Stability in Graphs and Games*

Tomáš Brázdil[1], Vojtěch Forejt[2], Antonín Kučera[3], and Petr Novotný[4]

1 **Faculty of Informatics, Masaryk University, Czech Republic**
2 **Department of Computer Science, University of Oxford, UK**
3 **Faculty of Informatics, Masaryk University, Czech Republic**
4 **IST Austria, Klosterneuburg, Austria**

Abstract

We study graphs and two-player games in which rewards are assigned to states, and the goal of the players is to satisfy or dissatisfy certain property of the generated outcome, given as a mean payoff property. Since the notion of mean-payoff does not reflect possible fluctuations from the mean-payoff along a run, we propose definitions and algorithms for capturing the *stability* of the system, and give algorithms for deciding if a given mean payoff and stability objective can be ensured in the system.

1998 ACM Subject Classification F.1.1 Automata, D.2.4 Formal methods

Keywords and phrases Games, Stability, Mean-Payoff, Window Objectives

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2016.10

1 Introduction

Finite-state graphs and games are used in formal verification as foundational models that capture behaviours of systems with controllable decisions and possibly with an adversarial environment. States correspond to possible configurations of a system, and edges describe how configurations can change. In a game, each state is owned by one of two players, and the player owning the state decides what edge will be taken. A graph is a game where only one of the players is present. When the choice of the edges is resolved, we obtain an *outcome* which is an infinite sequence of states and edges describing the execution of the system.

The long-run average performance of a run is measured by the associated *mean-payoff*, which is the limit average reward per visited state along the run. It is well known that memoryless deterministic strategies suffice to optimize the mean payoff, and the corresponding decision problem is in $\mathsf{NP} \cap \mathsf{coNP}$ for games and in P for graphs. If the rewards assigned to the states are multi-dimensional vectors of numbers, then the problem becomes coNP-hard for games [21].

Although the mean payoff provides an important metric for the average behaviour of the system, by definition it neglects all information about the fluctuations from the mean payoff along the run. For example, a "fully stable" run where the associated sequence of rewards is $1, 1, 1, 1, \ldots$ has the same mean payoff (equal to 1) as a run producing $n, 0, 0, \ldots, n, 0, 0, \ldots$ where a state with the reward n is visited once in n transitions. In many situations, the first run is much more desirable that the second one. Consider, e.g., a video streaming

* The work has been supported by the Czech Science Foundation, grant No. 15-17564S, by EPSRC grant EP/M023656/1, and by the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement no [291734].

application which needs to achieve a sufficiently high bit-rate (a long-run average number of bits delivered per second) but, in addition, a sufficient level of "stability" to prevent buffer underflows and overflows which would cause data loss and stuttering. Similar problems appear also in other contexts. For example, production lines should be not only efficient (i.e., produce the number of items per time unit as high as possible), but also "stable" so that the available stores are not overfull and there is no "periodic shortage" of the produced items. A food production system should not only produce a sufficiently large amount of food per day on average, but also a certain amount of food daily. These and similar problems motivate the search for a suitable formal notion capturing the intuitive understanding of "stability", and developing algorithms that can optimize the performance under given stability constraints. That is, we are still seeking for a strategy optimizing the mean payoff, but the search space is restricted to the subset of all strategies that achieve a given stability constraint.

Since the mean-payoff $mp(\lambda)$ of a given run λ can be seen as the average reward of a state visited along λ, a natural idea is to define the stability of λ as *sample variance* of the reward assigned to a state along λ. More precisely, let r_i be the reward of the i-th state visited by λ, and let $c_0, c_1, \ldots$ be an infinite sequence where $c_i = (mp(\lambda) - r_i)^2$. The *long-run variance* of the reward assigned to a state along λ, denoted by $va(\lambda)$, is the limit-average of $c_0, c_1, \ldots$ The notion of long-run variance has been introduced and studied for Markov decision processes in [5]. If $va(\lambda)$ is small, then large fluctuations from $mp(\lambda)$ are rare. Hence, if we require that a strategy should optimize mean-payoff while keeping the long-run variance below a given threshold, we in fact impose a *soft* stability constraint which guarantees that "bad things do not happen too often". This may or may not be sufficient.

In this paper, we are particularly interested in formalizing *hard* stability constraints which guarantee that "bad things never happen". We introduce a new type of objectives called *window-stability multi-objectives* that can express a rich set of hard stability constraints, and we show that the set of *all* strategies that achieve a given window-stability multi-objective can be characterized by an effectively constructible *finite-memory permissive strategy scheme*. From this we obtain a meta-theorem saying that if an objective (such as mean-payoff optimization) is solvable for finite-state games (or graphs), then the same objective is solvable also under a given window-stability multi-objective constraint. We also provide the associated upper and lower complexity bounds demonstrating that the time complexity of our algorithms is "essentially optimal".

More specifically, a single *window-stability objective* (inspired by [8], see Related work below) is specified by a window length $W \geq 1$, a checkpoint distance $D \geq 1$, and two bounds μ and ν. For technical reasons, we assume that D divides W. Every run $\lambda = s_0, s_1, s_2, \ldots$ then contains infinitely many *checkpoints* $s_0, s_D, s_{2D}, s_{3D}, \ldots$ The objective requires that the average reward assigned to the states $s_j, \ldots, s_{j+W-1}$, where s_j is a checkpoint, is between μ and ν. In other words, the "local mean-payoff" computed for the states fitting into a window of length W starting at a checkpoint must be within the "acceptable" bounds μ and ν. The role of W is clear, and the intuition behind D is the following. Since D divides W, there are two extreme cases: $D = 1$ and $D = W$. For $D = 1$, the objective closely resembles the standard "sliding window" model over data streams [13]; we require that the local mean-payoff stays within the acceptable bounds "continuously", like the "local bit-rate" in video-streaming. If $D = W$, then the windows do not overlap at all. This is useful in situations when we wish to guarantee some time-bounded periodic progress. For example, if we wish to say that the number of items produced per day stays within given bounds, we set W so that it represents the (discrete) time of one day and put $D = W$. However, there can be also scenarios when we wish to check the local mean-payoff more often then

once during W transitions, but not "completely continuously". In these cases, we set D to some other divisor of W. A *window-stability multi-objective* is a finite conjunction of single window-stability objectives, each with dedicated rewards and parameters. Hence, window-stability multi-objectives allow for capturing more delicate stability requirements such as "a factory should produce between 1500 and 1800 gadgets every week, and in addition, within every one-hour period at least 50 computer chips are produced, and in addition, the total amount of waste produced in 12 consecutive hours does not exceed 500 kg."

Our contribution. The results of this paper can be summarized as follows:

(A) We introduce the concept of window-stability multi-objectives.

(B) We show that there is an algorithm which inputs a game G and a window-stability multi-objective Δ, and outputs a *finite-state permissive strategy scheme* for Δ and G. A finite-state permissive strategy scheme for Δ and G is a finite-state automaton Γ which reads the history of a play and constraints the moves of Player $\Box$ (who aims at satisfying Δ) so that a strategy σ achieves Δ in G iff σ is admitted by Γ. Hence, we can also compute a synchronized product $\mathsf{G} \times \Gamma$ which is another game where the set of *all* strategies for Player $\Box$ precisely represents the set of all strategies for Player $\Box$ in G which achieve the objective Δ. Consequently, *any* objective of the form $\Delta \wedge \Psi$ can be solved for G by solving the objective Ψ for $\mathsf{G} \times \Gamma$. In particular, this is applicable to mean-payoff objectives, and thus we solve the problem of optimizing the mean-payoff under a given window-stability multi-objective constraint. We also analyze the time complexity of these algorithms, which reveals that the crucial parameter which negatively influences the time complexity is the number of checkpoints in a window (i.e., W/D).

(C) We complement the upper complexity bounds of the previous item by lower complexity bounds that indicate that the time complexity of our algorithms is "essentially optimal". Some of these results follow immediately from existing works [21, 8]. The main contribution is the result which says that solving a (single) window-stability objective is PSPACE-hard for games and NP-hard for graphs, even if all numerical parameters (W, D, μ, ν, and the rewards) are encoded in *unary*. The proof is based on novel techniques and reveals that the number of checkpoints in a window (i.e., W/D) is a crucial parameter which makes the problem computationally hard. The window stability objective constructed in the proof satisfies $D = 1$, and the tight window overlapping is used to enforce a certain consistency in Player $\Box$ strategies.

(D) For variance-stability, we argue that while it is natural in terms of using standard mathematical definitions, it does not prevent unstable behaviours. In particular, we show that the variance-stability objective may demand an infinite-memory strategy which switches between two completely different modes of behaviour with smaller and smaller frequency. We also show that the associated variance-stability problem with single-dimensional rewards is in NP for graphs. For this we use some of the results from [5] where the variance-stability is studied in the context of Markov decision processes. The main difficulty is a translation from randomized stochastic-update strategies used in [5] to deterministic strategies.

Related work. Multi-dimensional mean-payoff games were studied in [21], where it was shown that the lim-inf problem, relevant to our setting, is coNP-hard. Further, [11] studies memory requirements for the objectives, and [20] shows that for a "robust" extension (where Boolean combinations of bounds on the resulting vector of mean-payoffs are allowed) the

problem becomes undecidable. Games with quantitative objectives in which both lower and upper bound on the target value of mean-payoff is given were studied in [15]. We differ from these approaches by requiring the "interval" bounds to be satisfied within finite windows, making our techniques and results very different.

As discussed above, we rely on the concept of *windows*, which was in the synthesis setting studied in [8] (see also [14]), as a conservative approximation of the standard mean-payoff objective. More concretely, the objectives in [8] are specified by a maximal window length W and a threshold t. The task is to find a strategy that achieves the following property of runs: a run can be partitioned into contiguous windows of length *at most* W such that in each window, the reward accumulated inside the window divided by the window length is at least t. The objective ensures a local progress in accumulating the reward, and it was not motivated by capturing stability constraints. The fundamental difference between our window-stability approach and windows in [8] is that in the latter one can easily get rid of windows overlapping due to so called *inductive window property*, which does not hold under stricter stability constraints. This results in different computational problems, as witnessed by the fact that our PSPACE lower bound discussed in the point (C) above does not (most likely) carry over to the setting of [8], where a similar-looking decision problem is in P.

The notion of finite-state permissive strategy scheme is based on the concept of *permissive strategies* [1] and multi-strategies [4, 3].

The notion of long-run variance has been introduced and studied for Markov decision processes in [5]. Since we consider deterministic strategies, none of our results is a special case of [5], and we have to overcome new difficulties as it is explained in Section 4.

More generally, our paper fits into an active field of multi-objective strategy synthesis, where some objectives capture the "hard" constraints and the other "soft", often quantitative, objectives. Examples of recent results in this area include [2], where a 2-EXPTIME algorithm is given for the synthesis of combined LTL and mean-payoff objectives, [9], where a combination of parity and mean-payoff performance objectives is studied, or [10], where the controlling player must satisfy a given ω-regular objective while allowing the adversary to satisfy another "environmental" objective.

2 Preliminaries

We use $\mathbb{N}$, $\mathbb{N}_0$, and $\mathbb{Q}$ to denote the sets of positive integers, non-negative integers, and rationals, respectively. Given a set M, we use M^* to denote the set of all finite sequences (words) over M, including the empty sequence. For a vector $\vec{v} = (v_1, \ldots, v_k)$ of numbers and a non-zero number a, we use $\vec{v}[i]$ for v_i, and $\vec{v}/a$ for the vector given by $(\vec{v}/a)[i] = \vec{v}[i]/a$.

A *game* is a tuple $\mathsf{G} = (S, (S_\Box, S_\Diamond), E)$ where S is a non-empty set of *states*, $(S_\Box, S_\Diamond)$ is a partition of S into two subsets controlled by Player $\Box$ and Player $\Diamond$, respectively, and $E \subseteq S \times S$ are the *edges* of the game such that for every $s \in S$ there is at least one edge $(s,t) \in E$. A *graph* is a game such that $S_\Diamond = \emptyset$. A *run* in G is an infinite path in the underlying directed graph of G. An *objective* Φ is a Borel property[1] of runs. Note that the class of all objectives is closed under conjunction.

A *strategy* for player $\odot$, where $\odot \in \{\Box, \Diamond\}$ is a function $\tau : S^* S_\odot \to S$ satisfying that $(s, \tau(hs)) \in E$ for all $s \in S_\odot$ and $h \in S^*$. The sets of all strategies of Player $\Box$ and Player $\Diamond$ are denoted by Σ_G and Π_G, respectively. When G is understood, we write just Σ and Π.

[1] Recall that the set of all runs can be given the standard Cantor topology. A property is Borel if the set of all runs satisfying the property belongs to the σ-algebra generated by all open sets in this topology.

A pair of strategies $(\sigma, \pi) \in \Sigma \times \Pi$ together with an initial state s induce a unique run $outcome_s^{\sigma,\pi}$ in the standard way. We say that a strategy $\sigma \in \Sigma$ *achieves* an objective Φ in a state s if $outcome_s^{\sigma,\pi}$ satisfies Φ for every $\pi \in \Pi$. The set of all $\sigma \in \Sigma$ that achieve Φ in s is denoted by $\Sigma^{\Phi}(s)$. An objective Φ is *solvable* for a given subclass $\mathcal{G}$ of finite-state games if there is an algorithm which inputs $\mathsf{G} \in \mathcal{G}$ and its state s, and decides whether $\Sigma^{\Phi}(s) = \emptyset$. If $\Sigma^{\Phi}(s) \neq \emptyset$, then the algorithm also outputs a (finite description of) $\sigma \in \Sigma^{\Phi}(s)$.

We often consider strategies of Player $\Box$ tailored for a specific initial state. A finite sequence of states $s_0, \ldots, s_n$ is *consistent* with a given $\sigma \in \Sigma$ if $s_0, \ldots, s_n$ is a finite path in the graph of G, and $\sigma(s_0, \ldots, s_i) = s_{i+1}$ for every $0 \leq i < n$ where $s_i \in V_{\Box}$. Given $\sigma, \sigma' \in \Sigma$ and $s \in S$, we say that σ and σ' are *s-equivalent*, written $\sigma \equiv_s \sigma'$, if σ and σ' agree on all finite sequences of states initiated in s that are consistent with σ. Note that if $\sigma \equiv_s \sigma'$, then $outcome_s^{\sigma,\pi} = outcome_s^{\sigma',\pi}$ for every $\pi \in \Pi$.

A *reward function* $\varrho : S \to \mathbb{N}_0^k$, where $k \in \mathbb{N}$, assigns non-negative integer vectors to the states of G. We use dim_{ϱ} to denote the dimension k of ϱ, and max_{ϱ} to denote the maximal number employed by ϱ, i.e., $max_{\varrho} = \max\{\varrho(s)[i] \mid 1 \leq i \leq k, s \in S\}$. An objective is *reward-based* if its defining property depends just on the sequence of rewards assigned to the states visited by a run. For every run $\lambda = s_0, s_1, \ldots$ of G, let $mp_{\varrho}(\lambda) = \liminf_{n\to\infty} \frac{1}{n+1} \sum_{i=0}^{n} \varrho(s_i)$ be the *mean payoff* of λ, where the $\liminf_{n\to\infty}$ is taken component-wise. A *mean-payoff* objective is a pair (ϱ, b), where $\varrho : S \to \mathbb{N}_0^k$ is a reward function and $b \in \mathbb{Q}^k$. A run λ satisfies a mean-payoff objective (ϱ, b) if $mp_{\varrho}(\lambda) \geq b$.

Similarly, the *long-run variance of the reward* of a run λ is defined by $va_{\varrho}(\lambda) = \limsup_{n\to\infty} \frac{1}{n+1} \sum_{i=0}^{n} (\varrho(s_i) - mp(\lambda))^2$; intuitively, the long-run variance is a limit superior of sample variances where the samples represent longer and longer run prefixes. A *variance-stability* objective is a triple (ϱ, b, c), where $\varrho : S \to \mathbb{N}_0^k$ is a reward function and $b, c \in \mathbb{Q}^k$. A run λ satisfies a variance-stability objective (ϱ, b, c) if $mp_{\varrho}(\lambda) \geq b$ and $va_{\varrho}(\lambda) \leq c$.

Let $W \in \mathbb{N}$ be a *window size* and $D \in \mathbb{N}$ a *checkpoint distance* such that D divides W. For every $\ell \in \mathbb{N}_0$, the *local mean payoff at the ℓ^{th} checkpoint in a run λ* is defined by $lmp_{W,D,\varrho,\ell}(\lambda) = \frac{1}{W} \sum_{i=0}^{W-1} \varrho(s_{\ell \cdot D + i})$. Thus, every run λ determines the associated infinite sequence $lmp_{W,D,\varrho,0}(\lambda), lmp_{W,D,\varrho,1}(\lambda), lmp_{W,D,\varrho,2}(\lambda), \ldots$ of local mean payoffs. A *window-stability* objective is a tuple $\Phi = (W, D, \varrho, \mu, \nu)$, where $W, D \in \mathbb{N}$ such that D divides W, $\varrho : S \to \mathbb{N}_0^k$ is a reward function, and $\mu, \nu \in \mathbb{Q}^k$. A run λ satisfies Φ if, for all $\ell \in \mathbb{N}$, we have that $\mu \leq lmp_{W,D,\varrho,\ell}(\lambda) \leq \nu$. A *window-stability multi-objective* is a finite conjunction of window-stability objectives.

In this paper, we study the solvability of variance-stability objectives, window-stability multi-objectives, and objectives of the form $\Delta \wedge \Psi$ where Δ is a window-stability multi-objective and Ψ a mean-payoff objective.

3 The Window-Stability Multi-Objectives

This section is devoted to the window-stability multi-objectives and objectives of the form $\Delta \wedge \Psi$, where Δ is a window-stability multi-objective. In Section 3.1, we show how to solve these objectives for finite-state games, and we derive the corresponding upper complexity bounds. The crucial parameter which makes the problem computationally hard is the number of checkpoints in a window. In Section 3.2, we show that this blowup is unavoidable assuming the expected relationship among the basic complexity classes.

3.1 Solving Games with Window-Stability Multi-Objectives

We start by recalling the concept of *most permissive strategies* which was introduced in [1]. Technically, we define *permissive strategy schemes* which suit better our needs, but the underlying idea is the same.

▶ **Definition 1.** Let $\mathsf{G} = (S, (S_\Box, S_\Diamond), E)$ be a game. A *(finite-memory) strategy scheme* for G is a tuple $\Gamma = (\mathit{Mem}, \mathit{Up}, \mathit{Const}, \mathit{Init})$, where $\mathit{Mem} \neq \emptyset$ is a finite set of *memory elements*, $\mathit{Up} : S \times \mathit{Mem} \to \mathit{Mem}$ is a *memory update* function, $\mathit{Const} : S_\Box \times \mathit{Mem} \to 2^S$ is a *constrainer* such that $\mathit{Const}(s, m) \subseteq \{s' \in S \mid (s, s') \in E\}$, and $\mathit{Init} : S \rightharpoonup M$ is a partial function assigning initial memory elements to some states of S.

We require[2] that $\mathit{Const}(s, m) \neq \emptyset$ for all $(s, m) \in \mathit{Reach}(\mathit{Init})$ such that $s \in S_\Box$. Here, $\mathit{Reach}(\mathit{Init})$ is the least fixed-point of $\mathcal{F} : 2^{S \times \mathit{Mem}} \to 2^{S \times \mathit{Mem}}$ where for all Ω the set $\mathcal{F}(\Omega)$ consists of all (s', m') such that either $(s', m') \in \mathit{Init}$, or there is some $(s'', m'') \in \Omega$ such that $(s'', s') \in E$ and $\mathit{Up}(s'', m'') = m'$; if $s'' \in S_\Box$, we further require $s' \in \mathit{Const}(s'', m'')$. ◀

We say that Γ is *memoryless* if the set Mem is a singleton. Every strategy scheme $\Gamma = (\mathit{Mem}, \mathit{Up}, \mathit{Const}, \mathit{Init})$ for a game $\mathsf{G} = (S, (S_\Box, S_\Diamond), E)$ determines a game $\mathsf{G}_\Gamma = (S \times \mathit{Mem}, (S_\Box \times \mathit{Mem}, S_\Diamond \times \mathit{Mem}), F)$, where

- for every $(s, m) \in S_\Diamond \times \mathit{Mem}$, $((s, m), (s', m')) \in F$ iff $\mathit{Up}(s, m) = m'$ and $(s, s') \in E$;
- for every $(s, m) \in S_\Box \times \mathit{Mem}$ where $\mathit{Const}(s, m) \neq \emptyset$, we have that $((s, m), (s', m')) \in F$ iff $\mathit{Up}(s, m) = m'$ and $(s, s') \in \mathit{Const}(s, m)$;
- for every $(s, m) \in S_\Box \times \mathit{Mem}$ where $\mathit{Const}(s, m) = \emptyset$, we have that $((s, m), (s', m')) \in F$ iff $s = s'$ and $m = m'$.

A strategy $\sigma \in \Sigma_\mathsf{G}$ is *admitted* by Γ in a given $s \in S$ if $\mathit{Init}(s) \neq \bot$ and for every finite path $s_0, \ldots, s_n$ in G initiated in s which is consistent with σ there is a finite path $(s_0, m_0), \ldots, (s_n, m_n)$ in G_Γ such that $m_0 = \mathit{Init}(s_0)$ and $s_{i+1} \in \mathit{Const}(s_i, m_i)$ for all $0 \leq i < n$ where $s_i \in S_\Box$. Observe that if σ is admitted by Γ in s, then σ naturally induces a strategy $\tau[\sigma, s] \in \Sigma_{\mathsf{G}_\Gamma}$ which is unique up to $\equiv_{(s_0, m_0)}$. Conversely, every $\tau \in \Sigma_{\mathsf{G}_\Gamma}$ and every $s \in S$ where $\mathit{Init}(s) \neq \bot$ induce a strategy $\sigma[\tau, s] \in \Sigma_\mathsf{G}$ such that, for every finite path $(s_0, m_0), \ldots, (s_n, m_n)$ initiated in $(s, \mathit{Init}(s))$ which is consistent with τ, we have that $\sigma[\tau, s](s_0, \ldots, s_n) = s_{n+1}$ iff $\tau((s_0, m_0), \ldots, (s_n, m_n)) = (s_{n+1}, m_{n+1})$. Note that $\sigma[\tau, s]$ is determined uniquely up to $\equiv_s$.

▶ **Definition 2.** Let G be a game, Γ a strategy scheme for G, $\Lambda_\mathsf{G} \subseteq \Sigma_\mathsf{G}$, $\Lambda_{\mathsf{G}_\Gamma} \subseteq \Sigma_{\mathsf{G}_\Gamma}$, and $s \in S$. We write $\Lambda_\mathsf{G} \approx_s \Lambda_{\mathsf{G}_\Gamma}$ if the following conditions are satisfied:

- Every $\sigma \in \Lambda_\mathsf{G}$ is admitted by Γ in s, and there is $\tau \in \Lambda_{\mathsf{G}_\Gamma}$ such that $\tau[\sigma, s] \equiv_{(s, \mathit{Init}(s))} \tau$.
- For every $\tau \in \Lambda_{\mathsf{G}_\Gamma}$ there is $\sigma \in \Lambda_\mathsf{G}$ such that $\sigma[\tau, s] \equiv_s \sigma$.

Further, we say that Γ is *permissive* for an objective Φ if $\Sigma^\Phi_\mathsf{G}(s) \approx_s \Sigma_{\mathsf{G}_\Gamma}(s)$ for all $s \in S$, where $\Sigma_{\mathsf{G}_\Gamma}(s)$ is either $\emptyset$ or $\Sigma_{\mathsf{G}_\Gamma}$, depending on whether $\mathit{Init}(s) = \bot$ or not, respectively.

The next proposition follows immediately.

▶ **Proposition 3.** Let G be a game, Φ, Ψ objectives, and Γ a strategy scheme permissive for Φ. Then, for every $s \in S$ we have that $\Sigma^{\Phi \wedge \Psi}_\mathsf{G}(s) \approx_s \Sigma^\Psi_{\mathsf{G}_\Gamma}(s)$.

Another simple but useful observation is that the class of objectives for which a permissive strategy scheme exists is closed under conjunction.

---

[2] Alternatively, we could stipulate $\mathit{Const}(s, m) \neq \emptyset$ for all $(s, m) \in S_\Box \times \mathit{Mem}$, but this would lead to technical complications in some proofs. The presented variant seems slightly more convenient.

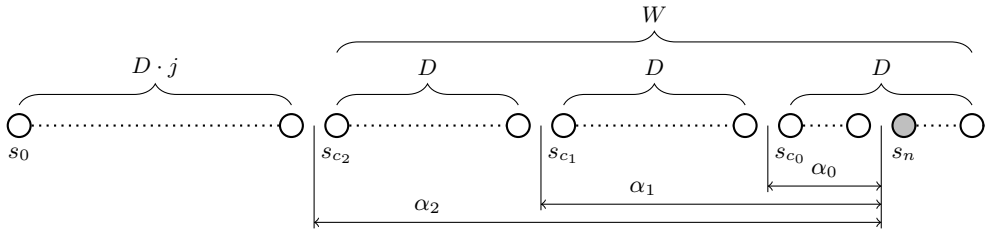

Figure 1 The information represented by the memory elements of Γ (for $\ell = 3$).

▶ **Proposition 4.** Let $\mathsf{G} = (S, (S_\Box, S_\Diamond), E)$ be a finite-state game, and $n \in \mathbb{N}$. Further, for every $1 \leq i \leq n$, let $\Gamma_i = (Mem_i, Up_i, Const_i, Init_i)$ be a strategy scheme for G which is permissive for Φ_i. Then there is a strategy scheme for G with $\prod_{i=1}^{n} |Mem_i|$ memory elements computable in $\mathcal{O}(|S|^2 \cdot |E| \cdot \prod_{i=1}^{n} |Mem_i|^2)$ time which is permissive for $\Phi_1 \wedge \cdots \wedge \Phi_n$.

As it was noted in [1], permissive strategy schemes do not exist for objectives which admit non-winning infinite runs that do not leave the winning region of player $\Box$, such as reachability, Büchi, parity, mean payoff, etc. On the other hand, permissive strategy schemes exists for "time bounded" variants of these objectives. Now we show how to compute a permissive strategy scheme for a given window-stability objective.

▶ **Theorem 5.** *Let $\mathsf{G} = (S, (S_\Box, S_\Diamond), E)$ be a finite-state game and $\Phi = (W, D, \varrho, \mu, \nu)$ a window-stability objective where $dim_\varrho = k$. Then there is a strategy scheme Γ with $W \cdot (max_\varrho \cdot W)^{k \cdot (W/D)}$ memory elements computable in $\mathcal{O}(|S|^2 \cdot |E| \cdot W^2 \cdot (max_\varrho \cdot W)^{2k \cdot (W/D)})$ time which is permissive for Φ.*

Proof. Let $\ell = W/D$ and $\mathcal{V} = \{0, \ldots, max_\varrho \cdot (W-1)\}^k$. We put

- $Mem = \{0, \ldots, D-1\} \times \{0, \ldots, \ell-1\} \times \mathcal{V}^\ell$.

Our aim is to construct Γ so that for every run $s_0, s_1, \ldots$ in G, the memory elements in the corresponding run $(s_0, m_0), (s_1, m_1), \ldots$ in G_Γ, where $(s_0, m_0) \in Init$, satisfy the following. Let $n \in \mathbb{N}_0$, and let $m_n = (i, j, \alpha_0, \ldots, \alpha_{\ell-1})$. Then

- $i = n \bmod D$ is the number of steps since the last checkpoint, and $j = \min\{\lfloor n/D \rfloor, \ell-1\}$ is a bounded counter which stores the number of checkpoint visited, up to $\ell - 1$ (this information is important for the initial W steps);
- for every $0 \leq r < \ell$, we put $c_r = n - r \cdot D - (n \bmod D)$ if $n - r \cdot D - (n \bmod D) \geq 0$, otherwise $c_r = n$. Intuitively, the state s_{c_r} is the r-th previous checkpoint visited along $s_0, s_1, \ldots$ before visiting the state s_n (see Figure 1). If the total number of checkpoints visited along the run up to s_n (including s_n) is less than r, we put $c_r = n$. The vector α_r stored in m_n is then equal to the total reward accumulated between s_{c_r} and s_n (not including s_n), i.e., $\alpha_r = \sum_{t=c_r}^{n-1} \varrho(s_t)$ where the empty sum is equal to $\vec{0}$. In particular $m_0 = (0, 0, \vec{0}, \ldots, \vec{0})$.

Note that by Definition 1, we are obliged to define $Up(s, m)$ for *all* pairs $(s, m) \in S \times Mem$, including those that will not be reachable in the end. Let '$\oplus$' be a bounded addition over $\mathbb{N}_0$ defined by $a \oplus b = \min\{a + b, max_\varrho \cdot (W-1)\}$. We extend '$\oplus$' to $\mathcal{V}$ in the natural (component-wise) way. The function Up is constructed as follows (consistently with the above intuition):

- For all $i, j \in \mathbb{N}_0$ such that $0 \leq i \leq D-2$ and $0 \leq j \leq \ell - 1$, we put $Up(s, (i, j, \alpha_0, \ldots, \alpha_{\ell-1})) = (i{+}1, j, \alpha_0 \oplus \varrho(s), \ldots, \alpha_j \oplus \varrho(s), \alpha_{j+1}, \ldots, \alpha_{\ell-1})$.

- For all $j \in \mathbb{N}_0$ such that $0 \leq j \leq \ell - 2$, we put $\mathit{Up}(s, (D{-}1, j, \alpha_0, \ldots, \alpha_{\ell-1})) = (0, j{+}1, \vec{0}, \alpha_0 \oplus \varrho(s), \ldots, \alpha_j \oplus \varrho(s), \alpha_{j+1}, \ldots, \alpha_{\ell-2})$.
- $\mathit{Up}(s, (D{-}1, \ell{-}1, \alpha_0, \ldots, \alpha_{\ell-1})) = (0, \ell{-}1, \vec{0}, \alpha_0 \oplus \varrho(s), \ldots, \alpha_{\ell-2} \oplus \varrho(s))$.

For every $(s, m) \in S \times \mathit{Mem}$, let $\mathit{succ}(s, m)$ be the set of all $(s', m') \in S \times \mathit{Mem}$ such that $(s, s') \in E$ and $\mathit{Up}(s, m) = m'$. Now we define a function $\mathcal{F} : 2^{S \times \mathit{Mem}} \to 2^{S \times \mathit{Mem}}$ such that, for a given $\Omega \subseteq S \times \mathit{Mem}$, the set $\mathcal{F}(\Omega)$ consists of all $(s, (i, j, \alpha_0, \ldots, \alpha_{\ell-1}))$ satisfying the following conditions:

- if $i = D{-}1$ and $j = \ell{-}1$, then $\mu \cdot W \leq \alpha_{\ell-1} + \varrho(s) \leq \nu \cdot W$.
- if $s \in S_\Diamond$, then $\mathit{succ}(s, (i, j, \alpha_0, \ldots, \alpha_{\ell-1})) \subseteq \Omega$.
- if $s \in S_\Box$, then $\mathit{succ}(s, (i, j, \alpha_0, \ldots, \alpha_{\ell-1})) \cap \Omega \neq \emptyset$.

Observe that $\mathcal{F}$ is monotone. Let $\mathit{gfix}(\mathcal{F})$ be the greatest fixed-point of $\mathcal{F}$. For every $(s, m) \in S_\Box \times \mathit{Mem}$, we put $\mathit{Const}(s, m) = \mathit{succ}(s, m) \cap \mathit{gfix}(\mathcal{F})$. Further, the set Init consists of all $(s, (0, 0, \vec{0}, \ldots \vec{0})) \in \mathit{gfix}(\mathcal{F})$. It follows directly from the definition of Γ that $\mathit{Const}(s, m) \neq \emptyset$ for all $(s, m) \in \mathit{Reach}(\mathit{Init})$ such that $s \in S_\Box$.

Since $\mathit{gfix}(\mathcal{F})$ can be computed in $\mathcal{O}(|S|^2 \cdot |E| \cdot W^2 \cdot (\mathit{max}_\varrho \cdot W)^{2k \cdot (W/D)})$ time by the standard iterative algorithm, the strategy scheme $\Gamma = (\mathit{Mem}, \mathit{Up}, \mathit{Const}, \mathit{Init})$ can also be computed in this time. Further, observe the following:

(A) Let $(s_0, m_0), (s_1, m_1), \ldots$ be a run in G_Γ such that $(s_0, m_0) \in \mathit{Init}$. Then $s_0, s_1, \ldots$ is a run in G that satisfies the window-stability objective Φ.

(B) Let $(s, m) \notin \mathit{gfix}(\mathcal{F})$, and let Γ^* be a strategy scheme which is the same as Γ except for its constrainer Const^* which is defined by $\mathit{Const}^*(s, m) = \mathit{succ}(s, m)$ for all $(s, m) \in S_\Box \times \mathit{Mem}$. Then there is a strategy $\pi^* \in \Pi_{\mathsf{G}_{\Gamma^*}}$ such that for every strategy $\sigma^* \in \Sigma_{\mathsf{G}_{\Gamma^*}}$ we have that $\mathit{outcome}^{\sigma^*, \pi^*}_{(s,m)}$ visits a configuration $(t, (D-1, \ell-1, \alpha_0, \ldots, \alpha_{\ell-1}))$ where $\alpha_{\ell-1} + \varrho(t) < \mu \cdot W$ or $\alpha_{\ell-1} + \varrho(t) > \nu \cdot W$.

Both (A) and (B) follow directly from the definition of $\mathcal{F}$. Now we can easily prove that Γ indeed encodes the window-stability objective Φ, i.e., $\Sigma^\Phi_\mathsf{G}(s) \approx_s \Sigma_{\mathsf{G}_\Gamma}(s)$ for all $s \in S$.

Let $\tau \in \Sigma_{\mathsf{G}_\Gamma}(s)$. We need to show that $\sigma[\tau, s]$ achieves the objective Φ in s. So, let $\pi \in \Pi_\mathsf{G}$, and let $s_0, s_1, \ldots$ be the run $\mathit{outcome}^{\sigma[\tau,s],\pi}_s$. Obviously, there is a corresponding run $(s_0, m_0), (s_1, m_1), \ldots$ in G_Γ initiated in $(s, \mathit{Init}(s))$, which means that $s_0, s_1, \ldots$ satisfies Φ by applying (A). Now let $\sigma \in \Sigma^\Phi_\mathsf{G}(s)$. We need to show that σ is admitted by Γ in s. Suppose it is not the case. If $\mathit{Init}(s) = \bot$, then $(s, (0, 0, \vec{0}, \ldots, \vec{0})) \notin \mathit{gfix}(\mathcal{F})$, and hence $\sigma \notin \Sigma^\Phi_\mathsf{G}(s)$ by applying (B). If $\mathit{Init}(s) \neq \bot$, there is a finite path $s_0, \ldots, s_n, s_{n+1}$ of *minimal length* such that $s_0 = s$, $s_n \in S_\Box$, and the corresponding finite path $(s_0, m_0), \ldots, (s_n, m_n), (s_{n+1}, m_{m+1})$ in G_{Γ^*}, where $m_0 = \mathit{Init}(s)$ and $m_{i+1} = \mathit{Up}(s_i, m_i)$ for all $0 \leq i \leq n$, satisfies that $s_{n+1} \notin \mathit{Const}(s_n, m_n)$. Note that for all $s_i \in S_\Box$ where $i < n$ we have that $s_{i+1} \in \mathit{Const}(s_i, m_i)$, because otherwise we obtain a contradiction with the minimality of $s_0, \ldots, s_n, s_{n+1}$. Since $(s_{n+1}, m_{n+1}) \notin \mathit{gfix}(\mathcal{F})$, by applying (B) we obtain a strategy $\pi^* \in \Pi_{\mathsf{G}_{\Gamma^*}}$ such that for every $\sigma^* \in \Sigma_{\mathsf{G}_{\Gamma^*}}$ we have that $\mathit{outcome}^{\sigma^*, \pi^*}_{(s_{n+1}, m_{n+1})}$ visits a configuration $(t, (D-1, \ell-1, \alpha_0, \ldots, \alpha_{\ell-1}))$ where $\alpha_{\ell-1} + \varrho(t) < \mu \cdot W$ or $\alpha_{\ell-1} + \varrho(t) > \nu \cdot W$. Let $\pi \in \Pi_\mathsf{G}$ be a strategy satisfying the following conditions:

- $\mathit{outcome}^{\sigma,\pi}_s$ starts with $s_0, \ldots, s_{n+1}$.
- For all finite paths of the form $s_0, \ldots, s_{n+1}, \ldots, s_t$ in G such that $s_t \in S_\Diamond$, let $(s_0, m_0), \ldots, (s_{n+1}, m_{n+1}), \ldots, (s_t, m_t)$ be the unique corresponding finite path in G_{Γ^*}. We put $\pi(s_0, \ldots, s_n, \ldots s_t) = s_{t+1}$, where $\pi^*((s_{n+1}, m_{n+1}), \ldots, (s_t, m_t)) = (s_{t+1}, m_{t+1})$.

Clearly, the run $\mathit{outcome}^{\sigma,\pi}_s$ does *not* satisfy the objective Φ, which contradicts the assumption $\sigma \in \Sigma^\Phi_\mathsf{G}(s)$. ◀

For every window-stability multi-objective $\Delta = \Phi_1 \wedge \cdots \wedge \Phi_n$ where we have $\Phi_i = (W_i, D_i, \varrho_i, \mu_i, \nu_i)$, we put $M_\Delta = \prod_{i=1}^{n} W_i \cdot (max_{\varrho_i} \cdot W_i)^{k_i \cdot (W_i/D_i)}$, where $k_i = dim_{\varrho_i}$. As a direct corollary to Theorem 5 and Proposition 4, we obtain the following:

▶ **Corollary 6.** *Let $\mathsf{G} = (S, (S_\Box, S_\Diamond), E)$ be a finite-state game and Δ a window-stability multi-objective. Then there is a permissive strategy scheme for Δ with M_Δ memory elements constructible in time $\mathcal{O}(|S|^2 \cdot |E| \cdot M_\Delta^2)$.*

Now we can formulate a (meta)theorem about the solvability of objectives of the form $\Delta \wedge \psi$, where Δ is a window-stability multi-objective and ψ is a reward-based objective such that the time complexity of solving Ψ for a game $\mathsf{G} = (S, (S_\Box, S_\Diamond), E)$ and a reward function ϱ can be asymptotically bounded by a function f in $|S|$, $|E|$, max_ϱ, and dim_ϱ.

▶ **Theorem 7.** *Let Ψ be a reward-based objective solvable in $\mathcal{O}(f(|S|, |E|, max_\varrho, dim_\varrho))$ time for every finite-state game $\mathsf{G} = (S, (S_\Box, S_\Diamond), E)$ and every reward function ϱ for Ψ. Further, let Δ be a window-stability multi-objective. Then the objective $\Delta \wedge \Psi$ is solvable in time*

$$\mathcal{O}(\max\{f(|S| \cdot M_\Delta, |E| \cdot M_\Delta, max_\varrho, dim_\varrho), |S|^2 \cdot |E| \cdot M_\Delta^2\})$$

for every finite-state game $\mathsf{G} = (S, (S_\Box, S_\Diamond), E)$ and every reward function ϱ for Ψ.

Note that Theorem 7 is a simple consequence of Corollary 6 and Proposition 3.

Since mean-payoff objectives are solvable in $\mathcal{O}(|S| \cdot |E| \cdot max_\varrho)$ time when $dim_\varrho = 1$ [7] and in $\mathcal{O}(|S|^2 \cdot |E| \cdot max_\varrho \cdot k \cdot (k \cdot |S| \cdot max_\varrho)^{k^2+2k+1})$ time when $dim_\varrho = k \geq 2$ [12], we finally obtain:

▶ **Theorem 8.** *Let $\mathsf{G} = (S, (S_\Box, S_\Diamond), E)$ be a finite-state game, Δ a window-stability multi-objective, and $\Psi = (\varrho, b)$ a mean-payoff objective. If $dim_\varrho = 1$, then the objective $\Delta \wedge \Psi$ is solvable in time $\mathcal{O}(|S|^2 \cdot |E| \cdot M_\Delta^2 \cdot max_\varrho)$. If $dim_\varrho = k \geq 2$, then the objective $\Delta \wedge \Psi$ is solvable in time $\mathcal{O}(|S|^2 \cdot |E| \cdot M_\Delta^3 \cdot max_\varrho \cdot k \cdot (k \cdot |S| \cdot M_\Delta \cdot max_\varrho)^{k^2+2k+1})$.*

Let us note that for a given window-stability multi-objective Δ and a given one-dimensional reward function ϱ, there exists the *maximal* bound b such that the objective $\Delta \wedge (\varrho, b)$ is achievable. Further, this bound b is rational and computable in time $\mathcal{O}(|S|^2 \cdot |E| \cdot M_\Delta^2 \cdot max_\varrho)$.

3.2 Lower Bounds for Window-Stability Objectives

We now focus on proving lower bounds for solving the window-stability objectives. More precisely, we establish lower complexity bounds for the problem whose instances are triples of the form (G, s, Φ), where G is a game (or a graph), s is a state of G, $\Phi = (W, D, \varrho, \mu, \nu)$ is a window-stability objective, and the question is whether there exists a strategy $\sigma \in \Sigma$ which achieves Φ in s. The components of Φ can be encoded in unary or binary, which is explicitly stated when presenting a given lower bound.

The main result of this section is Theorem 12 which implies that solving a window-stability objective is PSPACE-hard for games and NP-hard for graphs even if $dim_\varrho = 1$, $D = 1$, and W as well as the values $\varrho(s)$ for all $s \in S$ are encoded in *unary*. Note that an upper time complexity bound for solving these objectives is $\mathcal{O}(|S|^2 \cdot |E| \cdot W \cdot (max_\varrho \cdot W)^{W/D})$ by Corollary 6. Hence, the parameter which makes the problem hard is W/D.

As a warm-up, we first show that lower bounds for solving the window-stability objectives where the reward function is of higher dimension, or W, D, and the rewards are encoded in binary, follow rather straightforwardly from the literature. Then, we develop some new insights and use them to prove the main result.

▶ **Theorem 9.** *Solving the window-stability objectives (where $\dim_\varrho$ is not restricted) is* EXPTIME*-hard. The hardness result holds even if the problem is restricted to instances*

1. *where each component of each reward vector is in $\{-1, 0, 1\}$, or*
2. *where the reward vectors have dimension one (but the rewards are arbitrary binary-encoded numbers).*

Proof. The result can be proven by a straightforward adaptation of the proof of EXPTIME-hardness of multi-dimensional fixed-window mean-payoff problem [8, Lemma 23 and 24]. The reductions in [8] that we can mimic are from the acceptance problem for polynomial-space alternating Turing machines (item 1.) and *countdown games* [17] (item 2.). Although the fixed-window mean-payoff problem differs from ours (see Section 1), an examination of the proofs in [8] reveals that almost the same constructions work even in our setting. In particular, while the problem to which countdown games are reduced in [8] assumes two-dimensional rewards, in our setting we can restrict to single dimension due to window-stability objective imposing both a lower and an upper bound on local mean payoff. ◀

The reductions in the previous theorem require that the window size W is encoded in binary, as the windows need to be exponentially long in the size of the constructed graph. For the case when W is given in unary encoding, the following result can be adapted from [8].

▶ **Theorem 10.** *Solving the window-stability objectives (where $\dim_\varrho$ is not restricted) where the window size W is encoded in unary is* PSPACE*-hard, even if it is restricted to instances where the components of reward functions are in $\{-1, 0, 1\}$.*

A proof of Theorem 10 is obtained by adapting a proof from [8, Lemma 25], where a reduction from generalized reachability games is given.

The results of [8] do not yield lower bounds for window-stability objectives with one-dimensional reward functions in which either the windows size or the rewards are encoded in unary. In our setting, for the case of binary rewards/unary window size one can come up with NP-hardness for graphs and PSPACE-hardness for games via reductions from the Subset-Sum problem and its quantified variant [18], respectively. Similarly, for unary rewards/binary window size a PSPACE-hardness for games via reduction from emptiness of 1-letter alternating finite automata [16] seems plausible. We do not follow these directions, since we are able to prove an even stronger and somewhat surprising result: solving window-stability objectives with one-dimensional reward functions is PSPACE-hard for games and NP-hard for graphs even if *all* the numbers in the input instance are encoded in unary. The proof of this result requires a new proof technique sketched below.

We rely on reductions from special variants of the SAT and QBF problems. An instance of the Balanced-3-SAT problem is a propositional formula φ in a 3-conjunctive normal form which contains an even number of variables. Such an instance is positive if and only if φ admits a satisfying assignment which maps exactly half of φ's variables to 1 (*true*). We can also define a quantified variant, a Balanced-QBF problem: viewing a quantified Boolean formula $\psi = \exists x_1 \forall x_2 \cdots \exists x_{n-1} \forall x_n \, \varphi$ (where φ is quantifier-free), as a game between player controlling existentially quantified variables (who strives to satisfy φ) and player controlling universal variables (who aims for the opposite), we ask whether the existential player can enforce assignment mapping exactly half of the variables to 1 and satisfying φ (a formal definition of Balanced-QBF is given in [6]. The following lemma is easy.

▶ **Lemma 11.** *The* Balanced-QBF *problem is* PSPACE*-complete. The* Balanced-3-SAT *is* NP*-complete.*

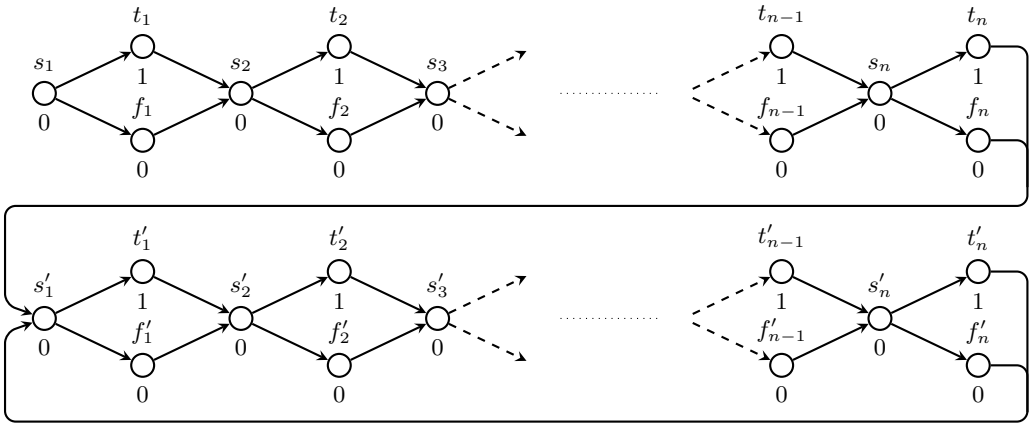

Figure 2 In the lower gadget, Player □ must mimic the assignment she chose in the upper one.

Let G be a finite-state game and $\Phi = (W, D, \varrho, \mu, \nu)$ a window-stability objective. An instance (G, s, Φ) is *small* if dim_ϱ=1, and W, D, max_ϱ, and the numerators and denominators of the fully reduced forms of μ and ν, are bounded by the number of states of G.

▶ **Theorem 12.** *Solving the window-stability objectives with one-dimensional reward functions is* PSPACE*-hard for games and* NP*-hard for graphs, even for small instances.*

Proof (sketch). We proceed by reductions from Balanced-3-SAT for graphs and from Balanced-QBF for games. As the reductions are somewhat technical, we explain just their core idea. The complete reduction can be found in [6].

Assume a formula φ in 3-CNF with variables $\{x_1, \ldots, x_n\}$, n being even. Consider the graph G in Figure 2. Both the "upper" gadget (consisting of non-primed states) and the "lower" gadget (with primed states) represent a standard "assignment choice" gadget, in which Player □ selects an assignment to variables in φ (e.g. choosing an edge going to t_1 from s_1 corresponds to setting variable x_1 to *true* etc.). With no additional constraints, □ can choose different assignments in the two gadgets, and she may change the assignment upon every new traversal of the lower gadget. Now assign reward 1 to states that correspond to setting some variable to *true* and 0 to all the other states, let window size $W = 2n$, checkpoint distance $D = 1$, $\mu = \frac{n}{2}$, and $\nu = \frac{n}{2} + \frac{1}{3n}$ (say). In order to satisfy the window-stability objective $(W, D, \varrho, \mu, \nu)$ from s_1, □ has to select a balanced assignment in the upper gadget and moreover, mimic this assignment in all future points in the lower gadgets. The necessity of the first requirement is easy. For the second, assume that there is some ℓ such that in the ℓ-th step of the run λ the player chooses to go from, say, s_i to t_i (or from s'_i to t'_i), while in the $(\ell + 2n)$-th step she goes from s'_i to f'_i. Then the rewards accumulated within windows starting in the ℓ-th and $(\ell + 1)$-th step, respectively, differ by exactly one. Thus, $|lmp_{W,D,\ell}(\lambda) - lmp_{W,D,\ell+1}(\lambda)| = 1/2n > 1/3n$, which means that the local mean payoffs at the ℓ-th and $(\ell + 1)$-th checkpoint cannot both fit into the interval $[\mu, \nu]$.

Note that we use the balanced variant of 3-SAT and QBF, as to set up μ and ν we need to know in advance the number of variables assigned to *true*.

Once we force the player to commit to some assignment using the above insight, we can add more copies of the "primed" gadget that are used to check that the assignment satisfies φ. Intuitively, we form a cycle consisting of several such gadgets, one gadget per clause of φ, the gadgets connected by paths of suitable length (not just by one edge as above). In each clause-gadget, satisfaction of the corresponding clause C by the chosen

assignment is checked by allowing the player to accrue a small additional reward whenever she visits a state representing satisfaction of some literal in C. This small amount is then subtracted and added again on a path that connects the current clause-gadget with the next one: subtracting forces the player to satisfy at least one literal in the previous clause-gadget (and thus accrue the amount needed to "survive" the subtraction) while adding ensures that this "test" does not propagate to the next clause-gadget. Rewards have to be chosen in a careful way to prevent the player from cheating. For PSPACE-hardness of the game version we simply let the adversary control states in the initial gadget (but not in clause-gadgets) corresponding to universally quantified variables. ◀

4 The variance-stability problem

In this section, we prove the results about variance-stability objectives promised in Section 1.

▶ **Theorem 13.** *The existence of a strategy achieving a given one-dimensional variance-stability objective for a given state of a given graph is in* NP. *Further, the strategy may require infinite memory.*

Let us now prove the above theorem. Consider a graph $\mathsf{G} = (S, (S, \emptyset), E)$ and an instance of the variance-stability problem determined by a reward function ϱ together with a mean-payoff bound $b \in \mathbb{Q}$ and a variance bound $c \in \mathbb{Q}$. We assume that all runs are initiated in a fixed initial state $\bar{s}$. A *frequency vector* is a tuple $(f_e)_{e \in E} \in [0,1]^{|E|}$ with $\sum_{e \in E} f_e = 1$ and

$$\sum_{s' : (s',s) \in E} f_{(s',s)} = \sum_{s' : (s,s') \in E} f_{(s,s')} \tag{1}$$

for all $s \in S$. Now consider the following constraints:

$$mp := \sum_{s \in S} f_s \cdot \varrho(s) \geq a \quad \text{and} \quad va := \sum_{s \in S} f_s \cdot (\varrho(s) - mp)^2 \leq b \tag{2}$$

Here $f_s = \sum_{(s',s) \in E} f_{(s',s)}$ for every $s \in S$. As every graph is a special case of a Markov decision process, we may invoke Proposition 5 of [5] and obtain the following proposition.

▶ **Proposition 14** ([5])**.** Assume that there is a solution to the given variance-stability problem. Then there is a frequency vector $(f_e)_{e \in E}$ satisfying the inequalities (2). All $e \in E$ satisfying $f_e > 0$ belong to the same strongly connected component of G reachable from $\bar{s}$.

The above inequalities (2) can be turned into a negative semi-definite program, using techniques of [5], and hence their satisfiability can be decided in non-deterministic polynomial time [19]. To finish our algorithm, we need to show that a solution to the above inequalities can also be turned into a strategy which visits each $e \in E$ with the frequency f_e.

Let $\lambda = s_0 s_1 \ldots$ be a run. Given $e \in E$ and $i \in \mathbb{N}$ we define a random variable $a_i^e(\lambda)$ to take value 1 if $(s_i, s_{i+1}) = e$, and 0 otherwise.

▶ **Lemma 15.** *Suppose $(f_e)_{e \in E}$ is a frequency vector such that all $e \in E$ satisfying $f_e > 0$ belong to the same strongly connected component reachable from the initial state $\bar{s}$. Then there is a strategy σ_f with $\lim_{i \to \infty} \frac{1}{i+1} \sum_{j=0}^{i} a_i^e(\lambda) = f_e$ for all $e \in E$, where λ is the run induced by σ_f (initiated in $\bar{s}$).*

Proof. Let us assume, w.l.o.g., that G itself is strongly connected. If $(f_e)_{e \in E}$ is rational and all edges e satisfying $f_e > 0$ induce a strongly connected graph, we may easily construct the strategy σ_f as follows. We multiply all numbers f_e with the least-common-multiple of

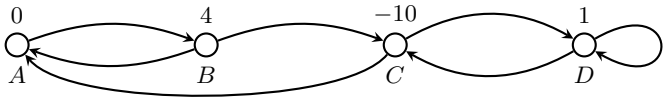

Figure 3 One player game in which there is an infinite-memory strategy σ such that $mp(outcome_s^{\sigma,\pi}) \geq 3/2$ and $va(outcome_s^{\sigma,\pi}) \leq 9/4$ (here π is the only "trivial" strategy of the environment). However, there is no finite-memory σ with this property.

their denominators and obtain a vector of natural numbers f'_e that still satisfy the above equation (1). Now we may imagine the game as a multi-digraph, where each edge e has the multiplicity f'_e. It is easy to show that the flow equations are exactly equivalent to existence of a directed Euler cycle. From this Euler cycle in the digraph we immediately get a cycle in our game which visits each edge exactly f'_e times. By repeating the cycle indefinitely we obtain a run with the desired frequencies f_e of edges.

For vectors with irrational frequencies we adapt the above approach and use a sequence of converging rational approximations. The proof is technical, and is given in [6]. ◀

This finishes the proof of Theorem 13.

We now show that variance-stability objectives may require strategies with infinite memory. Consider the graph in Figure 3, and the variance-stability objective which requires to achieve a mean payoff of at least $3/2$ and long-run variance at most $9/4$. Observe that there is an infinite-memory strategy achieving the above bounds. It works as follows: We start in the state A, the strategy proceeds in infinitely many phases. In the n-th phase it goes n times from A to B and back. Afterwards it goes to D, makes $2n$ steps on the loop on D, and then returns back to A. One can show that the mean payoff converges along this run. The limit is $4/4 + 0/4 + 1/2 = \frac{3}{2}$ since the -10 reward is obtained with zero frequency. The long-run variance is $\frac{1}{4}\left(-\frac{3}{2}\right)^2 + \frac{1}{4}\left(4-\frac{3}{2}\right)^2 + \frac{1}{2}\left(1-\frac{3}{2}\right)^2 = \frac{9}{4}$. Now we show that there is no finite-memory strategy achieving a mean payoff of $3/2$ and a long-run variance of $9/4$. Note that the maximal mean payoff achievable (without any constraints) in the graph is 2. Assume that there is a finite memory strategy σ yielding mean payoff x with $3/2 \leq x \leq 2$, and variance at most $9/4$. We first argue that σ visits C with zero frequency. Denote by f_Y the frequency of state Y. Because $x = 0 \cdot f_A + 4 \cdot f_B + (-10) \cdot f_C + 1 \cdot f_D$ by the definition of mean payoff, and also $f_A = f_B$ and $f_D = 1 - f_A - f_B - f_C$ by the definition of our graph, we have $f_A = (x + 11 \cdot f_C - 1)/2$ and $f_D = 2 - x - 12 \cdot f_C$. Thus, the variance is

$$
\begin{aligned}
&f_A \cdot (0-x)^2 + f_B \cdot (4-x)^2 + f_C \cdot (-10-x)^2 + f_D (1-x)^2 \\
&\quad = \frac{x-1}{2} \cdot \left((0-x)^2 + (4-x)^2\right) + (2-x) \cdot (1-x)^2 \\
&\qquad + f_C \cdot \left(\frac{11}{2} \cdot \left((0-x)^2 + (4-x)^2\right) + (-10-x)^2 - 12 \cdot (1-x)^2\right).
\end{aligned}
$$

Using calculus techniques one can easily show that the first term is at least $9/4$ for all $x \in [3/2, 2]$, while the parenthesized expression multiplied by f_C is positive for all such x. Hence $f_C = 0$. But any finite-memory strategy that stays in C with frequency 0 either eventually loops on D, in which case the mean payoff is only 1, or it eventually loops on A and B, in which case the variance is 4.

Even finite-memory strategies that approximate the desired variance-stability (up to some $\varepsilon > 0$) must behave in a peculiar way: Infinitely many times stay in $\{A, B\}$ for a large number of steps (depending on ε) and also stay in C for a large number of steps. Hence, in a real-life system, a user would observe two repeating phases, one with low mean payoff but high instability, and one with low stability and high mean payoff.

References

1 Julien Bernet, David Janin, and Igor Walukiewicz. Permissive strategies: from parity games to safety games. *RAIRO – Theoretical Informatics and Applications*, 36(3):261–275, 2002.

2 Aaron Bohy, Véronique Bruyère, Emmanuel Filiot, and Jean-François Raskin. Synthesis from LTL specifications with mean-payoff objectives. In *Proc. of TACAS 2013*, volume 7795 of *LNCS*, pages 169–184. Springer, 2013.

3 Patricia Bouyer, Marie Duflot, Nicolas Markey, and Gabriel Renault. Measuring permissivity in finite games. In *Proc. of CONCUR 2009*, volume 5710 of *LNCS*, pages 196–210. Springer, 2009.

4 Patricia Bouyer, Nicolas Markey, Jörg Olschewski, and Michael Ummels. Measuring permissiveness in parity games: Mean-payoff parity games revisited. In *Proc. of ATVA 2011*, volume 6996 of *LNCS*, pages 135–149. Springer, 2011.

5 Tomáš Brázdil, Krishnendu Chatterjee, Vojtěch Forejt, and Antonín Kučera. Trading performance for stability in Markov decision processes. In *Proc. of LICS 2013*, pages 331–340. IEEE, 2013.

6 Tomáš Brázdil, Vojtech Forejt, Antonín Kučera, and Petr Novotný. Stability in graphs and games. *CoRR*, abs/1604.06386, 2016.

7 L. Brim, J. Chaloupka, L. Doyen, R. Gentilini, and J.-F. Raskin. Faster algorithms for mean-payoff games. *Formal Methods in System Design*, 38(2):97–118, 2010.

8 Krishnendu Chatterjee, Laurent Doyen, Mickael Randour, and Jean-François Raskin. Looking at mean-payoff and total-payoff through windows. *Inf. Comput.*, 242:25–52, 2015.

9 Krishnendu Chatterjee, Thomas A. Henzinger, and Marcin Jurdzinski. Mean-payoff parity games. In *Proc. of LICS 2005*, pages 178–187. IEEE, 2005.

10 Krishnendu Chatterjee, Florian Horn, and Christof Löding. Obliging games. In *Proc. of CONCUR 2010*, volume 6269 of *LNCS*, pages 284–296. Springer, 2010.

11 Krishnendu Chatterjee, Mickael Randour, and Jean-Franccois Raskin. Strategy synthesis for multi-dimensional quantitative objectives. *Acta Informatica*, 51:129–163, 2014.

12 Krishnendu Chatterjee and Yaron Velner. Hyperplane separation technique for multidimensional mean-payoff games. In *Proc. of CONCUR 2013*, volume 8052 of *LNCS*, pages 500–515. Springer, 2013.

13 Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM Journal on Computing*, 31(6):1794–1813, 2002.

14 Paul Hunter, Guillermo A. Pérez, and Jean-François Raskin. Looking at mean-payoff through foggy windows. In *Proc. of ATVA 2015*, volume 9364 of *LNCS*, pages 429–445. Springer, 2015.

15 Paul Hunter and Jean-François Raskin. Quantitative games with interval objectives. In *Proc. of FST&TCS 2014*, volume 29 of *LIPIcs*, pages 365–377, 2014.

16 Petr Jančar and Zdeněk Sawa. A note on emptiness for alternating finite automata with a one-letter alphabet. *Inf. Process. Lett.*, 104(5):164 – 167, 2007.

17 Marcin Jurdzinski, Jeremy Sproston, and François Laroussinie. Model checking probabilistic timed automata with one or two clocks. *Logical Methods in Comp. Sci.*, 4(3), 2008.

18 Stephen Travers. The complexity of membership problems for circuits over sets of integers. *Theoretical Computer Science*, 369(1–3):211 – 229, 2006.

19 Stephen A. Vavasis. Quadratic programming is in NP. *Inf. Process. Lett.*, 36(2):73–77, 1990.

20 Yaron Velner. Robust multidimensional mean-payoff games are undecidable. In *Proc. of FOSSACS 2015*, volume 9034 of *LNCS*, pages 312–327. Springer, 2015.

21 Yaron Velner, Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, Alexander Rabinovich, and Jean-Francois Raskin. The complexity of multi-mean-payoff and multi-energy games. *Inf. Comput.*, 241:177 – 196, 2015.