the following example.

Consider the following (a bit idealized) problem of supplying shops with goods such as, e.g., bottles of drinking water. This problem may be described as follows: Imagine a map with c cities connected by roads, n of these cities contain shops to be supplied, k cities contain warehouses with huge amounts of the goods that should be distributed among the shops. The company distributing the goods owns d cars, each car has a bounded capacity. The goal is to distribute the goods from warehouses to all shops in as short time as possible. This situation can be modeled using a one-player consumption game as follows. States would be tuples of the form $(c_1, \ldots, c_d, A)$ where each $c_i \in \{1, \ldots, c\}$ corresponds to the city in which the i-th car is currently located, $A \subseteq \{1, \ldots, n\}$ lists the shops that have already been supplied (initially $A = \emptyset$ and the goal is to reach $A = \{1, \ldots, n\}$). Loads of individual cars and the total time would be modelled by a vector of resources, $(\ell(1), \ldots, \ell(d), t)$, where each $\ell(i)$ models the current load of the i-th car and t models the amount of time which elapsed from the beginning (this resource is steadily decreased until $A = \{1, ..., n\}$). Player $\Box$ chooses where each car should go next. Whenever the i-th car visits a city with a warehouse, the corresponding resource $\ell(i)$ may be reloaded. Whenever the i-th car visits a city containing a shop, player $\Box$ may choose to supply the shop, i.e. decrease the resource $\ell(i)$ of the car by the amount demanded by the shop. Now the last component of a minimal safe configuration indicates how much time is needed to supply all shops. A cover configuration indicates not only how much time is needed but also how large cars are needed to supply all shops. This model can be further extended with an information about the fuel spent by the individual cars, etc.

As in the previous section, we fix a consumption game $C = (S, E, (S_\Box, S_\Diamond), L)$ of dimension d, and we use ℓ to denote the maximal finite $|\delta(i)|$ such that $1 \leq i \leq d$ and δ is a label of some transition. We first establish the complexity of emptiness and membership problem, and then present an algorithm to compute the minimal safe configurations.

4.1 The Emptiness and Membership Problems

We first establish the complexity of the emptiness problem for decreasing games by a polynomial time reduction to *generalized Büchi games*. A generalized Büchi game is a tuple $\mathcal{B} = (V, \mapsto, (V_\Box, V_\Diamond), B)$, where $(V, \mapsto, (V_\Box, V_\Diamond))$ is a 2-player game with finitely many vertices, and $B = \{F_1, \ldots, F_m\}$, where $m \geq 1$ and $F_i \subseteq \mapsto$ for all $1 \leq i \leq m$. We say that infinite path w *satisfies* the generalized Büchi condition defined by B iff $\inf(w) \cap F_i \neq \emptyset$ for every $1 \leq i \leq m$. A strategy $\sigma \in \Sigma_\mathcal{B}$ is *winning* in $v \in V$ if for every $\pi \in \Pi_\mathcal{B}$ we have that the $Play_{\sigma,\pi}(v)$ satisfies the generalized Büchi condition. The

problem whether player $\Box$ has a winning strategy in state s can be decided in polynomial time, with an algorithm of complexity $O(|V| \cdot | \mapsto | \cdot m)$ (see [8]).

We claim that the following holds:

Lemma 13. *If C is a decreasing game, then $Safe(s) \neq \emptyset$ if and only if the player $\Box$ has winning strategy in generalized Büchi game $\mathcal{B}_C = (S, E, (S_\Box, S_\Diamond), \{R_1, \ldots, R_d\})$, where for each $1 \leq i \leq d$ we have $R_i = \{(s,t) \in E \mid L(s,t)(i) = \omega\}$.*

Previous lemma immediately gives us that the emptiness of $Safe(s)$ in decreasing games is decidable in time $O(|S| \cdot |E| \cdot d)$. We now argue that the emptiness of $Safe(s)$ for one-player games can also be achieved in polynomial time. Note that from Lemma 3 we have that $Safe(s) \neq \emptyset$ if and only if player $\Box$ has a winning strategy in state s of one-player Streett game $\mathcal{S}_C$. The problem of deciding the existence of winning strategy in one-player Streett game is exactly the nonemptiness problem for Streett automata that can be solved in time $O((|S| \cdot d + |E|) \cdot \min\{|S|, d\})$ [11].

Theorem 14. *Given a consumption game C and a state s, the emptiness problems of whether $Safe(s) = \emptyset$ and $Cover(s) = \emptyset$ can be decided in time $O(|S| \cdot |E| \cdot d)$ if C is decreasing, and in time $O((|S| \cdot d + |E|) \cdot \min\{|S|, d\})$ if C is a one-player game.*

We now study the complexity of the membership problem for $Safe(s)$. We prove two key lemmas that bound the number of steps before all resources are reloaded. The key idea is to make player $\Box$ reload resources as soon as possible. Formally, we say that a play $Play_{\sigma,\pi}(s,\alpha)$ induced by a sequence of transitions $s_0 \xrightarrow{\delta_1} \cdots \xrightarrow{\delta_k} s_k$ *reloads i-th resource in j-th step* if $\delta_j(i) = \omega$. We first present a lemma for decreasing games and then for one-player games.

Lemma 15. *Consider a decreasing consumption game C and a configuration (s,α) such that $\alpha \in Safe(s)$. There is a safe strategy σ for player $\Box$ in (s,α) such that every $Play_{\sigma,\pi}(s,\alpha)$ reloads all resources in the first $d \cdot |S|$ steps.*

Now let us consider *one-player* games. As player $\Diamond$ has only one trivial strategy, π, we write only $Play_\sigma(s,\alpha)$ instead of $Play_{\sigma,\pi}(s,\alpha)$.

Lemma 16. *Consider a one-player consumption game C and a configuration (s,α) such that $\alpha \in Safe(s)$. There is a safe strategy σ for player $\Box$ in (s,α) such that for the $Play_\sigma(s,\alpha)$ and every $1 \leq i \leq d$ we have that either the i-th resource is reloaded in the first $d \cdot |S|$ steps, or it is never decreased from the $(d \cdot |S| + 1)$-st step on.*

As a consequence of Lemma 15, Lemma 16 and the hardness results presented in [3] we obtain the following:

Theorem 17. *The membership problem of whether $\alpha \in Safe(s)$ is* **NP***-complete for one-player consumption games and* **PSPACE***-complete for decreasing consumption games. The problem whether α is a minimal element of $Safe(s)$ is* **DP***-complete for one-player consumption games and* **PSPACE***-complete for decreasing consumption games.*

4.2 Minimal Safe Configurations and Multi-Distance Reachability

In the rest of the paper we present algorithms for computing the minimal safe configurations in one-player and decreasing consumption games. Both algorithms use the iterative algorithm for *multi-distance reachability problem*, which is described below, as a subprocedure. Although their worst-case complexity is the same as the complexity of generic algorithm from Section 3, we still deem them to be more suitable for practical computation due to some of their properties that we state here in advance:

- The generic algorithm always constructs game of size $(|S| \cdot d \cdot \ell)^{O(d)}$. In contrast, algorithms based on solving multi-distance reachability construct a game whose size is linear in size of C *for every fixed choice of parameter* d.
- The multi-distance reachability algorithms iteratively construct sets of configurations that are safe but may not be minimal before the algorithm stops. Although the time complexity of this iterative computation is $(|S| \cdot d \cdot \ell)^{O(d)}$ at worst, it may be the case that the computation terminates much earlier. Thus, these algorithms have a chance to terminate earlier than in $(|S| \cdot d \cdot \ell)^{O(d)}$ steps (unlike the generic algorithm, where the necessary construction of the "large" safety game always requires this number of steps).
- Moreover, the algorithm for one-player games presented in Section 4.3 decomposes the problem into many parallel subtasks that can be processed independently.

Let $\mathcal{D}$ denote a d-dimensional consumption game with transitions labeled by vectors over $\mathbb{Z}_{\leq 0}$ (i.e. there is no ω in any label). Also denote D the set of states of game $\mathcal{D}$. We say that vector α is a *safe multi-distance* (or just *safe distance*) from state s to state r if there is a strategy σ for player $\Box$ such that for any strategy π for player $\Diamond$ the infinite path $Play_{\sigma,\pi}(s,\alpha)$ visits a configuration of the form (r,β). That is, α is a safe distance from s to r if player $\Box$ can enforce reaching r from s in such a way that the total decrease in resource values is less than α.

We denote by $Safe_{\mathcal{D}}(s,r)$ the set of all safe distances from s to r in $\mathcal{D}$, and by $\lambda_{\mathcal{D}}(s,r)$ the set of all minimal elements of $Safe_{\mathcal{D}}(s,r)$. If $Safe_{\mathcal{D}}(s,r) = \emptyset$, then we set $\lambda_{\mathcal{D}}(s,r) = \{(\infty,\dots,\infty)\}$, where the symbol ∞ is treated accordingly with the usual conventions (for any $c \in \mathbb{Z}$ we have $\infty - c = \infty$, $c < \infty$; we do not use the ω symbol to avoid confusions).

We present a simple fixed-point iterative algorithm which computes the set of minimal safe distances from s to r. Apart from the standard set operations, the algorithm uses the following operations on sets of vectors: for a given set M and a given vector α, the operation $\texttt{min-set}(M)$ returns the set of minimal elements of M, and $M - \alpha$ returns the set $\{\beta - \alpha \mid \beta \in M\}$. Further, given a sequence of sets of vectors $M_1,\dots,M_m$ the operation $\texttt{cwm}(M_1,\dots,M_m)$ returns the set $\{\alpha_1 \vee \cdots \vee \alpha_m \mid \alpha_1 \in M_1,\dots,\alpha_m \in M_m\}$, where each $\alpha_1 \vee \cdots \vee \alpha_m$ denotes a component-wise maximum of the vectors $\alpha_1,\dots,\alpha_m$.

Technically, the algorithm iteratively solves the following optimality equations: for any state q with outgoing transitions $q \xrightarrow{\delta_1} q_1,\dots,q \xrightarrow{\delta_m} q_m$ we have that

$$\lambda_{\mathcal{D}}(q,r) = \begin{cases} \texttt{min-set}\,(\lambda_{\mathcal{D}}(q_1,r) - \delta_1 \cup \cdots \cup \lambda_{\mathcal{D}}(q_m,r) - \delta_m) & \text{if } q \in D_{\Box} \\ \texttt{min-set}\,(\ \texttt{cwm}(\lambda_{\mathcal{D}}(q_1,r) - \delta_1,\dots,\lambda_{\mathcal{D}}(q_m,r) - \delta_m)\ ) & \text{if } q \in D_{\Diamond} \end{cases}$$

The algorithm iteratively computes the k-step approximations of $\lambda_{\mathcal{D}}(q,r)$, which are denoted by $\lambda^k_{\mathcal{D}}(q,r)$. Intuitively, each set $\lambda^k_{\mathcal{D}}(q,r)$ consists of all minimal safe dis-

tances from q to r over all plays with at most k steps. The set $\lambda^0_{\mathcal{D}}(q,r)$ is initialized to $\{(\infty,\ldots,\infty)\}$ for $q \neq r$, and to $\{(1,\ldots,1)\}$ for $q = r$. Each $\lambda^{k+1}_{\mathcal{D}}(q,r)$ is computed from $\lambda^k_{\mathcal{D}}(q,r)$ using the above optimality equations until a fixed point is reached. In [3] we show that this fixed point is the correct solution for the minimal multi-distance problem.

Since the algorithm is based on standard methods, we omit its presentation (which can be found in [3]) and state only the final result. We call *branching degree* of $\mathcal{D}$ the maximal number of transitions outgoing from any state of $\mathcal{D}$.

Theorem 18. *There is an iterative procedure Min-dist$(\mathcal{D}, s, r)$ that correctly computes the set of minimal safe distances from s to r in time $O\left(|D| \cdot a \cdot b \cdot N^2\right)$, where b is the branching degree of $\mathcal{D}$, a is the length of a longest acyclic path in $\mathcal{D}$ and $N = \max_{0\leq k\leq a} |\lambda^k_{\mathcal{D}}(q,r)|$.*

Moreover, the procedure requires at most a iterations to converge to the correct solution and thus the resulting set $\lambda_{\mathcal{D}}(s,r)$ has size at most N. Finally, the number N can be bounded from above by $(a \cdot \ell)^d$.

Note that the complexity of the procedure Min-dist$(\mathcal{D}, s, r)$ crucially depends on parameter N. The bound on N presented in the previous theorem follows from the obvious fact that components of all vectors in $\lambda^k_{\mathcal{D}}(s,r)$ are either all equal to ∞ or are all bounded from above by $k \cdot \ell$. However, for concrete instances the value of N can be substantially smaller. For example, if the consumption game $\mathcal{D}$ models some real-world problem, then it can be expected that the number of k-step minimal distances from states of $\mathcal{D}$ to r is small, because changes in resources are not entirely independent in these models (e.g., action that consumes a large amount of some resource may consume a large amount of some other resources as well). This observation forms the core of our claim that algorithms based on multi-distance reachability may terminate much earlier than the generic algorithm from Section 3.

4.3 Computing *Safe*(s) in One-Player Consumption Games

Now we present an algorithm for computing minimal elements of *Safe*(s) in one-player consumption games. The algorithm computes the solution by solving several instances of minimum multi-distance reachability problem. We assume that all states s with *Safe*$(s) = \emptyset$ were removed from the game. This can be done in polynomial time using the algorithm for emptiness (see Theorem 14).

We denote by $\Pi(d)$ the set of all permutations of the set $\{1,\ldots,d\}$. We view each element of $\Pi(d)$ as a finite sequence $\pi_1 \ldots \pi_d$, e.g., $\Pi(2) = \{12, 21\}$. We use the standard notation π for permutations: confusion with strategies of player $\diamond$ should not arise since $S_\diamond = \emptyset$ in one-player games.

We say that a play $Play_\sigma(s,\alpha)$ *matches* a permutation π if for every $1 \leq i < j \leq d$ the following holds: If the π_j-th resource is reloaded along $Play_\sigma(s,\alpha)$, then the π_i-th resource is also reloaded along this play and the first reload of π_i-th resource occurs before or at the same time as the first reload of π_j-th resource. A configuration (s,α) matches π if there is a strategy σ that is safe in (s,α) and $Play_\sigma(s,\alpha)$ matches π. We denote by *Safe*(s,π) the set of all vectors α such that (s,α) matches π. Note that *Safe*$(s) = \bigcup_{\pi\in\Pi(d)}$ *Safe*(s,π).

As indicated by the above equality, computation of safe configurations in C reduces to the problem of computing, for every permutation π, safe configurations that match π. The latter problem, in turn, easily reduces to the problem of computing safe multi-distances in specific one-player consumption games $C(\pi)$. Intuitively, each game $C(\pi)$ simulates the game C where the resources are forced to be reloaded in the order specified by π. So the states of each $C(\pi)$ are pairs (s, k) where s corresponds to the current state of the original game and k indicates that the first k resources, in the permutation π, have already been reloaded. Now the crucial point is that if the first k resources have been reloaded when some configuration $c = (s, \beta)$ of the original game is visited, and there is a safe strategy in c which does not decrease any of the resources with the index greater than k, then we may safely conclude that the *initial* configuration is safe. So, in such a case we put a transition from the state (s, k) of $C(\pi)$ to a distinguished target state r (whether or not to put in such a transition can be decided in polynomial time due to Theorem 14). Other transitions of $C(\pi)$ correspond to transitions of C except that they have to update the information about already reloaded resources, cannot skip any resource in the permutation (such transitions are removed), and the components indexed by $\pi_1, \ldots, \pi_k$ are substituted with 0 in transitions incoming to states of the form (q, k) (since already reloaded resources become unimportant as indicated by the above observation).

A complete construction of $C(\pi)$ is presented in [3] as a part of a formal proof of the following theorem:

Theorem 19. *For every permutation π there is a polynomial time constructible consumption game $C(\pi)$ of size $O(|S| \cdot d)$ and branching degree $O(|S|)$ such that for every vector α we have that $\alpha \in Safe(s, \pi)$ in C iff α is a safe distance from $(s, 0)$ to r in $C(\pi)$.*

By the previous theorem, every minimal element of $Safe(s)$ is an element of $\lambda_{C(\pi)}((s, 0), r)$ for at least one permutation π. Our algorithm examines all permutations $\pi \in \Pi(d)$, and for every permutation it constructs game $C(\pi)$ and computes $\lambda_{C(\pi)}((s, 0), r)$ using the procedure Min-dist from Theorem 18. The algorithm also stores the set of all minimal vectors that appear in some $\lambda_{C(\pi)}((s, 0), r)$. In this way, the algorithm eventually finds all minimal elements of $Safe(s)$. The pseudocode of the algorithm is presented in [3].

From complexity bounds of Theorems 14 and 18 we obtain that the worst case running time of this algorithm is $d! \cdot (|S| \cdot \ell \cdot d)^{O(d)}$. In contrast with the generic algorithm of Section 3, that constructs an exponentially large safety game, the algorithm of this section computes $d!$ "small" instances of the minimal multi-distance reachability problem. We can solve many of these instances in parallel. Moreover, as argued in previous section, each call of Min-dist$(C(\pi), (s, 0), r)$ may have much better running time than the worst-case upper bound suggests.

4.4 Computing *Safe*(s) in Decreasing Consumption Games

We now turn our attention to computing minimal elements of $Safe(s)$ in decreasing games. The main idea is again to reduce this task to the computation of minimal multi-distances in certain consumption game. We again assume that states with $Safe(s) = \emptyset$ were removed from the game.

The core of the reduction is the following observation: if C is decreasing, then $\alpha \in Safe(s)$ iff player $\Box$ is able to ensure that the play satisfies these two conditions: all resources are reloaded somewhere along the play; and the i-th resource is reloaded for the first time before it is decreased by at least $\alpha(i)$, for every $1 \leq i \leq d$. Now if we augment the states of C with an information about which resources have been reloaded at least once in previous steps, then the objective of player $\Box$ is actually to reach a state which tells us that all resources were reloaded at least once.

So the algorithm constructs a game $\widehat{C}$ by augmenting states of C with an information about which resources have been reloaded at least once, and by substituting updates of already reloaded resources (i.e., the corresponding components of the labels) with zeros. Note that the construction of $\widehat{C}$ closely resembles the construction of games $C(\pi)$ from the previous section. However, in two-player case we cannot fix an order in which resources are to be reloaded, because the optimal order depends on a strategy chosen by player $\Diamond$. Thus, we need to remember exactly which resources have been reloaded in the past (we only need to remember the set of resources that have been reloaded, but not the order in which they were reloaded).

The formal construction of $\widehat{C}$ can be found in [3] along with a proof of the following theorem.

Theorem 20. *There is a consumption game $\widehat{C}$ of size $O(2^d \cdot |S|)$, branching degree $O(S)$ and with maximal acyclic path of length $O(|S| \cdot d)$, with the following properties: $\widehat{C}$ is constructible in time $O(2^d \cdot (|S| + |E|))$ and for every vector α we have $\alpha \in Safe(s)$ in C iff α is a safe distance from $(s, \emptyset)$ to r in $\widehat{C}$.*

The previous theorem shows that we can find minimal elements of $Safe(s)$ with a single call of procedure Min-dist$(\widehat{C}, (s, \emptyset), r)$. Straightforward complexity analysis reveals that the worst-case running time of this algorithm is $(|S| \cdot d \cdot \ell)^{O(d)}$. However, the game $\widehat{C}$ constructed during the computation is still smaller than the safety game constructed by the generic algorithm of Section 3. Moreover, the length of the longest acyclic path in $\widehat{C}$ is bounded by $|S| \cdot d$, so the procedure Min-dist does not have to perform many iterations, despite the exponential size of $\widehat{C}$. Finally, let us once again recall that the procedure Min-dist$(\widehat{C}, (s, \emptyset), r)$ may actually require much less than $(|S| \cdot d \cdot \ell)^{O(d)}$ steps.

5 Conclusions

As it is witnessed by the results presented in previous sections, consumption games represent a convenient trade-off between expressive power and computational tractability. The presented theory obviously needs further development before it is implemented in working software tools. Some of the issues are not yet fully understood, and there are also other well-motivated problems about consumption games which were not considered in this paper. The list of important open problems includes the following:

- Improve the complexity of algorithms for $Cover(s)$. This requires further insights into the structure of these sets.
- Find efficient controller synthesis algorithms for objectives that combine safety with other linear-time properties. That is, decide whether player $\Box$ has a safe strategy such that a play satisfies a given LTL property no matter what player $\Diamond$ does.

- Find algorithms for more complicated optimization problems, where the individual resources may have different priorities. For example, it may happen that fuel consumption or the price of batteries with large capacity are much more important than the time spent, and in that case we might want to optimize some weight function over the tuple of all resources. It may happen (and we have concrete examples) that some of these problems are actually solvable even more efficiently than the general ones where all resources are treated equally w.r.t. their importance.

The above list is surely incomplete. The problem of optimal resource consumption is rather generic and appears in many different contexts, which may generate other interesting questions about consumption games.

References

1. Beeri, C.: On the membership problem for functional and multivalued dependencies in relational databases. ACM Trans. on Database Systems 5, 241–259 (1980)
2. Bouyer, P., Fahrenberg, U., Larsen, K.G., Markey, N.: Timed automata with observers under energy constraints. In: Johansson, K., Yi, W. (eds.) HSCC 2010. pp. 61–70. ACM (2010)
3. Brázdil, T., Chatterjee, K., Kučera, A., Novotný, P.: Efficient controller synthesis for consumption games with multiple resource types. CoRR abs/1202.0796 (2012)
4. Brázdil, T., Jančar, P., Kučera, A.: Reachability games on extended vector addition systems with states. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010 (2). LNCS, vol. 6199, pp. 478–489. Springer, Heidelberg (2010)
5. Chakrabarti, A., de Alfaro, L., Henzinger, T.A., Stoelinga, M.: Resource interfaces. In: Alur, R., Lee, I. (eds.) EMSOFT 2003. LNCS, vol. 2855, pp. 117–133. Springer, Heidelberg (2003)
6. Chaloupka, J.: Z-reachability problem for games on 2-dimensional vector addition systems with states is in P. In: Kučera, A., Potapov, I. (eds.) RP 2010. LNCS, vol. 6227, pp. 104–119. Springer, Heidelberg (2010)
7. Chatterjee, K., Doyen, L., Henzinger, T.A., Raskin, J.F.: Generalized mean-payoff and energy games. In: Lodaya, K., Mahajan, M. (eds.) Proc. of FSTTCS 2010. LIPIcs, vol. 8, pp. 505–516. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2010)
8. Dziembowski, S., Jurdzinski, M., Walukiewicz, I.: How much memory is needed to win infinite games? In: LICS'97. pp. 99–110. IEEE (1997)
9. Emerson, E., Jutla, C.: The complexity of tree automata and logics of programs. In: FOCS'88. pp. 328–337. IEEE (1988)
10. Immerman, N.: Number of quantifiers is better than number of tape cells. Journal of Computer and System Sciences 22(3), 384–406 (1981)
11. Kurshan, R.: Computer-aided verification of coordinating processes: the automata-theoretic approach. Princeton University Press (1994)
12. Lipton, R.: The reachability problem requires exponential space (1976), technical report 62, Yale University
13. Piterman, N., Pnueli, A.: Faster solution of Rabin and Streett games. In: LICS'06. pp. 275–284. IEEE (2006)