Journal of Computer and System Sciences 88 (2017) 236–259

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Hyperplane separation technique for multidimensional mean-payoff games ☆

Krishnendu Chatterjee [a,*], Yaron Velner [b]

[a] IST Austria, Austria
[b] Tel Aviv University, Israel

ARTICLE INFO

Article history:
Received 4 May 2015
Received in revised form 11 April 2017
Accepted 19 April 2017
Available online 4 May 2017

Keywords:
Finite-state graph games
Mean-payoff objectives
Multidimensional objectives
Pushdown graphs and games
Computer-aided verification

ABSTRACT

We consider finite-state and recursive game graphs with multidimensional mean-payoff objectives. In recursive games two types of strategies are relevant: global strategies and modular strategies. Our contributions are: (1) We show that finite-state multidimensional mean-payoff games can be solved in polynomial time if the number of dimensions and the maximal absolute value of weights are fixed; whereas for arbitrary dimensions the problem is coNP-complete. (2) We show that one-player recursive games with multidimensional mean-payoff objectives can be solved in polynomial time. Both above algorithms are based on hyperplane separation technique. (3) For recursive games we show that under modular strategies the multidimensional problem is undecidable. We show that if the number of modules, exits, and the maximal absolute value of the weights are fixed, then one-dimensional recursive mean-payoff games under modular strategies can be solved in polynomial time, whereas for unbounded number of exits or modules the problem is NP-hard.

1. Introduction

In this work we present a hyperplane based technique that solves several fundamental algorithmic open questions for multidimensional mean-payoff objectives. We first present an overview of mean-payoff games, then the important extensions, followed by the open problems, and finally our contributions.

Mean-payoff games on graphs. Two-player games played on finite-state graphs provide the mathematical framework to analyze several important problems in computer science as well as in mathematics, such as formal analysis of reactive systems [11,37,36]. Games played on graphs are dynamic games that proceed for an infinite number of rounds. The vertex set of the graph is partitioned into player-1 vertices and player-2 vertices. The game starts at an initial vertex, and if the current vertex is a player-1 vertex, then player 1 chooses an outgoing edge, and if the current vertex is a player-2 vertex, then player 2 does likewise. This process is repeated forever, and gives rise to an outcome of the game, called a *play*, that consists of the infinite sequence of vertices that are visited. The most well-studied payoff criteria in such games is the *mean-payoff* objective, where a weight (representing a reward) is associated with every transition and the goal of one of the players is to maximize the long-run average of the weights; and the goal of the opponent is to minimize it. Mean-payoff games and the special case of graphs (with only one player) with mean-payoff objectives have been extensively studied

☆ A preliminary version of the paper appeared in CONCUR 2013 [19].

* Corresponding author.
E-mail address: krish.chat@gmail.com (K. Chatterjee).

http://dx.doi.org/10.1016/j.jcss.2017.04.005
0022-0000/

over the last three decades; e.g. [31,23,28,44]. Graphs with mean-payoff objectives can be solved in polynomial time [31], whereas mean-payoff games can be decided in NP ∩ coNP [23,44]. The mean-payoff games problem is an intriguing problem and one of the rare combinatorial problems that is known to be in NP ∩ coNP, but for which no polynomial time algorithm is known. However, pseudo-polynomial time algorithms exist for mean-payoff games [44,10], and if the weights are bounded by a constant, then the algorithm is polynomial.

The extensions. Motivated by applications in formal analysis of reactive systems, the study of mean-payoff games has been extended in two directions: (1) pushdown mean-payoff games; and (2) multidimensional mean-payoff games on finite game graphs. Pushdown games, aka games on recursive state machines, can model reactive systems with recursion (i.e., model the control flow of sequential programs with recursion). Pushdown games have been studied widely with applications in verification, synthesis, and program analysis in [42,43,2] (also see [24,7,25,8] for sample research in stochastic pushdown games). In applications of verification and synthesis, the quantitative objectives that typically arise are multidimensional quantitative objectives (i.e., conjunction of several objectives), e.g., to express properties like the average response time between a grant and a request is below a given threshold ν_1, and the average number of unnecessary grants is below a threshold ν_2. Thus mean-payoff objectives can express properties related to resource requirements, performance, and robustness; multiple objectives can express the different, potentially dependent or conflicting objectives. Moreover, recently many quantitative logics and automata theoretic formalisms have been proposed with mean-payoff objectives in their heart to express properties such as reliability requirements, and resource bounds of reactive systems [14,22,6,5], and several quantitative synthesis questions (such as synthesis from incompatible specifications [38]) translate directly to multidimensional mean-payoff games. Thus pushdown games and graphs with mean-payoff objectives, and finite-state game graphs with multidimensional mean-payoff objectives are fundamental theoretical questions in model checking of quantitative logics and quantitative analysis of reactive systems (along with recursion features). Pushdown games with multidimensional objectives are also a natural generalization to study. Furthermore, in applications related to reactive system analysis, the number of dimensions of mean-payoff objectives is typically small, say 2 or 3, as they denote the different types of resources; and the weights denoting the resource consumption amount are also bounded by constants; whereas the state space of the reactive system is huge; see [4,9] for examples.

Relevant aspects of pushdown games. In pushdown games two types of strategies are relevant and studied in the literature. The first one are the *global* strategies, where a global strategy can choose the successor vertex depending on the entire global history of the play; where history is the finite sequence of configurations of the current prefix of a play (note that in finite-state games only global strategies are relevant). The second are *modular* strategies, which are understood more intuitively in the model of games on recursive state machines. A *recursive state machine* (RSM) consists of a set of component machines (or modules). Each module has a set of *nodes* (atomic states) and *boxes* (each of which is mapped to a module), a well-defined interface consisting of *entry* and *exit* nodes, and edges connecting nodes/boxes. An edge entering a box models the invocation of the module associated with the box and an edge leaving the box represents return from the module. In the game version the nodes are partitioned into player-1 nodes and player-2 nodes. Due to recursion the underlying global state-space is infinite and isomorphic to pushdown games. The equivalence of pushdown games and recursive games has been established in [2]. A modular strategy is a strategy that has only local memory, and thus, the strategy does not depend on the context of invocation of the module, but only on the history within the current invocation of the module. Informally, modular strategies are appealing because they are stackless strategies, decomposable into one for each module.

Previous results and open questions. We now summarize the main previous results and open questions and then present our contributions.

1. *(Finite-state graphs).* Finite-state graphs (or one-player games) with mean-payoff objectives can be solved in polynomial time [31], and finite-state graphs with multidimensional mean-payoff objectives can also be solved in polynomial time [41,40] using the techniques to detect zero-circuits in graphs of [32].
2. *(Finite-state games).* Finite-state games with a one-dimensional mean-payoff objective can be decided in NP ∩ coNP [23,44], and pseudo-polynomial time algorithms exist for mean-payoff games [44,10]: the current fastest known algorithm works in time $O(n \cdot m \cdot W)$, where n is the number of vertices, m is the number of edges, and W is the maximal absolute value of the weights [10]. Finite-state games with multidimensional mean-payoff objectives are coNP-complete with weights in $\{-1, 0, 1\}$ (i.e., the weights are bounded by a constant) but with arbitrary dimensions [15,40], and the current best known algorithm works in time $O(2^n \cdot \text{poly}(n, m, \log W))$.
3. *(Pushdown graphs and games).* Pushdown graphs and games have been studied only for one-dimensional mean-payoff objectives [17], and also applied for quantitative interprocedural analysis for problems in programming languages [16]. Under global strategies, pushdown graphs with a one-dimensional mean-payoff objective can be solved in polynomial time, whereas pushdown games are undecidable. Under modular strategies, pushdown graphs with every module restricted to have single exit and weights restricted to $\{-1, 0, 1\}$ are NP-hard, and pushdown games with any number of exits and general weight function are in NP (i.e., the problems are NP-complete) [17].

Many fundamental algorithmic questions have remained open for analysis of finite-state and pushdown graphs and games with multidimensional mean-payoff objectives where the goal of player 1 is to ensure that the mean-payoff is at least zero

in all dimensions. The most prominent ones are: (A) Can finite-state game graphs with multidimensional mean-payoff objectives with 2 or 3 dimensions and constant weights be solved in polynomial time? (note that with arbitrary dimensions the problem is coNP-complete, and for arbitrary weights no polynomial time algorithm is known even for the one-dimensional case. Also note that when the limit supremum of the long-run average is considered, then pseudo-polynomial algorithm exists even for unbounded number of dimensions [41]); (B) Can pushdown graphs under global strategies with multidimensional mean-payoff objectives be solved in polynomial time?; (C) Can a polynomial time algorithm be obtained for pushdown games under modular strategies with a one-dimensional mean-payoff objective when relevant parameters (such as the number of modules) are bounded?; and (D) In what complexity class does pushdown games under modular strategies with multidimensional mean-payoff objectives lie? The above questions are not only of theoretical interest, but stem from practically motivated problems of formal analysis of reactive systems.

Our contributions. In this work we present a hyperplane separation technique to provide answers to many of the open fundamental questions. Our contributions are summarized as follows:

1. *(Hyperplane technique).* We use the separating hyperplane technique from computational geometry to answer the open questions (A) and (B) above. First, we present an algorithm for finite-state games with multidimensional mean-payoff objectives of k-dimensions that works in time $O(n^2 \cdot m \cdot k \cdot W \cdot (k \cdot n \cdot W)^{k^2+2\cdot k+1})$ (Section 2: Theorem 1), and thus for constant weights and any constant k (not only $k = 2$ or $k = 3$) our algorithm is polynomial. Second, we present a polynomial-time algorithm for pushdown graphs under global strategies with multidimensional mean-payoff objectives (Section 3: Theorem 2); the algorithm is polynomial for general weight function and any number of dimensions. Our key intuition is to reduce the multidimensional problem to searching for a separating hyperplane such that all realizable mean-payoff vectors lie on one side of the hyperplane. This intuition allows us to search for a vector, which is normal to the separating hyperplane, and reduce the multidimensional problem to a one-dimensional problem by multiplying the multidimensional weight function by the vector.
2. *(Modular pushdown games).* We first show that the hyperplane techniques do not extend for modular strategies in pushdown games: we show that pushdown games under modular strategies with multidimensional mean-payoff objectives with fixed number of dimensions are undecidable (Section 4: Theorem 3). Thus the only relevant algorithmic problem for pushdown games is the modular strategies problem for a one-dimensional mean-payoff objective; under global strategies even a one-dimensional mean-payoff objective problem is undecidable [17]. It was already shown in [17] that if the number of modules is unbounded, then even with single exits for every module the problem is NP-hard. We show that pushdown games under modular strategies with one-dimensional mean-payoff objectives are NP-hard with two modules and with weights $\{-1, 0, 1\}$ if the number of exits is unbounded (Section 4: Theorem 4). Thus to obtain a polynomial time algorithm we need to bound both the number of modules as well as the number of exits. We show that pushdown games under modular strategies with one-dimensional mean-payoff objectives can be solved in time $(n \cdot \mathsf{M})^{O(\mathsf{M}^5+\mathsf{M}\cdot\mathsf{E}^2)} \cdot W^{O(\mathsf{M}^2+\mathsf{E})}$, where n is the number of vertices, W is the maximal absolute weight, M is the number of modules, and E is the number of exits (Section 4: Theorem 6). Thus if M, E, and W are constants, our algorithm is polynomial. Hence we answer the open questions (C) and (D).

Given our polynomial-time algorithms when the parameters are fixed for finite-state multidimensional mean-payoff games and pushdown games with a one-dimensional mean-payoff objective under modular strategies, a natural question is whether they are fixed parameter tractable, e.g., could we obtain an algorithm that runs in time $f(k) \cdot O(\text{poly}(n, m, W))$ (resp. $f(\mathsf{M}, \mathsf{E}) \cdot O(\text{poly}(n, W))$) for finite-state multidimensional mean-payoff games (resp. for pushdown modular games with one-dimensional objective), for some computable function f (e.g., exponential or double exponential). In [19] we showed hardness of fixed parameter tractability (for both problems) by a reduction to the once long-standing open problem of fixed parameter tractability of parity games to both problems. However, it was recently proved that parity games are fixed parameter tractable [12].

Recent related work. A recent related work [30] considers finite-state games with multi-dimensional energy objectives (or equivalently, multi-dimensional mean-payoff objectives with the restriction of finite-memory strategies), and presents a pseudo-polynomial time algorithm for constant number of dimensions. The results of [30] build on the techniques that we introduce in this work, for example, the hyperplane separation technique, and strategy construction methods of our paper are used in [30, Section 4.3.1 and 4.3.2]. Thus the techniques introduced in the paper not only present interesting solutions for the problem we consider, but are also applicable in other scenarios.

2. Finite-state games with multidimensional mean-payoff objectives

In this section we will present an algorithm for finite-state multidimensional mean-payoff games for which the running time is polynomial when the number of dimensions and weights are fixed. We start with the basic definitions of finite-state games, strategies, and mean-payoff objectives.

Game graphs. A *game graph* $G = ((V, E), (V_1, V_2))$ consists of a *finite* directed graph (V, E) with a finite set V of n vertices and a set $E \subseteq V \times V$ of m edges, and a partition (V_1, V_2) of V into two sets. The vertices in V_1 are *player-1 vertices*,

where player 1 chooses the outgoing edges, and the vertices in V_2 are *player-2 vertices*, where player 2 (the adversary to player 1) chooses the outgoing edges. Intuitively game graphs are the same as AND-OR graphs. For a vertex $u \in V$, we write $\mathsf{Out}(u) = \{v \in V \mid (u, v) \in E\}$ for the set of successor vertices of u. We assume that every vertex has at least one outgoing edge, i.e., $\mathsf{Out}(u)$ is non-empty for all vertices $u \in V$.

Plays. A game is played by two players: player 1 and player 2, who form an infinite path in the game graph by moving a token along edges. They start by placing the token on an initial vertex, and then they perform moves indefinitely in the following way. If the token is on a vertex in V_1, then player 1 moves the token along one of the edges going out of the vertex. If the token is on a vertex in V_2, then player 2 does likewise. The result is an infinite path in the game graph, called *plays*. Formally, a *play* is an infinite sequence $\pi = \langle v_0, v_1, v_2, \ldots \rangle$ of vertices such that $(v_j, v_{j+1}) \in E$ for all $j \geq 0$.

Strategies. A strategy for a player is a rule that specifies how to extend plays. Formally, a *strategy* τ for player 1 is a function $\tau\colon V^* \cdot V_1 \to V$ that, given a finite sequence of vertices (representing the history of the play so far) which ends in a player 1 vertex, chooses the next vertex. The strategy must choose only available successors, i.e., for all $w \in V^*$ and $v \in V_1$ we have $\tau(w \cdot v) \in \mathsf{Out}(v)$. The strategies for player 2 are defined analogously. A strategy is *memoryless* if it is independent of the history and only depends on the current vertex. Formally, a memoryless strategy for player 1 is a function $\tau\colon V_1 \to V$ such that $\tau(v) \in \mathsf{Out}(v)$ for all $v \in V_1$, and analogously for player 2 strategies. Given a starting vertex $v \in V$, a strategy τ for player 1, and a strategy σ for player 2, there is a unique play, denoted $\pi(v, \tau, \sigma) = \langle v_0, v_1, v_2, \ldots \rangle$, which is defined as follows: $v_0 = v$ and for all $j \geq 0$, if $v_j \in V_1$, then $\tau((v_0, v_1, \ldots v_j)) = v_{j+1}$, and if $v_j \in V_2$, then $\sigma((v_0, v_1, \ldots, v_j)) = v_{j+1}$.

Graphs obtained under memoryless strategies. A player-1 graph is a special case of a game graph where all vertices in V_2 have a unique successor (and player-2 graphs are defined analogously). Given a memoryless strategy σ for player 2, we denote by G^σ the player-1 graph obtained by removing from all player-2 vertices the edges that are not chosen by σ.

Multidimensional mean-payoff objectives. For multidimensional mean-payoff objectives we will consider game graphs along with a weight function $w : E \to \mathbb{Z}^k$ that maps each edge to a vector of integer weights. We denote by W the maximal absolute value of the weights. For a finite path π, we denote by $w(\pi)$ the sum of the weight vectors of the edges in π and $\mathsf{Avg}(\pi) = \frac{w(\pi)}{|\pi|}$, where $|\pi|$ is the length of π, denotes the average vector of the weights. We denote by $\mathsf{Avg}_i(\pi)$ the projection of $\mathsf{Avg}(\pi)$ to the i-th dimension. For an infinite path π, let ρ_t denote the finite prefix of length t of π; and we define $\mathsf{LimInfAvg}_i(\pi) = \liminf_{t\to\infty} \mathsf{Avg}_i(\rho_t)$ and analogously $\mathsf{LimSupAvg}_i(\pi)$ with $\liminf$ replaced by $\limsup$. For an infinite path π, we denote by $\mathsf{LimInfAvg}(\pi) = (\mathsf{LimInfAvg}_1(\pi), \ldots, \mathsf{LimInfAvg}_k(\pi))$ (resp. $\mathsf{LimSupAvg}(\pi) = (\mathsf{LimSupAvg}_1(\pi), \ldots, \mathsf{LimSupAvg}_k(\pi))$) the limit-inf (resp. limit-sup) vector of the averages (long-run average or mean-payoff objectives). The objective of player 1 we consider is to ensure that the mean-payoff is non-negative in every dimension, i.e., to ensure $\mathsf{LimInfAvg}(\pi) \geq \vec{0}$, where $\vec{0}$ denotes the vector of all zeros.

Remark 1. A mean-payoff objective is invariant to the shift operation, i.e., if in a dimension i, we require that the mean-payoff is at least ν_i, then we simply subtract ν_i in the weight vector from every edge in the i-th dimension and require the mean-payoff is at least 0 in dimension i. Hence the comparison with $\vec{0}$ is without loss of generality. We will present all the results for LimInfAvg objectives because the results for LimSupAvg objectives are simpler. A pseudo polynomial algorithm for arbitrary multidimensional LimSupAvg was given in [41]. [41] also show a reduction from multidimensional mean-payoff games where some of the coordinates are computed according to LimSupAvg and some are computed according to LimInfAvg, into a game where all coordinates are computed according to LimInfAvg. Hence, in the sequel we will write LimAvg for LimInfAvg. Moreover, all the results we will present would also hold if we replace the non-strict inequality comparison ($\geq \vec{0}$) with a strict inequality ($> \vec{0}$).

Winning strategies. A player-1 strategy τ is a winning strategy from a set U of vertices, if for all player-2 strategies σ and all $v \in U$ we have $\mathsf{LimAvg}(\pi(v, \tau, \sigma)) \geq \vec{0}$. A player-2 strategy is a winning strategy from a set U of vertices if for all player-1 strategies τ and for all $v \in U$ we have that the path $\pi(v, \tau, \sigma)$ does not satisfy $\mathsf{LimAvg}(\pi(v, \tau, \sigma)) \geq \vec{0}$. The winning region for a player is the largest set U such that the player has a winning strategy from U.

We now give an intuitive overview on our algorithm to decide the existence of a winning strategy for player 1 in finite-state multidimensional mean-payoff games.

Hyperplane separation technique. Our key insight is to search for a hyperplane $\mathcal{H}$, which contains the origin, such that player 2 can ensure a mean-payoff vector that lies below $\mathcal{H}$ (i.e., the product of the vector and the vector that is normal [1] to the hyperplane is negative). Intuitively, we show that if such a hyperplane exists, then any point in space that is below $\mathcal{H}$ is negative in at least one dimension, and thus the multidimensional mean-payoff objective for player 1 is violated. Conversely, we show that if for all hyperplanes $\mathcal{H}$ player 1 can achieve a mean-payoff vector that lies above $\mathcal{H}$, then player 1 can ensure the multidimensional mean-payoff objective. The technical argument relies on the fact that if we have an infinite sequence of unit vectors $\vec{b}_1, \vec{b}_2, \ldots$ and $\vec{b}_\ell$ lies above the hyperplane that is normal to $\sum_{j=1}^{\ell-1} \vec{b}_j$, then $\liminf_{\ell\to\infty} \frac{1}{\ell} \cdot \sum_{j=1}^{\ell} \vec{b}_j = \vec{0}$.

[1] The orientation of the normal is according to the right-hand rule.

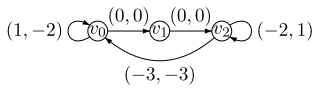

Fig. 1. Game graph G_1.

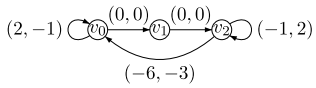

Fig. 2. Game graph G_2.

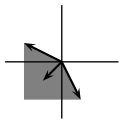

Fig. 3. Feasible vectors for G_1.

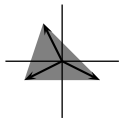

Fig. 4. Feasible vectors for G_2.

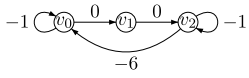

Fig. 5. Game graph G_1 with weight function $\vec{\lambda} \cdot w_1$ for $\vec{\lambda} = (1, 1)$.

Multiple dimensions to one dimension. Given a multidimensional weight function w and a vector $\vec{\lambda}$, we denote by $w \cdot \vec{\lambda}$ the one-dimensional weight function that assigns every edge e the weight value $w(e)^T \cdot \vec{\lambda}$, where $w(e)^T$ is the transpose of the weight vector $w(e)$. We show that with the hyperplane technique we can reduce a game with multidimensional mean-payoff objective to the same game with a one-dimensional mean-payoff objective. A vector $\vec{b}$ lies above a hyperplane $\mathcal{H}$ if $\vec{\lambda}$ is the normal vector of $\mathcal{H}$ and $\vec{b}^T \cdot \vec{\lambda} \geq 0$. Hence, player 1 can achieve a mean-payoff vector that lies above $\mathcal{H}$ if and only if player 1 can ensure the one-dimensional mean-payoff objective with weight function $w(e) \cdot \vec{\lambda}$.

Examples. Consider the game graph G_1 (Fig. 1) where all vertices belong to player 1. The weight function w_1 labels each edge with a two-dimensional weight vector. In G_1, player 1 can ensure all mean-payoff vectors that are convex combination of $(1, -2), (-2, 1)$ and $(-1, -1)$ (see Fig. 3). By Fig. 3, all the vectors reside below the hyperplane $y = -x$, and consider the normal vector $\vec{\lambda} = (1, 1)$ to the hyperplane $y = -x$. All the cycles in G_1 with weight function $w_1 \cdot \vec{\lambda}$ (shown in Fig. 5) have negative weights. Therefore player 1 loses in the one-dimensional mean-payoff objective. Consider the game graph G_2 (Fig. 2) with all player-1 vertices; where player 1 can achieve any mean-payoff vector that is a convex combination of $(2, -1), (-1, 2)$ and $(-2, -1)$ (see Fig. 4). By Fig. 4, every hyperplane that passes through the origin intersects with the feasible region. Thus, no separating hyperplane exists.

Basic lemmas and assumptions. We now prove two lemmas to formalize the intuition related to reduction to one-dimensional mean-payoff games. Lemma 1 requires two assumptions, which we later show (in Lemma 4) how to deal with. The assumptions are as follows: (1) The first assumption (to which we refer as Assumption 1) is that every outgoing edge of player-2 vertices is to a player-1 vertex; formally, $E \cap (V_2 \times V) \subseteq E \cap (V_2 \times V_1)$. (2) The second assumption (to which we refer as Assumption 2) is that every player-1 vertex has k (recall that k is the dimension of the weight function) self-loop edges $e_1, \ldots, e_k$ such that $w_i(e_j) = 0$ if $i \neq j$ and $w_i(e_i) = -1$ (technically, adding several self-loop edges creates a multi-graph, but a dummy player-2 vertex can be put for every such edge to ensure that we do not have a multi-graph).

Intuitively, we can make such assumption w.l.o.g. since we can add to the graph dummy player-1 vertices, where he can make no choices (i.e., with out-degree 1) without changing the winning regions for the mean-payoff objective (and thus Assumption 1 is satisfied). In addition, adding the self-loop as described in Assumption 2 would also not change the winning region. Hence, the assumptions are made w.l.o.g. (we formally prove this intuition in Lemma 4). On the other hand, with the two assumptions we get that for every i player 1 can enforce a mean-payoff vector with -1 in its i-th coordinate. Thus, there exists a separating hyperplane if and only if in addition player 1 could not also enforce a mean-payoff vector that resides in $(0, \infty)^k$ (i.e., satisfy the multidimensional winning condition).

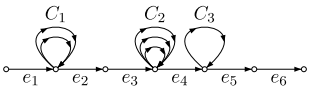

Fig. 6. The path segment ρ_i is decomposed into the cycles (possibly repeated) as $\rho_i^2 = C_1 \cdot C_1 \cdot C_2 \cdot C_2 \cdot C_2 \cdot C_3$; and the acyclic part $\rho_i^1 = e_1 \cdot e_2 \cdot e_3 \cdot e_4 \cdot e_5 \cdot e_6$.

Let us denote by Win^2 the player-2 winning region in the multidimensional mean-payoff game with weight function w, and by $\mathsf{Win}^2_{\vec{\lambda}}$ the player-2 winning region in the one-dimensional mean-payoff game with the weight function $w \cdot \vec{\lambda}$. The next lemma shows that if $\mathsf{Win}^2_{\vec{\lambda}} \neq \emptyset$, then $\mathsf{Win}^2 \neq \emptyset$; i.e., presents a sufficient condition for the non-emptiness of Win^2.

Lemma 1. *Given a game graph G that satisfies Assumption 1 and Assumption 2, and a multidimensional mean-payoff objective with weight function w, for every $\vec{\lambda} \in \mathbb{R}^k$ we have $\mathsf{Win}^2_{\vec{\lambda}} \subseteq \mathsf{Win}^2$; (hence, if $\mathsf{Win}^2_{\vec{\lambda}} \neq \emptyset$, then $\mathsf{Win}^2 \neq \emptyset$).*

Proof. Intuitively, due to Assumptions 1 and 2 we may assume that all the coefficients of $\vec{\lambda}$ are positive. Hence, if player 2 wins in the single-dimensional game, i.e., if the long-run average weight according to $w \cdot \vec{\lambda}$ is negative, then the long-run average weight (according to w) of one of the dimensions must be negative.

Formally, let σ be a player-2 winning strategy in G from an initial vertex v_0 (i.e., winning strategy from the set $\{v_0\}$) for the one-dimensional mean-payoff objective with weight function $w \cdot \vec{\lambda}$. We first observe that we must have $\vec{\lambda} \in (0, \infty)^k$; otherwise if $\lambda_i \in (-\infty, 0]$ then by Assumption 2 the weight of the i-th self-loop of a player-1 vertex would be non-negative, and player 1 can ensure the mean-payoff objective from all vertices (by Assumption 1 all plays arrive to a player-1 vertex within one step), contradicting v_0 is winning for player 2. We claim that σ is also a player-2 winning strategy with respect to the multidimensional mean-payoff objective. Indeed, let ρ be a play that is consistent with σ. Since σ is a player-2 winning strategy for the mean-payoff objective with weight function $w \cdot \vec{\lambda}$, it follows that there exists a constant $c > 0$ such that there are infinitely many prefixes of ρ with average weight (according to $w \cdot \vec{\lambda}$) at most $-c$. Let $\lambda_{\min} = \min\{\vec{\lambda}_i \mid 1 \leq i \leq k\}$ be the minimum value of $\vec{\lambda}$ among its dimensions. Since $\vec{\lambda} \in (0, \infty)^k$, it follows that $\lambda_{\min} > 0$. Since there are finitely many dimensions there must be a dimension i for which there are infinitely many prefixes of ρ with average weight at most $-\frac{c \cdot \lambda_{\min}}{k} < 0$ in dimension i (according to the weight function w). Hence, the mean-payoff value of dimension i is negative, and thus the multidimensional mean-payoff objective is violated. Hence σ is a player-2 winning strategy from v_0 against the multidimensional mean-payoff objective. □

We now present a lemma that will complement Lemma 1, and the following lemma does not require Assumption 1 nor Assumption 2.

Lemma 2. *Given a game graph G and a multidimensional mean-payoff objective with weight function w, if for all $\vec{\lambda} \in \mathbb{R}^k$ we have $\mathsf{Win}^2_{\vec{\lambda}} = \emptyset$, then we have $\mathsf{Win}^2 = \emptyset$.*

Proof. Intuitively, if player 1 wins for the single dimensional weight function $w \cdot \vec{\lambda}$, then in the multidimensional game he can always decrease the size (Euclidean norm) of the average weight vector $\vec{\lambda}'$ by playing to achieve a vector that resides in the same hyperplane as $-\vec{\lambda}'$. The latter is always possible as player 1 can obtain non-negative mean-payoff according to $w \cdot (-\vec{\lambda}')$ for every $\vec{\lambda}'$. We note that such strategy requires infinite-memory. However, it is known that in multidimensional mean-payoff games infinite-memory strategies are necessary in general [15].

Formally, since $\mathsf{Win}^2_{\vec{\lambda}} = \emptyset$ for every $\vec{\lambda} \in \mathbb{R}^k$, it follows by the determinacy of one-dimensional mean-payoff games [23] that for all $\vec{\lambda} \in \mathbb{R}^k$, player 1 can ensure the one-dimensional mean-payoff objective with weight function $w \cdot \vec{\lambda}$ in G (from all initial vertices). We now present an explicit construction of a player-1 winning strategy for the multidimensional mean-payoff objective in G from all vertices. For a vector $\vec{\lambda} \in \mathbb{R}^k$, let $\tau_{\vec{\lambda}}$ be a memoryless player-1 winning strategy in G from all vertices for the one-dimensional mean-payoff game with weight function $w \cdot \vec{\lambda}$ (note that uniform memoryless winning strategies that ensure winning from all vertices in the winning region exist in one-dimensional mean-payoff games by the results of [23]). We construct a player-1 winning strategy τ for the multidimensional objective in the following way:

- Initially, set $\vec{b}_0 := (1, 1, \ldots, 1)$.
- For $i = 1, 2, \ldots, \infty$, in iteration i play as follows:
  - Set $\vec{\lambda}_{b_i} := -\vec{b}_{i-1}$. In τ, player 1 plays according to $\tau_{\vec{\lambda}_{b_i}}$ for i rounds.
  - Let ρ_i be the play suffix that was formed in the last i rounds (or steps) of the play (i.e., the play suffix that was formed during the i-th iteration). From ρ_i we obtain the part of ρ_i that consists of cycles (that are possibly repeated) and denote the part as ρ_i^2; and an acyclic part ρ_i^1 of length at most n (where n is the number of states in the game). Informally, ρ_i^2 consists of cycles C that appear in ρ_i, and if cycle C is repeated j times in ρ_i then it is included j times in ρ_i^2; see Fig. 6 for an illustration.
  - Set $\vec{b}_i := \vec{b}_{i-1} + w(\rho_i^2)$; and proceed to the next iteration.

In order to prove that τ is a winning strategy, it is enough to prove that for every play ρ that is consistent with τ, the Euclidean norm of the average weight vector tends to zero as the length of the play tends to infinity.

We first compute the Euclidean norm of $\vec{b}_i$. For this purpose we observe that $\tau_{\vec{\lambda}_{b_i}}$ is a memoryless winning strategy for the one-dimensional mean-payoff game with weight function $w \cdot \vec{\lambda}_{b_i}$; and hence it follows that for every cycle C in the graph $G^{\tau_{\vec{\lambda}_{b_i}}}$ the sum of the weights of C according to $w \cdot \vec{\lambda}_{b_i}$ is non-negative. Since ρ_i^2 is composed of cyclic paths (from $G^{\tau_{\vec{\lambda}_{b_i}}}$), we must have $w(\rho_i^2)^T \cdot \vec{\lambda}_{b_i} \geq 0$; and hence, we have $w(\rho_i^2)^T \cdot \vec{b}_{i-1} \leq 0$. Thus we get that

$$|\vec{b}_i| = |\vec{b}_{i-1} + w(\rho_i^2)| = \sqrt{|\vec{b}_{i-1}|^2 + 2 \cdot w(\rho_i^2)^T \cdot \vec{b}_{i-1} + |w(\rho_i^2)|^2} \leq \sqrt{|\vec{b}_{i-1}|^2 + |w(\rho_i^2)|^2}$$

Since W is the maximal absolute value of the weights, it follows that $W \cdot \sqrt{k}$ is a bound on the Euclidean norm of any average weight vector. Since the length of ρ_i^2 is at most i (it was a part of the suffix of last i rounds) we get that

$$|\vec{b}_i| \leq \sqrt{|\vec{b}_{i-1}|^2 + k \cdot W^2 \cdot i^2}.$$

By a simple induction we obtain that $|\vec{b}_i| \leq \sqrt{k \cdot W^2 \cdot \sum_{j=1}^{i} j^2}$. Thus we have

$$|\vec{b}_i| \leq \sqrt{k \cdot W^2 \cdot \sum_{j=1}^{i} j^2} \leq \sqrt{k \cdot W^2 \cdot i^3} \ .$$

We are now ready to compute the Euclidean norm of the play after the i-th iteration. We denote the weight vector after the i-th iteration by $\vec{x}_i$ and observe that

$$\vec{x}_i = \vec{b}_i + \sum_{j=1}^{i} w(\rho_j^1) \ .$$

By the Triangle inequality we get that

$$|\vec{x}_i| \leq |\vec{b}_i| + \sum_{j=1}^{i} |w(\rho_j^1)| \ .$$

Since the length of every ρ_j^1 is at most n and by the bound we obtained over $\vec{b}_i$ we get that

$$|\vec{x}_i| \leq \sqrt{k \cdot W^2 \cdot i^3} + i \cdot n \cdot W \cdot \sqrt{k}$$

For a position j of the play between iteration i and iteration $i+1$, let us denote by $\vec{y}_j$ the weight vector after the play prefix at position j. Since there are i steps played in iteration i we have $|\vec{y}_j| \leq |\vec{x}_i| + i \cdot W \cdot \sqrt{k}$. Finally, since after the $(i-1)$-th iteration $\sum_{t=1}^{i-1} t = i \cdot (i-1)/2$ rounds were played, we get that the Euclidean norm of the average weight vector, namely, $|\frac{\vec{y}_j}{j}| \leq \frac{|\vec{y}_j|}{i \cdot (i-1)/2}$, tends to zero as i tends to infinity. Formally we have

$$\lim_{j \to \infty} \frac{|\vec{y}_j|}{j} \leq \lim_{i \to \infty} \frac{\sqrt{k \cdot W^2 \cdot i^3} + i \cdot n \cdot W \cdot \sqrt{k} + i \cdot W \cdot \sqrt{k}}{i \cdot (i-1)/2} = 0$$

It follows that the limit average of the weight vectors is zero and hence the desired result follows. □

Lemma 1 and Lemma 2 suggest that in order to check if player-2 winning region is non-empty in a multidimensional mean-payoff game it suffices to go over all (uncountably many) $\vec{\lambda} \in \mathbb{R}^k$ and check whether player-2 winning region is non-empty in the one-dimensional mean-payoff game with weight function $w \cdot \vec{\lambda}$. The next lemma shows that we need to consider only finitely many vectors; and we first introduce some notations that we will use.

Notations. For the rest of this section, we let $M = (k \cdot n \cdot W)^{k+1}$, where W is the maximal absolute value of the weight function, n is the number of states in the graph and k is the number of dimensions. For a positive integer ℓ, we will denote by $\mathbb{Z}_\ell^{\pm} = \{i \mid -\ell \leq i \leq \ell\}$ (resp. $\mathbb{Z}_\ell^{+} = \{i \mid 1 \leq i \leq \ell\}$) the set of integers (resp. positive integers) from $-\ell$ to ℓ.

Lemma 3. *Let G be a game graph with a multidimensional mean-payoff objective with a weight function w. There exists $\vec{\lambda}_0 \in \mathbb{R}^k$ for which player-2 winning region is non-empty in G for the one-dimensional mean-payoff objective with weight function $w \cdot \vec{\lambda}_0$ if and only if there exists $\vec{\lambda} \in (\mathbb{Z}_M^{\pm})^k$ such that the player-2 winning region is non-empty in G for the one-dimensional mean-payoff objective with weight function $w \cdot \vec{\lambda}$.*

Proof. Suppose that player 2 has a memoryless winning strategy σ in G from an initial vertex v_0 for the one-dimensional mean-payoff objective with weight function $w \cdot \vec{\lambda}_0$. Let $C_1, \ldots, C_m$ be the simple cycles that are reachable from v_0 in the graph G^σ. Since σ is a player-2 winning strategy it follows that $w(C_i)^T \cdot \vec{\lambda}_0 < 0$ for every $i \in \{1, \ldots, m\}$. We note that for all $1 \leq i \leq m$ we have $w(C_i) \in (\mathbb{Z}^{\pm}_{n \cdot W})^k$ (since C_i is a simple cycle, in every dimension the sum of the weights is between $-n \cdot W$ and $n \cdot W$). Then by [35, Lemma 2, items c and d] it follows that there is a vector of integers $\vec{\lambda}$ such that $w(C_i)^T \cdot \vec{\lambda} \leq -1 < 0$, for all $1 \leq i \leq m$; and $\vec{\lambda} \in (\mathbb{Z}^{\pm}_M)^k$. Since all the reachable cycles from v_0 in G^σ are negative according to $w \cdot \vec{\lambda}$, we get that σ is a winning strategy for the one-dimensional mean-payoff game with weight function $w \cdot \vec{\lambda}$; and hence the proof for the direction from left to right follows. The proof for the converse direction is trivial. □

The next lemma removes the two assumptions of Lemma 1 and strengthens Lemma 2.

Lemma 4. *Let G be a game graph with a multidimensional mean-payoff objective with a weight function w. The following assertions hold: (1) $\bigcup_{\vec{\lambda} \in (\mathbb{Z}^{\pm}_M)^k} \mathsf{Win}^2_{\vec{\lambda}} \subseteq \mathsf{Win}^2$. (2) If $\bigcup_{\vec{\lambda} \in (\mathbb{Z}^{\pm}_M)^k} \mathsf{Win}^2_{\vec{\lambda}} = \emptyset$, then $\mathsf{Win}^2 = \emptyset$.*

Proof. We first show how to construct a game graph $\widehat{G}$ from G that satisfies the two assumptions (Assumption 1 and Assumption 2) and has the same winning regions (for the multidimensional objective) as in G.

1. *(Assumption 1).* Given any game graph G there exists a linear transformation to satisfy Assumption 1 by simply adding a dummy vertex for every outgoing edge of a player 2 vertex (i.e., for every edge $e = (u, v)$ with $u, v \in V_2$, we add a vertex e, edges (u, e) with weight $w(e)$ and (e, v) with weight $\vec{0}$, and e is a player-1 vertex with a single outgoing edge).
2. *(Assumption 2).* First, recall that adding several self-loop edges creates a multi-graph, but a dummy player-2 vertex can be put for every such edge to ensure that we do not have a multi-graph. Second we observe that adding the self-loop edges of Assumption 2 do not affect winning for player 1, as if there is a winning strategy for player 1, then there is one that never chooses the self-loop edges of Assumption 2 because the self-loop edges are non-positive in every dimension and negative in one dimension.

For a game graph G we denote by $\widehat{G}$ the graph that is formed by the transformations above. We now establish the following claim:

Claim. The following two properties hold for the game graph $\widehat{G}$: (i) if a vector $\vec{\lambda}$ is non-positive in (at least) one dimension, then player-2 winning region in $\widehat{G}$ for the one-dimensional mean-payoff objective with weight function $w \cdot \vec{\lambda}$ is empty; and (ii) if a vector $\vec{\lambda}$ is positive in all dimensions, then player-2 winning region in G and in $\widehat{G}$ is the same for the one-dimensional mean-payoff objective with weight function $w \cdot \vec{\lambda}$. The first item of the claim holds due to the self-loops of Assumption 2, and Assumption 1 ensures that a player-1 vertex is reached within two steps (the same reasoning as used in Lemma 1). The second item of the claim holds because the weight of any simple cycle in G is the same as in $\widehat{G}$, and the weight of Assumption 2 self-loops are non-positive in every dimension and negative in one dimension (since $\vec{\lambda}$ is positive in all dimensions). We note that since the winning condition only requires non-negative mean-payoff then only the weights of the cycles matter and not their length. Hence, a memoryless winning strategy in G is also winning in $\widehat{G}$ and vice-versa.

We now prove the two assertions of the lemma.

1. *(First assertion).* Consider that in G we have $v \in \mathsf{Win}^2_{\vec{\lambda}}$, for some vertex v and a vector $\vec{\lambda} \in (\mathbb{Z}^{+}_M)^k$. Then by the second item of the claim we get that $v \in \mathsf{Win}^2_{\vec{\lambda}}$ also in $\widehat{G}$, and then by Lemma 1 we get that $v \in \mathsf{Win}^2$ (in $\widehat{G}$). Finally, by the definition of the transformations, we get that player 2 is winning from v for the multidimensional mean-payoff objective in $\widehat{G}$ if and only if player 2 is winning from v for the multidimensional mean-payoff objective in G. Thus $v \in \mathsf{Win}^2$ in G and the first assertion follows.
2. *(Second assertion).* For the second assertion consider that $\mathsf{Win}^2 \neq \emptyset$ (in G) and we show that for some $\vec{\lambda} \in (\mathbb{Z}^{+}_M)^k$ we have $\mathsf{Win}^2_{\vec{\lambda}} \neq \emptyset$ (in G). Suppose that $v \in \mathsf{Win}^2$ for some vertex v in G. Then by the definition of the transformation we have that $v \in \mathsf{Win}^2$ also in $\widehat{G}$. By Lemma 2 and Lemma 3 it follows that there is $\vec{\lambda} \in (\mathbb{Z}^{\pm}_M)^k$ such that $v \in \mathsf{Win}^2_{\vec{\lambda}}$ (in $\widehat{G}$). By the first item of the claim we get that $\vec{\lambda} \in (\mathbb{Z}^{+}_M)^k$. Finally, by the second item of the claim, and since $\vec{\lambda} \in (\mathbb{Z}^{+}_M)^k$, we get that $v \in \mathsf{Win}^2_{\vec{\lambda}}$ also in G, and thus the second assertion follows.

The desired result follows. □

To use the result of Lemma 4 iteratively to solve finite-state games with multidimensional mean-payoff objectives, we need the notion of *attractors*. For a set U of vertices, $Attr_2(U)$ is defined inductively as follows: $U_0 = U$ and for all $i \geq 0$ we have $U_{i+1} = U_i \cup \{v \in V_1 \mid \mathsf{Out}(v) \subseteq U_i\} \cup \{v \in V_2 \mid \mathsf{Out}(v) \cap U_i \neq \emptyset\}$, and $Attr_2(U) = \bigcup_{i \geq 0} U_i$. Intuitively, from U_{i+1} player 2 can ensure to reach U_i in one step against all strategies of player 1, and thus $Attr_2(U)$ is the set of vertices such that player 2 can ensure to reach U against all strategies of player 1 in finitely many steps. The set $Attr_2(U)$ can be computed

in linear time [3,29]. Observe that if G is a game graph, then for all U, the game graph induced by the set $V \setminus Attr_2(U)$ is also a game graph (i.e., all vertices in $V \setminus Attr_2(U)$ have outgoing edges in $V \setminus Attr_2(U)$). The following lemma shows that in multidimensional mean-payoff games, if U is a set of vertices such that player 2 has a winning strategy from every vertex in U, then player 2 has a winning strategy from all vertices in $Attr_2(U)$, and we can recurse in the game graph after removal of $Attr_2(U)$.

Lemma 5. *Consider a multidimensional mean-payoff game G with weight function w. Let U be a set of vertices such that from all vertices in U there is a winning strategy for player 2. Then the following assertions hold: (1) From all vertices in $Attr_2(U)$ there is a winning strategy for player 2. (2) Let Z be the set of vertices from which player 2 has a winning strategy in the game graph induced after the removal of $Attr_2(U)$. Then from all vertices in Z, player 2 has a winning strategy in the original game graph.*

Proof. The proof of the first item is as follows: from vertices in $Attr_2(U)$ first consider a strategy to ensure to reach U (within finitely many steps), and once U is reached switch to a winning strategy from vertices in U. The proof of second item is as follows: fix a winning strategy in the remaining game graph for vertices in Z and a winning strategy from $Attr_2(U)$ for player 2. Consider any counter strategy for player 1. If $Attr_2(U)$ is ever reached, then the winning strategy from $Attr_2(U)$ ensures winning for player 2, and otherwise the winning strategy of the remaining game graph ensures winning. □

Algorithm. We now present our iterative algorithm that is based on Lemma 4 and Lemma 5. In the current iteration i of the game graph execute the following steps: sequentially iterate over vectors $\vec{\lambda} \in (\mathbb{Z}_M^+)^k$; and if for some $\vec{\lambda}$ we obtain a non-empty set U of winning vertices for player 2 for the one-dimensional mean-payoff objective with weight function $w \cdot \vec{\lambda}$ in the current game graph, remove $Attr_2(U)$ from the current game graph and proceed to iteration $i+1$. Otherwise if for all $\vec{\lambda} \in (\mathbb{Z}_M^+)^k$, player 1 wins from all vertices for the one-dimensional mean-payoff objective with weight function $w \cdot \vec{\lambda}$, then the set of current vertices is the set of winning vertices for player 1. The correctness of the algorithm follows from Lemma 4 and Lemma 5.

Complexity. The algorithm has at most n iterations, and each iteration solves at most $O(M^k)$ one-dimensional mean-payoff games (where k is the dimension of the weight function). Thus the iterative algorithm requires to solve $O(n \cdot M^k)$ one-dimensional mean-payoff games with m edges, n vertices, and the maximal weight is at most $k \cdot W \cdot M$ (as the maximum coefficient of $\vec{\lambda}$ is M, the maximum weight in every dimension is W and we sum k dimensions). Since one-dimensional mean-payoff games with n vertices, m edges, and maximal weight W can be solved in time $O(n \cdot m \cdot W)$ [10], we obtain the following result.

Theorem 1. *The set of winning vertices for player 1 in a multidimensional mean-payoff game with n vertices, m edges, k-dimensions, and maximal absolute weight W can be computed in time $O(n^2 \cdot m \cdot k \cdot W \cdot (k \cdot n \cdot W)^{k^2+2 \cdot k+1})$.*

Comparison with convex-hull techniques. Previous works [1,13,39] addressed multidimensional mean-payoff automata that correspond to a game in which only Player 1 makes decisions (i.e., every player-2 state has out-degree one). Their solution is based on enumerating all simple cycle average weight vectors in the game and compute their convex-hull. If the graph is strongly connected, then player 1 can achieve any mean-payoff vector that resides in the convex-hull. In [41] it is also shown that the enumeration process can be replaced with a construction of a linear-programming problem that is solvable in polynomial time. However, it is not clear how to scale the linear-programming construction for a two-player game. It is also not clear how to enumerate all possible simple cycles in a two-player game and in any case such an enumeration will yield an exponential time algorithm. Our hyperplane-separation technique replace the explicit enumeration by checking whether all the simple cycle weight vectors that Player 1 can obtain reside below a certain hyperplane that contain the origin. If this is the case, then the convex-hull does not contain the origin and it is impossible to have a non-negative mean-payoff across all dimensions.

3. Pushdown graphs with multidimensional mean-payoff objectives

In this section we consider pushdown graphs (or pushdown systems) with multidimensional mean-payoff objectives, and we give an algorithm that determines if there exists a path that satisfies a multidimensional objective. The algorithm we propose runs in polynomial time even for arbitrary number of dimensions and for arbitrary weight function. As in the previous section, we use the hyperplane separation technique to reduce the problem into a one-dimensional pushdown graph, and a polynomial solution for the latter is known [17].

Key obstacles and overview of the solution. We first describe the key obstacles for the polynomial time algorithm to solve pushdown graphs with multidimensional mean-payoff objectives (as compared to finite-state graphs and finite-state games). For pushdown graphs we need to overcome the next three main obstacles: (a) The mean-payoff value of a finite-state graph is uniquely determined by the weights of the simple cycles of the graph. However, for pushdown graphs it is also possible to *pump* special types of acyclic paths. Hence, we first need to characterize the *pumpable paths* that uniquely

determine the possible mean-payoff vectors in a pushdown graph. (b) Lemma 2 does not hold for arbitrary infinite-state graphs and we need to show that it does hold for pushdown graphs. (c) We require an algorithm to decide whether there is a hyperplane such that all the weights of the pumpable paths of a pushdown graph lie below the hyperplane (also for arbitrary dimensions). The overview of our solutions to the above obstacles are as follows: (a) In the first part of the section (until Proposition 1) we present a characterization of the pumpable paths in a pushdown graph. (b) We use Gordan's Lemma [26] (a special case of Farkas' Lemma) and in Lemma 11 we prove that Lemma 1 and Lemma 2 hold also for pushdown graphs (Lemma 1 holds for any infinite-state graph). (c) Conceptually, we find the separating hyperplane by constructing a matrix A, such that every row in A is a weight vector of a pumpable path, and we solve the linear inequality $\vec{\lambda} \cdot A < \vec{0}$. However, in general the matrix A can be of exponential size. Thus we need to use advanced linear-programming technique that solves in polynomial time linear inequalities with polynomial number of variables and exponential number of constraints. This technique requires a polynomial-time oracle that for a given $\vec{\lambda}$ returns a violated constraint (or says that all constraints are satisfied). We show that in our case the required oracle is the algorithm for pushdown graphs with one-dimensional mean-payoff objective (which we obtain from [17]), and thus we establish a polynomial-time hyperplane separation technique for pushdown graphs.

Stack alphabet and commands. We start with the basic notion of stack alphabet and commands. Let Γ denote a finite set of *stack alphabet*, and $\mathsf{Com}(\Gamma) = \{skip, pop\} \cup \{push(z) \mid z \in \Gamma\}$ denotes the set of *stack commands* over Γ. Intuitively, the command *skip* does nothing, *pop* deletes the top element of the stack, and $push(z)$ puts z on the top of the stack. For a stack command *com* and a stack string $\alpha \in \Gamma^+$ we denote by $com(\alpha)$ the stack string obtained by executing the command *com* on α, i.e., when performing the command *com* when the value of the stack is α (in a stack string the top denotes the right end of the string).

Multi-weighted pushdown systems. A *multi-weighted pushdown system (WPS)* (or a multi-weighted pushdown graph) is a tuple:

$$\mathcal{A} = \langle Q, \Gamma, q_0 \in Q, E \subseteq (Q \times \Gamma) \times (Q \times \mathsf{Com}(\Gamma)), w : E \to \mathbb{Z}^k \rangle,$$

where Q is a finite set of *states* with q_0 as the initial state; Γ the finite *stack alphabet* and we assume there is a special initial stack symbol $\bot \in \Gamma$; E describes the set of edges or transitions of the pushdown system; and w is a weight function that assigns an integer weight vector to every edge; we denote by w_i the projection of w to the i-th dimension. We assume that $\bot$ can be neither put on nor removed from the stack. A *configuration* of a WPS is a pair (α, q) where $\alpha \in \Gamma^+$ is a stack string and $q \in Q$. For a stack string α we denote by $\mathsf{Top}(\alpha)$ the top symbol of the stack. The initial configuration of the WPS is $(\bot, q_0)$. We use W to denote the maximal absolute weight of the edge weights.

Successor configurations and runs. Given a WPS $\mathcal{A}$, a configuration $c_{i+1} = (\alpha_{i+1}, q_{i+1})$ is a *successor* configuration of a configuration $c_i = (\alpha_i, q_i)$, if there is an edge $(q_i, \gamma_i, q_{i+1}, com) \in E$ such that $com(\alpha_i) = \alpha_{i+1}$, where $\gamma_i = \mathsf{Top}(\alpha_i)$. A *path* π is a sequence of configurations. A path $\pi = \langle c_1, \ldots, c_{n+1} \rangle$ is a *valid path* if for all $1 \le i \le n$ the configuration c_{i+1} is a successor configuration of c_i (and the notation is similar for infinite paths). In the sequel we shall refer only to valid paths. Let $\pi = \langle c_1, c_2, \ldots, c_i, c_{i+1}, \ldots \rangle$ be a path. We denote by $\pi[j] = c_j$ the j-th configuration of the path and by $\pi[i_1, i_2] = \langle c_{i_1}, c_{i_1+1}, \ldots, c_{i_2} \rangle$ the segment of the path from the i_1-th to the i_2-th configuration. A path can equivalently be defined as a sequence $\langle c_1 e_1 e_2 \ldots e_n \rangle$, where c_1 is a configuration and e_i are valid transitions. Our goal is to obtain an algorithm that given a WPS $\mathcal{A}$ decides if there exists an infinite path π in $\mathcal{A}$ from q_0 such that $\mathsf{LimAvg}(\pi) \ge \vec{0}$.

Notations. We shall use (i) γ or γ_i for an element of Γ; (ii) e or e_i for a transition (equivalently an edge) from E; (iii) α or α_i for a string from Γ^*. For a path $\pi = \langle c_1, c_2, \ldots \rangle = \langle c_1 e_1 e_2 \ldots \rangle$ we denote by (i) q_i: the state of configuration c_i, and (ii) α_i: the stack string of configuration c_i.

Stack height and additional stack height of paths. For a path $\pi = \langle (\alpha_1, q_1), \ldots, (\alpha_n, q_n) \rangle$, the *stack height* of π is the maximal height of the stack in the path, i.e., $\mathsf{SH}(\pi) = \max\{|\alpha_1|, \ldots, |\alpha_n|\}$. The *additional stack height* of π is the additional height of the stack in the segment of the path, i.e., the additional stack height $\mathsf{ASH}(\pi)$ is $\mathsf{SH}(\pi) - \max\{|\alpha_1|, |\alpha_n|\}$.

Pumpable pair of paths. Let $\pi = \langle c_1 e_1 e_2 \ldots \rangle$ be a finite or infinite path. A *pumpable pair of paths* for π is a pair of non-empty sequences of edges in π: $(p_1, p_2) = (e_{i_1} e_{i_1+1} \ldots e_{i_1+n_1}, e_{i_2} e_{i_2+1} \ldots e_{i_2+n_2})$, for $n_1, n_2 \ge 0$, $i_1 \ge 0$ and $i_2 > i_1 + n_1$ such that for every $j \ge 0$ the path $\pi^j_{(p_1,p_2)}$ obtained by pumping the pair of paths p_1 and p_2 for j times each is a valid path, i.e., for every $j \ge 0$ we have

$$\pi^j_{(p_1,p_2)} = \langle c_1 e_1 e_2 \ldots e_{i_1-1} (e_{i_1} e_{i_1+1} \ldots e_{i_1+i_n})^j e_{i_1+i_n+1} \ldots e_{i_2-1} (e_{i_2} e_{i_2+1} \ldots e_{i_2+n_2})^j e_{i_2+n_2} \ldots \rangle$$

is a valid path.

We will show that large additional stack height implies the existence of pumpable pair of paths. To prove the results we need the notion of *local minima* of paths.

Local minima of a path. Let $\pi = \langle c_1, c_2, \ldots \rangle$ be a path. A configuration $c_i = (\alpha_i, q_i)$ is a *local minima* if for every $j \ge i$ we have $\alpha_i \sqsubseteq \alpha_j$ (i.e., the stack string α_i is a prefix string of α_j). One basic fact is that every infinite path has infinitely many local minima. We discuss the proof of the basic fact and some properties of local minima. Consider a path $\pi = \langle c_1, c_2, \ldots \rangle$.

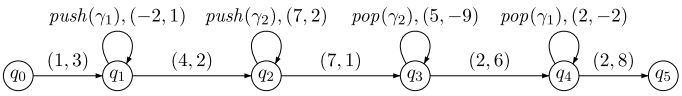

Fig. 7. A WPS $\mathcal{A}$. If an edge is not labeled with a command, then the command is *skip*. The label $pop(\gamma)$ stands for: if the top symbol is γ, then a pop transition is possible.

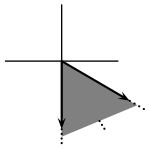

Fig. 8. $\mathsf{PCone}(\pi)$.

If there is a finite integer j such that from some point on (say after i-th index) the stack height is always at least j, and the stack height is j infinitely often, then every configuration after i-th index with stack height j is a local minima (and there are infinitely many of them). Otherwise, for every integer j, there exists an index i, such that for every index after i the stack height exceeds j, and then for every j, the last configuration with stack height j is a local minima and we have infinitely many local minima. This shows the basic fact about infinitely many local minima of a path. We now discuss a property of consecutive local minima in a path. For every two consecutive local minima c_i and c_j in the sequence of all local minima of a path: either c_i and c_j have the same stack height, or else c_j is obtained from c_i with a sequence that contains one push operation and no pop operations.

Non-decreasing paths and cycles, and proper cycles. A path from configuration $(\alpha\gamma, q_1)$ to configuration $(\alpha\gamma\alpha_2, q_2)$ is a *non-decreasing $\alpha\gamma$-path* if $(\alpha\gamma, q_1)$ is a local minima. Note that if π is a non-decreasing $\alpha\gamma$-path for some $\alpha \in \Gamma^*$, then the same sequence of transitions leads to a non-decreasing $\beta\gamma$-path for every $\beta \in \Gamma^*$. Hence we say that π is a non-decreasing path if there exists $\alpha \in \Gamma^*$ and $\gamma \in \Gamma \cup \{\bot\}$ such that π is a non-decreasing $\alpha\gamma$-path. A *non-decreasing cycle* is a non-decreasing path from (α_1, q) to (α_2, q) such that the top symbols of α_1 and α_2 are the same. A non-decreasing cycle from (α_1, q) to (α_2, q) is a *proper cycle* if $\alpha_1 = \alpha_2$ (i.e., returns to the same configuration). By convention, when we say that a path π is a non-decreasing path from (γ_1, q_1) to (γ_2, q_2), it means that for some $\alpha_1, \alpha_2 \in \Gamma^*$, the path π is a non-decreasing path from $(\alpha_1\gamma_1, q_1)$ to $(\alpha_1\gamma_1\alpha_2\gamma_2, q_2)$.

Cone of pumpable pairs. We let $\mathbb{R}_+ = [0, +\infty)$. For a finite non-decreasing path π we denote by $\mathsf{PPS}(\pi)$ the (finite) set of pumpable pairs that occur in π, that is, $\mathsf{PPS}(\pi) = \{(p_1, p_2) \in (E^* \times E^*) \mid p_1 \text{ and } p_2 \text{ are a pumpable pair in } \pi\}$. Let $\mathsf{PPS}(\pi) = \{P_1 = (p_1^1, p_2^1), P_2 = (p_1^2, p_2^2), \ldots, P_j = (p_1^j, p_2^j)\}$, and we denote by $\mathsf{PumpMat}(\pi)$ the matrix that is formed by the weight vectors of the pumpable pairs of π, that is, the matrix has j rows and the i-th row of the matrix is $w(p_1^i) + w(p_2^i)$ (every weight vector is a row in the matrix). We denote by $\mathsf{PCone}(\pi)$ the cone of the weight vectors in $\mathsf{PPS}(\pi)$, formally, $\mathsf{PCone}(\pi) = \{\mathsf{PumpMat}(\pi) \cdot \vec{x} \mid \vec{x} \in (\mathbb{R}_+^k \setminus \{\vec{0}\})\}$.

Example. We illustrate the definitions with the aid of an example. Consider the WPS shown in Fig. 7. Consider all the possible paths from $(\bot, q_0)$ to $(\bot, q_5)$. Every such path is of the form

$$q_0 \to q_1 \to (q_1 \to q_1)^m \to q_2 \to (q_2 \to q_2)^n \to q_3 \to (q_3 \to q_3)^n \to q_4 \to (q_4 \to q_4)^m \to q_5$$

for some non-negative numbers m and n. Let π be the path that is formed by taking $m = n = 1$. In π there are two pumpable pairs, namely, $P_1 = (q_1 \to q_1, q_4 \to q_4)$ and $P_2 = (q_2 \to q_2, q_3 \to q_3)$. Given the weight function w (as shown in the figure) we have $w(P_1) = (0, -1)$ and $w(P_2) = (12, -7)$. Therefore we have the following:

- $\mathsf{PPS}(\pi) = \{(q_1 \to q_1, q_4 \to q_4), (q_2 \to q_2, q_3 \to q_3)\}$;
- $\mathsf{PumpMat}(\pi) = \begin{pmatrix} 0 & -1 \\ 12 & -7 \end{pmatrix}$; and
- $\mathsf{PCone}(\pi) = \{x_1 \cdot (0, -1) + x_2 \cdot (12, -7) \mid x_1, x_2 \geq 0 \wedge x_1 + x_2 > 0\}$ (see Fig. 8).

The example illustrates the various concepts we have introduced.

Notations and abbreviations. Fix $\ell = (|Q| \cdot |\Gamma|)^{(|Q| \cdot |\Gamma|)^2 + 1}$ for the rest of the section. For $q_1, q_2 \in Q$ and $\gamma_1, \gamma_2 \in \Gamma$, by abuse of notation we denote by $\mathsf{PPS}((\gamma_1, q_1), (\gamma_2, q_2))$ the (finite) set of all pumpable pairs of paths, not longer than ℓ, that occur in a non-decreasing path from (γ_1, q_1) to (γ_2, q_2); we similarly define $\mathsf{PumpMat}((\gamma_1, q_1), (\gamma_2, q_2))$ and $\mathsf{PCone}((\gamma_1, q_1), (\gamma_2, q_2))$. If $q_1 = q_2$ and $\gamma_1 = \gamma_2$, then we abbreviate $\mathsf{PPS}((\gamma_1, q_1), (\gamma_1, q_1))$ by $\mathsf{PPS}((\gamma_1, q_1))$, and similarly for PumpMat and PCone. The next lemma was proved in [17].

Lemma 6 *([17]). Let π be a finite path such that $\mathsf{ASH}(\pi) > (|Q| \cdot |\Gamma|)^2$. Then π has a pumpable pair of paths.*

In the next lemma we show that any sufficiently long non-decreasing path contains a pumpable pair of paths.

Lemma 7. *Every non-decreasing path longer than ℓ has a pumpable pair of paths.*

Proof. Let π be a non-decreasing path longer than ℓ. If $\mathsf{ASH}(\pi) > (|Q| \cdot |\Gamma|)^2$, then by Lemma 6 we get the desired result; otherwise, it is an easy observation that π contains a proper cycle, which is by definition a pumpable pair of paths (where both paths of the pair are the proper cycle). □

Corollary 1. *Every non-decreasing path longer than ℓ has a pumpable pair of paths with length at most ℓ.*

The next two lemmas show basic properties of PPS. The first lemma asserts that we can decompose every non-decreasing path to a set of pumpable pairs and a short non-decreasing path.

Lemma 8. *For every non-decreasing path π from (γ_1, q_1) to (γ_2, q_2) there exists a tuple of pumpable pairs of paths $P_1 = (p_1^1, p_2^1), P_2 = (p_1^2, p_2^2), \ldots, P_j = (p_1^j, p_2^j) \in \mathsf{PPS}((\gamma_1, q_1), (\gamma_2, q_2))$ each of length at most ℓ (i.e., for all $1 \le i \le j$ we have $|P_i| \le \ell$), a finite non-decreasing path π_0 from (γ_1, q_1) to (γ_2, q_2) with length at most ℓ, and non-negative constants $m_1, \ldots, m_j$ such that $w(\pi) = w(\pi_0) + \sum_{i=1}^{j} m_i \cdot w(P_i)$ and $|\pi| = |\pi_0| + \sum_{i=1}^{j} m_i \cdot |P_i|$.*

Proof. The proof is by induction on the length of π. If $|\pi| \le \ell$, then we are done by choosing $j = 0$ and $\pi_0 = \pi$. Otherwise, by Corollary 1, the path has a pumpable pair $P = (p_1, p_2)$ with length less than ℓ (and hence $P \in \mathsf{PPS}((q_1, \gamma_1), (q_2, \gamma_2))$). Let π' be the path that is obtained from π by pumping P zero times (i.e., π' is obtained by omitting P from π); clearly π' is a non-decreasing path from (γ_1, q_1) to (γ_2, q_2) and shorter than π, any by the induction hypothesis we get the desired result. □

The following lemma shows the connection between the average weight of a path and PPS.

Lemma 9. *If $\mathsf{PCone}((\gamma_1, q_1), (\gamma_2, q_2)) \cap \mathbb{R}^k_+ = \emptyset$, then there exist constants $\epsilon > 0$ and $m \in \mathbb{N}$, such that for every finite non-decreasing path π from (γ_1, q_1) to (γ_2, q_2), there exists a dimension t such that $w_t(\pi) \le m - \epsilon \cdot |\pi|$.*

Proof. In order to define ϵ, we consider the following linear programming problem with the variables $x_1, x_2, \ldots$ and r: the objective function is to maximize r subject to the constraints below

$$\sum_{z \in \mathsf{PPS}((\gamma_1, q_1), (\gamma_2, q_2))} x_z \cdot w_t(z) \ge r \qquad \text{for } t = 1, \ldots, k \tag{1}$$

$$\sum_{z \in \mathsf{PPS}((\gamma_1, q_1), (\gamma_2, q_2))} x_z = 1 \tag{2}$$

$$x_z \ge 0 \qquad \text{for all } z \in \mathsf{PPS}((\gamma_1, q_1), (\gamma_2, q_2)) \tag{3}$$

Intuitively, the first constraint specifies that there is a convex combination of the weights of the pumpable pairs to ensure at least r in every dimension; and the following two constraints ensure that it is a convex combination. As the domain of the variables is closed and bounded, there exists a maximum value to the linear program, and let r^* be the maximum value. If $r^* \ge 0$, then we get a contradiction to the assumption that $\mathsf{PCone}((\gamma_1, q_1), (\gamma_2, q_2)) \cap \mathbb{R}^k_+ = \emptyset$. Hence we have $r^* < 0$. We define $m = (\ell + 1) \cdot W - r^*$, $\epsilon = -\frac{r^*}{\ell}$ and we claim that for every non-decreasing path π from (γ_1, q_1) to (γ_2, q_2) there is a dimension t such that $w_t(\pi) \le m - \epsilon \cdot |\pi|$.

By Lemma 8, there exists a path π_0 with length at most ℓ, a (finite) sequence of pumpable pairs $P_1, \ldots, P_j \in \mathsf{PPS}((\gamma_1, q_1), (\gamma_2, q_2))$ each of length at most ℓ and constants $m_1, \ldots, m_j$ such that $w(\pi) = w(\pi_0) + \sum_{i=1}^{j} m_i \cdot w(P_i)$ and $|\pi| = |\pi_0| + \sum_{i=1}^{j} m_i \cdot |P_i|$. We define $M = \sum_{i=1}^{j} m_i$. As all $|P_i|$ and $|\pi_0|$ are bounded by ℓ, we get that $M \ge \frac{|\pi| - \ell}{\ell}$. Observe that if we set $x_i = \frac{m_i}{M}$ for $i = 1$ to j, and let $x_z = 0$ for all other $z \in \mathsf{PPS}((\gamma_1, q_1), (\gamma_2, q_2))$, then they satisfy the constraints for convex combination. Hence there must exist a dimension t for which $\frac{1}{M} \sum_{i=1}^{j} m_i \cdot w_t(P_i) \le r^*$ (since r^* is the maximum among the feasible solutions). Thus $w_t(\pi) \le w_t(\pi_0) + M \cdot r^*$ and since $r^* < 0$ we have

$$w_t(\pi) \le w_t(\pi_0) + \frac{|\pi| - \ell}{\ell} \cdot r^* = w_t(\pi_0) - r^* + |\pi| \cdot \frac{r^*}{\ell}.$$

Therefore, for the choice of $m \ge \ell \cdot W - r^*$ and $\epsilon = -\frac{r^*}{\ell}$, we obtain the desired result. □

The next proposition gives a sufficient and necessary condition for the existence of a path with non-negative mean-payoff values in all the dimensions.

Proposition 1. *There exists an infinite path π such that $\mathsf{LimInfAvg}(\pi) \geq \vec{0}$ if and only if there exists a (reachable) non-decreasing cycle π such that $\mathbb{R}^k_+ \cap \mathsf{PCone}(\pi) \neq \emptyset$.*

Proof. We first prove the direction from right to left. If there exists a path π such that $\mathbb{R}^k_+ \cap \mathsf{PCone}(\pi) \neq \emptyset$, then by definition there are j pumpable pairs $P_1, P_2, \ldots, P_j$ with weight vectors $y_1 = w(P_1), \ldots, y_j = w(P_j)$ such that there exist j positive constants (w.l.o.g. natural positive constants) $n_1, \ldots, n_j$ such that $\sum_{i=1}^{j} n_i \cdot y_i \geq \vec{0}$. For every $a, b \in \mathbb{N}$ we denote by $\pi^{a,b}$ the (finite) path that is formed by pumping the a-th pumpable pair b times. We denote by $\widehat{\pi}^b = \pi^{1,b\cdot n_1} \cdot \pi^{2,b\cdot n_2} \cdot \ldots \cdot \pi^{j,b\cdot n_j} \cdots$, where the i-th pumpable pair is pumped $b \cdot n_i$ times, respectively. We note that $\pi^{a,b}$ is a non-decreasing cycle, and for the infinite path

$$\pi^* = \widehat{\pi}^1 \cdot \widehat{\pi}^2 \cdot \widehat{\pi}^3 \cdots \widehat{\pi}^b \cdots$$

we get that π^* is a valid path and $\mathsf{LimAvg}(\pi^*) \geq \vec{0}$. The reason we have $\mathsf{LimAvg}(\pi^*) \geq \vec{0}$ is as b tends to infinity, the average weight is determined only by the weights of the j pumpable pairs and their coefficients $n_1, \ldots, n_j$, and we have $\sum_{i=1}^{j} n_i \cdot y_i \geq \vec{0}$. This completes the proof for the direction from right to left.

For the converse direction, let π be an infinite path such that $\mathsf{LimAvg}(\pi) \geq \vec{0}$, and let (γ, q) be a top configuration that occurs infinitely often in the local minima of π. Since $\mathsf{LimAvg}(\pi) \geq \vec{0}$ it follows that for every $\epsilon > 0$ there exists a non-decreasing cycle that begins at (γ, q) with average weight at least $-\epsilon$ in every dimension. Hence, by Lemma 9 it follows that $\mathsf{PCone}(\gamma, q) \cap \mathbb{R}^k_+ \neq \emptyset$, and hence, there exists a non-decreasing cycle π that starts in (γ, q) for which $\mathsf{PCone}(\pi) \cap \mathbb{R}^k_+ \neq \emptyset$. □

By Proposition 1, we can decide whether there is an infinite path π for which $\mathsf{LimAvg}(\pi) \geq \vec{0}$ by checking if there exists a tuple $(\gamma, q) \in \Gamma \times Q$ for which there is a non-negative (and non-trivial) solution for the equation $\mathsf{PumpMat}((\gamma, q)) \cdot \vec{x} \geq 0$. As in Lemma 1 by adding k self-loop transitions with weights, where the weight of transition i is -1 in the i-th dimension and 0 in the other dimensions, we reduce the problem to finding q and γ such that there is a non-negative solution for $\mathsf{PumpMat}((\gamma, q)) \cdot \vec{x} = 0$. Inspired by the techniques of [20], we present an algorithm that solves the problem by a reduction to a corresponding one-dimensional problem. As before given a k-dimensional weight function w and a k-dimensional vector $\vec{\lambda}$ we denote by $w \cdot \vec{\lambda}$ the one-dimensional weight function obtained by multiplying the weight vectors by $\vec{\lambda}$. The reduction to one-dimensional objective requires the use of Gordan's lemma.

Lemma 10 *(Gordan's Lemma [26] (see also Lemma 2 in [35])). For a matrix A, either $A \cdot \vec{x} = \vec{0}$ has a non-trivial non-negative solution for $\vec{x}$, or there exists a vector $\vec{y}$ such that $\vec{y} \cdot A^T$ is negative in every dimension.*

The next lemma suggests that we can reduce the multidimensional problem to a corresponding one-dimensional problem.

Lemma 11. *Given a WPS $\mathcal{A}$ with a k-dimensional weight function w, and $(\gamma, q) \in \Gamma \times Q$, there exists a non-trivial non-negative solution for $\mathsf{PumpMat}((\gamma, q)) \cdot \vec{x} = \vec{0}$ if and only if for every $\vec{\lambda} \in \mathbb{R}^k$ there is a non-decreasing path from (γ, q) to (γ, q) that contains a pumpable pair $P = (p_1, p_2)$ such that $(w \cdot \vec{\lambda})(P) \geq 0$ (i.e., the weight of the path for one-dimensional weight function $w \cdot \vec{\lambda}$ is non-negative).*

Proof. The proof is straightforward application of Gordan's Lemma to the matrix $\mathsf{PumpMat}((\gamma, q))$. □

Proposition 2. *There is a polynomial time algorithm that given a WPS $\mathcal{A}$ with k-dimensional weight function w, $(\gamma, q) \in \Gamma \times Q$, a vector $\vec{\lambda} \in \mathbb{Q}^k$, and a rational number $r \in \mathbb{Q}$ decides if there exists a pumpable pair of paths P in a non-decreasing cyclic path that begins at (γ, q) in $\mathcal{A}$, with $\frac{(w\cdot\vec{\lambda})(P)}{|P|} > r$ and $|P| \leq \ell$, and if such pair exists, it returns $\frac{w(P)}{|P|}$.*

Intuitively, the algorithm for Proposition 2 is based on the algorithm for solving WPSs with one-dimensional mean-payoff objectives. We postpone the technically detailed proof to Section 3.1. We first show how to use the result of the proposition and a result from linear programming to solve the problem. We first state the result for linear programming.

Linear program with exponential constraints and polynomial-time separating oracle. Consider a linear program over n variables and exponentially many constraints in n. Given a polynomial time *separating oracle* that for every point in space returns in polynomial time whether the point is feasible, and if infeasible returns a violated constraint, the linear program can be solved in polynomial time using the ellipsoid method [27]. We use the result to show the following result.

Proposition 3. *There exists a polynomial time algorithm that decides whether for a given state q and a stack alphabet symbol γ there exists a non-trivial non-negative solution for $\mathsf{PumpMat}((\gamma, q)) \cdot \vec{x} = \vec{0}$.*

Proof. Conceptually, given q and γ, we compute a matrix A, such that each row in A corresponds to the average weight vector of a row in $\mathsf{PumpMat}((\gamma, q))$ (that is, the weight of a pumpable pair divided by its length), and solves the following linear programming problem: For variables r and $\vec{\lambda} = (\lambda_1, \ldots, \lambda_k)$, the objective function is to minimize r subject to the constraints below:

$$\vec{\lambda} \cdot A^T \leq \vec{r} \quad \text{where } \vec{r} = (r, r, \ldots, r)^T \tag{4}$$

$$\sum_{i=1}^{k} \lambda_i = 1 \tag{5}$$

Once the minimal r is computed, by Lemma 11, there exists a solution for $\mathsf{PumpMat}((\gamma, q)) \cdot \vec{x} = 0$ if and only if $r \geq 0$.

The number of rows of A in the worst case is exponential (to be precise at most $\ell \cdot (2 \cdot W \cdot \ell)^k$, since the length of the path is at most ℓ, the sum of weights is between $-W \cdot \ell$ and $W \cdot \ell$ and there are k dimensions). However, we do not enumerate the constraints of the linear programming problem explicitly but use the result of linear programs with polynomial time separating oracle. By Proposition 2 we have an algorithm that verifies the feasibility of a solution (that is, an assignment for $\vec{\lambda}$ and r) and if the solution is infeasible it returns a constraint that is not satisfied by the solution. Thus the result of Proposition 2 provides the desired polynomial-time separating oracle and we have the desired result. □

Hence, we get the following theorem.

Theorem 2. *Given a WPS $\mathcal{A}$ with multidimensional weight function w, we can decide in polynomial time whether there exists a path π such that $\mathsf{LimAvg}(\pi) \geq \vec{0}$.*

3.1. Technical detailed proof of Proposition 2

In this section we prove Proposition 2. Throughout this section, we assume WLOG that $\vec{\lambda}$ is a vector of integers and that $r = 0$. Intuitively the solution is very similar to solving WPS with one-dimensional objectives, with some technical and tedious modifications. We will present the relevant details. Let $\mathcal{A}$ be a WPS with k-dimensional weight function w, and $w \cdot \vec{\lambda}$ be the one-dimensional weight function. Let $d = (|Q| \cdot |\Gamma|)^2 + 1$. We now recall the notion of summary function as defined in [17]. In the definition of summary function below we consider the weight function $w \cdot \vec{\lambda}$.

Summary function. Let $\mathcal{A}$ be a WPS. For $\alpha \in \Gamma^*$ we define $s_\alpha : Q \times \Gamma \times Q \to \{-\infty\} \cup \mathbb{Z} \cup \{\omega\}$ as following.

1. $s_\alpha(q_1, \gamma, q_2) = \omega$ iff for every $n \in \mathbb{N}$ there exists a non-decreasing path from $(\alpha\gamma, q_1)$ to $(\alpha\gamma, q_2)$ with weight at least n.
2. $s_\alpha(q_1, \gamma, q_2) = z \in \mathbb{Z}$ iff the weight of any non-decreasing path with maximal weight from configuration $(\alpha\gamma, q_1)$ to configuration $(\alpha\gamma, q_2)$ is z.
3. $s_\alpha(q_1, \gamma, q_2) = -\infty$ iff there is no non-decreasing path from $(\alpha\gamma, q_1)$ to $(\alpha\gamma, q_2)$.

Remark 2. For every $\alpha_1, \alpha_2 \in \Gamma^*$: $s_{\alpha_1} \equiv s_{\alpha_2}$.

Due to Remark 2 it is enough to consider only $s \equiv s_\perp$. The computation of the summary function will be achieved by considering stack height bounded summary functions defined below.

Stack height bounded summary function. For every $d \in \mathbb{N}$, the *stack height bounded summary function* $s_d : Q \times \Gamma \times Q \to \{-\infty\} \cup \mathbb{Z} \cup \{\omega\}$ is defined as follows: (i) $s_d(q_1, \gamma, q_2) = \omega$ iff for every $n \in \mathbb{N}$ there exists a non-decreasing path from $(\perp\gamma, q_1)$ to $(\perp\gamma, q_2)$ with weight at least n and additional stack height at most d; (ii) $s_d(q_1, \gamma, q_2) = z$ iff the weight of the maximum weight non-decreasing path from $(\perp\gamma, q_1)$ to $(\perp\gamma, q_2)$ with additional stack height at most d is z; and (iii) $s_d(q_1, \gamma, q_2) = -\infty$ iff there is no non-decreasing path with additional stack height at most d from $(\perp\gamma, q_1)$ to $(\perp\gamma, q_2)$. Before presenting the key lemma we recall the computation of s_{i+1} from s_i that will also introduce the relevant notions required for the lemma.

Computation of s_{i+1} from s_i and $\mathcal{A}$. Let $G_{\mathcal{A}}$ be the finite weighted graph that is formed by all the configurations of $\mathcal{A}$ with stack height either one or two, that is, the vertices are of the form (α, q) where $q \in Q$ and $\alpha \in \{\perp \cdot \gamma, \perp \cdot \gamma_1 \cdot \gamma_2 \mid \gamma, \gamma_1, \gamma_2 \in \Gamma\}$. The edges (and their weights) are according to the transitions of $\mathcal{A}$: formally, (i) (Skip edges): for vertices $(\perp \cdot \alpha, q)$ we have an edge to $(\perp \cdot \alpha, q')$ iff $e = (q, \mathsf{Top}(\alpha), \mathit{skip}, q')$ is an edge in $\mathcal{A}$ (and the weight of the edge in $G_{\mathcal{A}}$ is $(w \cdot \vec{\lambda})(e)$) where $\alpha = \gamma$ or $\alpha = \gamma_1 \cdot \gamma_2$ for $\gamma, \gamma_1, \gamma_2 \in \Gamma$; (ii) (Push edges): for vertices $(\perp \cdot \gamma, q)$ we have an edge to $(\perp \cdot \gamma \cdot \gamma', q')$ iff $e = (q, \gamma, \mathit{push}(\gamma'), q')$ is an edge in $\mathcal{A}$ (and the weight of the edge in $G_{\mathcal{A}}$ is $(w \cdot \vec{\lambda})(e)$) for $\gamma, \gamma' \in \Gamma$; and (iii) (Pop edges): for vertices $(\perp \cdot \gamma \cdot \gamma', q)$ we have an edge to $(\perp \cdot \gamma, q')$ iff $e = (q, \gamma', \mathit{pop}, q')$ is an edge in $\mathcal{A}$ (and the weight of the edge in $G_{\mathcal{A}}$ is $(w \cdot \vec{\lambda})(e)$) for $\gamma, \gamma' \in \Gamma$. Intuitively, $G_{\mathcal{A}}$ allows skips, push pop pairs, and only one additional push. Note that $G_{\mathcal{A}}$ has at most $2 \cdot |Q| \cdot |\Gamma|^2$ vertices, and can be constructed in polynomial time.

For every $i \geq 1$, given the function s_i, the graph $G^i_{\mathcal{A}}$ is constructed from $G_{\mathcal{A}}$ as follows: adding edges $((\bot\gamma_1\gamma_2, q_1), (\bot\gamma_1\gamma_2, q_2))$ (if the edge does not exist already) and changing its weight to $s_i(q_1, \gamma_2, q_2)$ for every $\gamma_1, \gamma_2 \in \Gamma$ and $q_1, q_2 \in Q$. The value of $s_{i+1}(q_1, \gamma, q_2)$ is exactly the weight of the maximum weight path between $(\bot\gamma, q_1)$ and $(\bot\gamma, q_2)$ in $G^i_{\mathcal{A}}$ (with the following convention: $-\infty < z < \omega$, $z + \omega = \omega$ and $z + -\infty = \omega + -\infty = -\infty$ for every $z \in \mathbb{Z}$). If in $G^i_{\mathcal{A}}$ there is a path from $(\bot\gamma, q_1)$ to $(\bot\gamma, q_2)$ that contains a cycle with positive weight, then we set $s_{i+1}(q_1, \gamma, q_2) = \omega$. Hence, given s_i and $\mathcal{A}$, the construction of $G^i_{\mathcal{A}}$ is achieved in polynomial time, and the computation of s_{i+1} is achieved using the Bellman–Ford algorithm [21] in polynomial time (the maximum weight path is the shortest weight if we define the edge length as the negative of the edge weight). Also note that the Bellman–Ford algorithm reports cycles with positive weight (that is, negative length) which is required to set ω values of s_{i+1}. It follows that we can compute s_{i+1} given s_i and $\mathcal{A}$ in polynomial time. In the computation of the summary function s_i we also store along with $s_i(q_1, \gamma, q_2)$ the weight vector $w(P)$ and the length $|P|$ of a witness path P that is maximal weight (according to $w \cdot \vec{\lambda}$) shortest non-decreasing path from (γ, q_1) to (γ, q_2) with additional stack height at most i. We denote by $\mathsf{VECT}(s_i(q_1, \gamma, q_2))$ the tuple $(w(P), |P|)$.

Lemma 12. *Let $q_1, q_2 \in Q$, $\gamma \in \Gamma$ and $d > (|Q| \cdot |\Gamma|)^2$, such that $s_d(q_1, \gamma, q_2) > s_{d-1}(q_1, \gamma, q_2)$, and let π be the shortest non-decreasing path from $(\bot\gamma, q_1)$ to $(\bot\gamma, q_2)$ with weight $s_{d+1}(q_1, \gamma, q_2)$ and additional stack height d, then the following assertions hold:*

1. *The path π contains a pumpable pair of paths $P = (p_1, p_2)$ with $(w \cdot \vec{\lambda})(P) > 0$ with length at most ℓ.*
2. *We can compute $w(P)$, and $\frac{w(P)}{|P|}$ in polynomial time.*

Proof. The first item was proved in [17]. For the second item, we consider the graphs $G^i_{\mathcal{A}}$ as defined above. Then for $G^d_{\mathcal{A}}$, we compute (based on the summary function s_d) the maximum weight non-decreasing path ρ from $(\bot\gamma, q_1)$ to $(\bot\gamma, q_2)$. In the path ρ, we find a sub-path of the form $(\bot\gamma, z), (\bot\gamma\delta, q'), (\bot\gamma\delta, q''), (\bot\gamma, z')$, for which

- $s_d(z, \gamma, z') > s_{d-1}(z, \gamma, z')$; and
- $s_{d-1}(q', \delta, q'') > s_{d-2}(q', \delta, q'')$;

(note that by definition such sub-path must exist). We store the value of the maximum weight paths from $(\bot\gamma, q_1)$ to $(\bot\gamma, z)$, and from $(\bot\gamma, z')$ to $(\bot\gamma, q_2)$. We also store the *push* and *pop* transitions and the corresponding vector of the weight function w, and repeat the process, recursively, for the maximum weight non-decreasing path from (δ, q') to (δ, q'') with $\mathsf{ASH}(d-1)$. We end up with a description of length $O(d)$ of the form

$$\begin{aligned}\rho^* = {} & (\bot\gamma_1, q^1_1) \overset{\rho_1}{\rightsquigarrow} (\bot\gamma_1, q^1_2) \overset{push_1}{\to} (\bot\gamma_1\gamma_2, q^2_1) \overset{\rho_2}{\rightsquigarrow} (\bot\gamma_1\gamma_2, q^2_2) \overset{push_2}{\to} (\bot\gamma_1\gamma_2\gamma_3, q^3_1) \overset{\rho_3}{\rightsquigarrow} (\bot\gamma_1\gamma_2\gamma_3, q^3_2) \overset{push_3}{\to} \\ & \cdots \overset{push_d}{\to} (\bot\gamma_1 \ldots \gamma_d, q^d_1) \overset{\rho_{d+1}}{\rightsquigarrow} (\bot\gamma_1 \ldots \gamma_d, q^d_2) \overset{pop_1}{\to} (\bot\gamma_1 \ldots \gamma_{d-1}, q^{d-1}_3) \overset{\rho_{d+2}}{\rightsquigarrow} (\bot\gamma_1 \ldots \gamma_{d-1}, q^{d-1}_4) \\ & \overset{pop_2}{\to} (\bot\gamma_1 \ldots \gamma_{d-2}, q^{d-2}_3) \overset{\rho_{d+3}}{\rightsquigarrow} (\bot\gamma_1 \ldots \gamma_{d-2}, q^{d-2}_4) \overset{pop_3}{\to} \cdots \overset{pop_d}{\to} (\bot\gamma_1, q^1_3) \overset{\rho_{2\cdot d+1}}{\rightsquigarrow} (\bot\gamma_1, q^1_4);\end{aligned}$$

where $q^1_1 = q_1$, $q^1_4 = q_2$ and $\gamma_1 = \gamma$. Intuitively, the path ρ^* is decomposed as the path $\rho_1\ push_1\ \rho_2\ push_2\ \cdots\ push_d\ \rho_{d+1}\ pop_1\ \rho_{d+2}\ \cdots\ pop_d\ \rho_{2\cdot d+1}$, where the ρ_1 realizes the value $s_d(q^1_1, \gamma_1, q^1_2)$, ρ_2 realizes the value $s_{d-1}(q^2_1, \gamma_2, q^2_2)$ and so on; and similarly ρ_{d+1} realizes the value $s_0(q^d_1, \gamma_d, q^d_2)$, ρ_{d+2} realizes the value $s_1(q^{d-1}_3, \gamma_{d-1}, q^{d-1}_4)$, ρ_{d+3} realizes the value $s_2(q^{d-2}_3, \gamma_{d-2}, q^{d-2}_4)$ and so on; and finally, $\rho_{2\cdot d+1}$ realizes $s_d(q^1_3, \gamma_1, q^1_4)$.

Since $d > (|Q| \cdot |\Gamma|)^2$, there must exist $1 \leq i < j \leq d$, and $h_1, h_2, h_3, h_4 \in \{1, \ldots, 4\}$ such that $q^i_{h_1} = q^j_{h_2}$, $q^i_{h_3} = q^j_{h_4}$, $\gamma_i = \gamma_j$, and the weight of the path from $(\bot\gamma_1 \ldots \gamma_i, q^i_{h_1})$ to $(\bot\gamma_1 \ldots \gamma_j, q^j_{h_2})$ plus the weight of the path from $(\bot\gamma_1 \ldots \gamma_j, q^i_{h_3})$ to $(\bot\gamma_1 \ldots \gamma_i, q^j_{h_4})$ is positive. We sequentially iterate over all such tuples of i, j, h_1, h_2, h_3 and h_4 in polynomial time, and a witness path P can be obtained as of the form of ρ^*. The computation of $w(P)$ and $\frac{w(P)}{|P|}$ is obtained from the vector of the summary function, and the *push* and *pop* transitions along with the vector of weights according to w of such transitions, i.e.,

$$\begin{aligned}(w(P), |P|) &= \Big(\sum_{i=1}^{d} w(push_i) + w(pop_i), 2 \cdot d\Big) + \sum_{i=1}^{2\cdot d+1} (w(\rho_i), |\rho_i|) \\ &= \Big(\sum_{i=1}^{d} w(push_i) + w(pop_i), 2 \cdot d\Big) + \\ &\quad \sum_{i=1}^{d+1} \mathsf{VECT}(s_{d+1-i}(q^i_1, \gamma_i, q^i_2)) + \sum_{i=d+2}^{2d+1} \mathsf{VECT}(s_{i-d-1}(q^i_3, \gamma_i, q^i_4)).\end{aligned}$$

Hence it follows that we can compute $w(P)$ and $\frac{w(P)}{|P|}$ in polynomial time and the proof follows. □

Our goal now is the computation of the ω values of the summary function. To achieve the computation of ω values we will define another summary function s^* and a new WPS $\mathcal{A}^*$ such that certain cycles in $\mathcal{A}^*$ will characterize the ω values of the summary function. We now define the summary function s^* and the pushdown system $\mathcal{A}^*$. Let $d = (|Q| \cdot |\Gamma|)^2$. The new summary function s^* is defined as follows: if the values of s_d and s_{d+1} are the same then it is assigned the value of s_d, and otherwise the value ω. Formally,

$$s^*(q_1, \gamma, q_2) = \begin{cases} s_d(q_1, \gamma, q_2) & \text{if } s_d(q_1, \gamma, q_2) = s_{d+1}(q_1, \gamma, q_2) \\ \omega & \text{if } s_d(q_1, \gamma, q_2) < s_{d+1}(q_1, \gamma, q_2). \end{cases}$$

The new WPS $\mathcal{A}^*$ is constructed from $\mathcal{A}$ by adding the following set of ω-edges: $\{(q_1, \gamma, q_2, \mathit{skip}) \mid s^*(q_1, \gamma, q_2) = \omega\}$.

Lemma 13 *([17]). For all $q_1, q_2 \in Q$ and $\gamma \in \Gamma$, the following assertion holds: the original summary function $s(q_1, \gamma, q_2) = \omega$ iff there exists a non-decreasing path in $\mathcal{A}^*$ from $(\bot\gamma, q_1)$ to $(\bot\gamma, q_2)$ that goes through an ω-edge.*

We will now present the required polynomial-time algorithm for Proposition 2, and we present the algorithm for the case with $r = 0$ (and this is without loss of generality). The algorithm is similar to solution of WPS with one-dimensional objective of [17]. The final ingredient is the notion of summary graph.

Summary graph and positive simple cycles. Given a WPS $\mathcal{A} = \langle Q, \Gamma, q_0 \in Q, E \subseteq (Q \times \Gamma) \times (Q \times \mathsf{Com}(\Gamma)), w \cdot \vec{\lambda} : E \to \mathbb{Z} \rangle$ and the summary function s, we construct the *summary graph* $\mathsf{Gr}(\mathcal{A}) = (\overline{V}, \overline{E})$ of $\mathcal{A}$ with a weight function $\overline{w} : \overline{E} \to \mathbb{Z} \cup \{\omega\}$ as follows: (i) $\overline{V} = Q \times \Gamma$; and (ii) $\overline{E} = E_{\mathit{skip}} \cup E_{\mathit{push}}$ where $E_{\mathit{skip}} = \{((q_1, \gamma), (q_2, \gamma)) \mid s(q_1, \gamma, q_2) > -\infty\}$, and $E_{\mathit{push}} = \{((q_1, \gamma_1), (q_2, \gamma_2)) \mid (q_1, \gamma_1, q_2, \mathit{push}(\gamma_2)) \in E\}$; and (iii) for all $e = ((q_1, \gamma), (q_2, \gamma)) \in E_{\mathit{skip}}$ we have $\overline{w}(e) = s(q_1, \gamma, q_2)$, and for all $e \in E_{\mathit{push}}$ we have $\overline{w}(e) = (w \cdot \vec{\lambda})(e)$ (i.e., according to weight function of $\mathcal{A}$). A simple cycle C in $\mathsf{Gr}(\mathcal{A})$ is a *positive simple cycle* iff one of the following conditions hold: (i) either C contains an ω-edge (i.e., edge labeled ω by $\overline{w}$); or (ii) the sum of the weights of the edges of the cycles according to $\overline{w}$ is positive. The summary functions and the summary graph can be constructed in polynomial time. The first step of the algorithm is to build the summary graph and to check if there is a path from (γ, q) to (γ, q) with a positive weight. We consider the following cases of existence of such a positive weight path.

1. If there is no such path, then there does not exist pumpable pair of paths $P = (p_1, p_2)$ with positive weight (i.e., there exists no pumpable pair P with $(w \cdot \vec{\lambda})(P) > 0$).
2. We now consider the case when such a positive weight path exists. If such a path exists, we consider the path with maximum weight that is shortest (i.e., among the ones with maximum weight we choose a path that is shortest). We have two distinct cases.
   (a) We first consider the case when the path do not go through an ω edge. Then the path does not have a pumpable pair for the following reason: if the pumpable pair is positive, then the weight is not the maximum, and if the pumpable pair is non-negative, removing it ensures we obtain a maximum weight path with shorter length. Hence the length of the path is at most ℓ. Since we have stored the vector of the summary function (which stores the weights according to w and length of the witness paths) we compute the weight of this path according to w (and not according to $w \cdot \vec{\lambda}$), and return the average weight of this path.
   (b) Otherwise, the path goes through an ω edge in the summary graph. If there is an ω edge due to a proper cycle with positive weight, then we can detect this cycle in the construction of the summary graph and compute its average weight according to w (since we have the vector of the summary function that stores the weight according to w and the length of the witness paths). Otherwise, by Lemma 13, it follows that there is a non-decreasing path from (γ, q) to (γ, q) that has a non-decreasing sub-path from (δ, q_1) to (δ, q_2) and $s_{d+1}(q_1, \delta, q_2) > s_d(q_1, \delta, q_2)$. We have already described a polynomial time algorithm for finding such q_1, q_2 and δ. Once we find q_1, q_2 and δ, by Lemma 12, we can compute $w(P)$ and $\frac{w(P)}{|P|}$ in polynomial time.

The proof of Proposition 2 follows.

Example 1. In this example we demonstrate the notion of summary graphs. Consider the multidimensional WPS $\mathcal{A}$ depict in Fig. 9 and a vector $\vec{\lambda} = (1, 3)$. The single-dimensional WPS $\mathcal{A} \cdot \vec{\lambda}$ that is constructed by multiplying the weight function of $\mathcal{A}$ by $\vec{\lambda}$ is depict in Fig. 10. The summary graph of $\mathcal{A} \cdot \vec{\lambda}$ is depict in Fig. 11. The summary graph contains a positive simple cycle, namely, $(q_1, \gamma) \to (q_1, \gamma)$, hence in $\mathcal{A} \cdot \vec{\lambda}$ there exists a path with positive mean-payoff. Indeed, the path with the suffix $\mathit{push}(\gamma)^\omega$ has mean-payoff 1.

4. Recursive games under modular strategies with mean-payoff objectives

In this section we will consider recursive games (which are equivalent to pushdown games) with modular strategies. Note that there is no intuitive interpretation of modular strategies for pushdown games and it is standard (as considered in all works in literature) to define and consider modular strategies in the context of recursive games. We start with

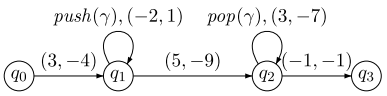

Fig. 9. A multidimensional WPS $\mathcal{A}$. If an edge is not labeled with a command, then the command is *skip*. The label $pop(\gamma)$ stands for: if the top symbol is γ, then a pop transition is possible.

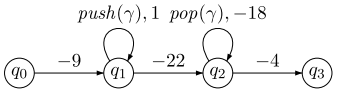

Fig. 10. A single-dimensional WPS $\mathcal{A} \cdot (1, 3)$.

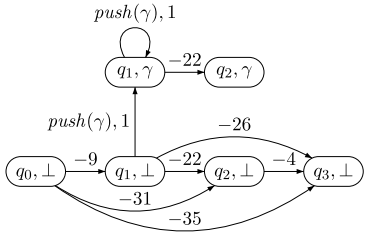

Fig. 11. Summary graph of $\mathcal{A} \cdot (1, 3)$. The reader should note that this is a finite graph. We explicitly label some of the transitions by push only to simplify the illustration.

the definitions and present four results for mean-payoff objectives in such games: (1) we show undecidability for the multidimensional problem, and hence focus on the one-dimensional case; (2) for the one-dimensional case we show a NP-hardness result; and finally (3) we present an algorithm that runs in polynomial time when relevant parameters are fixed.

Weighted recursive game graphs (WRGs). A *recursive game graph* $\mathcal{A}$ consists of a tuple $\langle A_0, A_1, \ldots, A_n \rangle$ of *game modules*, where each game module $A_i = (N_i, B_i, V_i^1, V_i^2, En_i, Ex_i, \delta_i)$ consists of the following components:

- A finite nonempty set of *nodes* N_i.
- A nonempty set of *entry* nodes $En_i \subseteq N_i$ and a nonempty set of *exit* nodes $Ex_i \subseteq N_i$.
- A set of *boxes* B_i.
- Two disjoint sets V_i^1 and V_i^2 that partition the set of nodes and boxes into two sets, i.e., $V_i^1 \cup V_i^2 = N_i \cup B_i$ and $V_i^1 \cap V_i^2 = \emptyset$. The set V_i^1 (resp. V_i^2) denotes the places where it is the turn of player 1 (resp. player 2) to play (i.e., choose transitions). We denote the union of V_i^1 and V_i^2 by V_i.
- A labeling $Y_i : B_i \to \{1, \ldots, n\}$ that assigns to every box an index of the game modules $A_1 \ldots A_n$.
- Let $\mathsf{Calls}_i = \{(b, e) \mid b \in B_i, e \in En_j, j = Y_i(b)\}$ denote the set of *calls* of module A_i and let $\mathsf{Retns}_i = \{(b, x) \mid b \in B_i, x \in Ex_j, j = Y_i(b)\}$ denote the set of *returns* in A_i. Then, $\delta_i \subseteq (N_i \cup \mathsf{Retns}_i) \times (N_i \cup \mathsf{Calls}_i)$ is the *transition relation* for module A_i.

A *weighted recursive game graph* (for short WRG) is a recursive game graph, equipped with a weight function w on the transitions. We also refer the readers to [2] for detailed descriptions and illustrations with figures of recursive game graphs. WLOG we shall assume that the boxes and nodes of all modules are disjoint. Let $B = \bigcup_i B_i$ denote the set of all boxes, $N = \bigcup_i N_i$ denote the set of all nodes, $En = \bigcup_i En_i$ denote the set of all entry nodes, $Ex = \bigcup_i Ex_i$ denote the set of all exit nodes, $V^1 = \bigcup_i V_i^1$ (resp. $V^2 = \bigcup_i V_i^2$) denote the set of all places under player 1's control (resp. player 2's control), and $V = V^1 \cup V^2$ denote the set of all vertices. We will also consider the special case of one-player WRGs, where either V^2 is empty (player-1 WRGs) or V^1 is empty (player-2 WRGs). WLOG we will assume that the every module has a unique entrance (a polynomial reduction from module with many entrances to one with a single entrance was given in [2]). The module A_0 is the initial module, and its entry node the starting node of the game.

Configurations, paths and local history. A *configuration* c consists of a sequence $(b_1, \ldots, b_r, u)$, where $b_1, \ldots, b_r \in B$ and $u \in N$. Intuitively, $b_1, \ldots, b_r$ denote the current stack (of modules), and u is the current node. A sequence of configurations is *valid* if it does not violate the transition relation. The *configuration stack height* of c is r. Let us denote by $\mathbb{C}$ the set of all configurations, and let $\mathbb{C}_1$ (resp. $\mathbb{C}_2$) denote the set of all configurations under player 1's control (resp. player 2's control). A *path* $\pi = \langle c_1, c_2, c_3, \ldots \rangle$ is a valid sequence of configurations. Let $\rho = \langle c_1, c_2, \ldots, c_k \rangle$ be a valid finite sequence of configurations, such that $c_i = (b_1^i, \ldots, b_{d_i}^i, u_i)$, and the stack height of c_i is d_i. Let c_i be the first configuration with stack height $d_i = d_k$, such that for every $i \leq j \leq k$, if c_j has stack height d_i, then $u_j \notin Ex$ (u_j is not an exit node). The *local history* of ρ, denoted by $\mathsf{LocalHistory}(\rho)$, is the sequence $(u_{j_1}, \ldots, u_{j_m})$ such that $c_{j_1} = c_i$, $c_{j_m} = c_k$, $j_1 < j_2 < \cdots < j_m$, and the

stack height of $c_{j_1}, \ldots, c_{j_m}$ is exactly d_i. Intuitively, the local history is the sequence of nodes in a module. Note that by definition, for every $\rho \in \mathbb{C}^*$, there exists $i \in \{1, \ldots, n\}$ such that all the nodes that occur in $\mathsf{LocalHistory}(\rho)$ belong to V_i. We say that $\mathsf{LocalHistory}(\rho) \in A_i$ if all the nodes in $\mathsf{LocalHistory}(\rho)$ belong to V_i.

Plays, strategies and modular strategies. A play is played in the usual sense over the global game graph (which is possibly an infinite graph). A (finite) play is a (finite) valid sequence of configurations $\langle c_1, c_2, c_3, \ldots \rangle$ (i.e., a path in the global game graph). A *strategy* for player 1 is a function $\tau : \mathbb{C}^* \times \mathbb{C}_1 \to \mathbb{C}$ respecting the edge relationship of the global game graph, i.e., for all $w \in \mathbb{C}^*$ and $c_1 \in \mathbb{C}_1$ we have that $(c_1, \tau(w \cdot c_1))$ is an edge in the global game graph. A *modular strategy* τ for player 1 is a set of functions $\{\tau_i\}_{i=1}^n$, one for each module, where for every i, we have $\tau_i : (N_i \cup \mathsf{Retns}_i)^* \to \delta_i$. The function τ is defined as follows: For every play prefix ρ we have $\tau(\rho) = \tau_i(\mathsf{LocalHistory}(\rho))$, where $\mathsf{LocalHistory}(\rho) \in A_i$. The function τ_i is the *local strategy* of module A_i. Intuitively, a modular strategy only depends on the local history, and not on the context of invocation of the module. A modular strategy $\tau = \{\tau_i\}_{i=1}^n$ is a *finite-memory* modular strategy if τ_i is a finite-memory strategy for every $i \in \{1, \ldots, n\}$. A *memoryless* modular strategy is defined in similar way, where every component local strategy is memoryless.

Mean-payoff objectives and winning modular strategies. The *modular winning strategy problem* asks if player 1 has a modular strategy τ such that against every strategy σ for player 2 the play π given the starting node and the strategies satisfy $\mathsf{LimAvg}(\pi) \geq \vec{0}$ (note that the counter strategy of player 2 is a general strategy).

4.1. Undecidability for multidimensional mean-payoff objectives

In this section we will show that the problem of deciding the existence of modular winning strategy for player 1 in WRGs with multidimensional mean-payoff objectives is undecidable. The reduction would be from reachability games over tuples of integers. We start by introducing these games.

Reachability games over $\mathbb{Z}^k$. A *reachability game over* $\mathbb{Z}^k$ consists of a finite-state game graph G, a k dimensional weight function $w : E \to \mathbb{Z}^k$, and an initial weight vector $\vec{\nu} \in \mathbb{Z}^k$. An infinite play π is winning for player 1 if there exists some finite prefix $\pi' \sqsubseteq \pi$ such that $w(\pi') + \vec{\nu} = 0$ and the last vertex in π' is a player-1 vertex.

Lemma 14. *The following problem is undecidable: Given a reachability game over $\mathbb{Z}^2$ and a starting vertex v, decide if there is a winning strategy τ for player 1 to ensure that for all strategies σ for player 2 the play $\pi(\tau, \sigma, v)$ is winning for player 1.*

Proof. Follows immediately from [34]. □

We will present a general reduction from reachability games over $\mathbb{Z}^k$ to WRGs under modular strategies with multidimensional mean-payoff objectives of $2 \cdot k + 2$ dimensions, with three modules (two of them with single exit, and an initial module without any exits). Given a reachability game over $\mathbb{Z}^k$ with game graph G, weight function w and initial vector $\vec{\nu}$, we construct a WRG graph $\mathcal{A} = \langle A_0, A_1, A_2 \rangle$ with a weight function of $2 \cdot k + 2$ dimensions in the following way.

- Module A_0: This module repeatedly invokes A_1 and A_2 (one call to A_1 and one call to A_2); and all the weights of the transitions are 0.
- Module A_1: This module has three nodes: entrance, exit and an additional one with a self-loop edge with weight 0 in the first $2 \cdot k$ dimensions, weight $+1$ in dimension $2 \cdot k + 1$ and weight -1 in dimension $2 \cdot k + 2$; the weight of the edges from the entrance node to the additional node and from the additional node to the exit node are 0 in every dimension. All the nodes are in the control of player 1.
- Module A_2: The nodes of this module are the entrance and exit nodes, the nodes V of the reachability game G, and an additional node v^*. The entrance node leads to the initial vertex of G with edge weight $(\vec{\nu}, -\vec{\nu}, 0, 0)$ (i.e., the first k dimensions are according to $\vec{\nu}$, dimensions $k+1$ to $2 \cdot k$ are according to $-\vec{\nu}$, and the last two dimensions are 0). For every edge $e = (u, v)$ in G, there is such transition in A_2 with weight $(w(e), -w(e), -1, +1)$. In addition, from every player-1 vertex in V there is a transition to v^* with weight 0 in every dimension. In v^* there is a self-loop transition with weight -1 in dimension $2k+1$, $+1$ in dimension $2k+2$ and 0 in the rest of the dimensions; and in addition there is a transition to the exit node with weight 0 in every dimension.

The pictorial descriptions of the modules A_0, A_1, and A_2 are shown in Fig. 12, Fig. 13, and Fig. 14, respectively.

Observation 1. The following observations hold:

1. If player-1 strategy τ_1 for module A_1 is to never exits, then it is not a winning strategy (since the mean-payoff in dimension $2 \cdot k + 2$ will be -1).
2. If for a player-1 strategy τ_2 for module A_2, there is a play π consistent with τ_2 that does not reach v^*, then τ_2 is not a winning strategy (since the mean-payoff of ρ in dimension $2 \cdot k + 1$ will be -1).

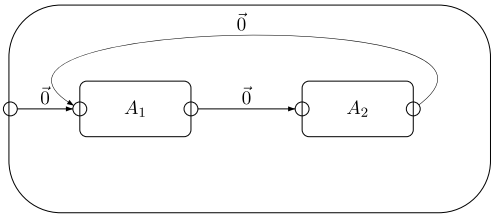

Fig. 12. Module A_0.

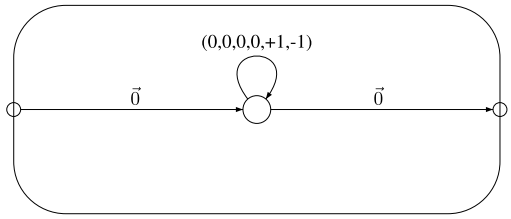

Fig. 13. Module A_1.

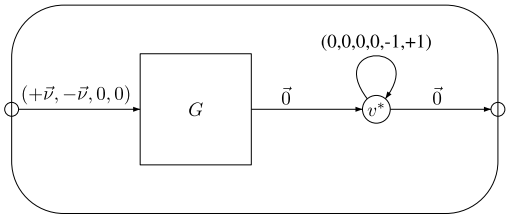

Fig. 14. Module A_2.

Lemma 15. *If player 1 does not have a winning strategy in the reachability game over $\mathbb{Z}^k$, then there is no modular winning strategy for player 1 in $\mathcal{A}$.*

Proof. If player 1 does not have a winning strategy in the reachability game over $\mathbb{Z}^k$, then it follows from determinacy of reachability games (Martin's Theorem [33]) that player-2 has a winning strategy. Let σ be a player-2 winning strategy for the reachability game. We fix player-2 strategy for the modular game to be σ according to the local history of A_2 and claim that it is a winning strategy for player 2 in the WRG against the multidimensional mean-payoff objective for player 1. Indeed, let $\tau = \{\tau_1, \tau_2\}$ be a player-1 modular strategy, and we consider the path π which is formed by playing according to τ and σ. By Observation 1 if π never exits A_1 or never reaches node v^*, then player 2 wins. Otherwise, since σ is a winning strategy in the reachability game, we get that in the first sub-path of π that leads from the entrance of A_2 to v^*, one of the dimensions $1 \leq i \leq 2 \cdot k$ has a negative weight. We note that both σ and τ are modular strategies, and thus the path π is periodic and the mean-payoff of π in dimension i is negative. To conclude, if player 2 is the winner in the reachability game, then player 1 does not have a modular winning strategy in $\mathcal{A}$. □

Lemma 16. *If player 1 has a winning strategy in the reachability game, then there is a modular winning strategy for player 1 in $\mathcal{A}$.*

Proof. Let τ_G be a player-1 winning strategy for the reachability game. By König's Lemma there exists a fixed constant $n \in \mathbb{N}$ such that player 1 can assure the reachability objective, against every player-2 strategy, with at most n rounds. We now derive a modular winning strategy in $\mathcal{A}$ from τ_G:

- Module A_1: Follow the self-loop edge for n rounds and exit.
- Module A_2: Follow strategy τ_G, until the weight in every dimension, according to the reachability game over G, is 0 and a player-1 vertex was reached, and then go to v^*. Let m be the number of rounds played according to τ_G in the current local history of A_2, then player 1 follows the self-loop in v^* for $n - m$ times and goes to the exit node.

It is easy to observe that any play according to the strategy above has a mean-payoff value of 0 in every dimension. □

From Lemma 14, Lemma 15, and Lemma 16 we obtain the following result:

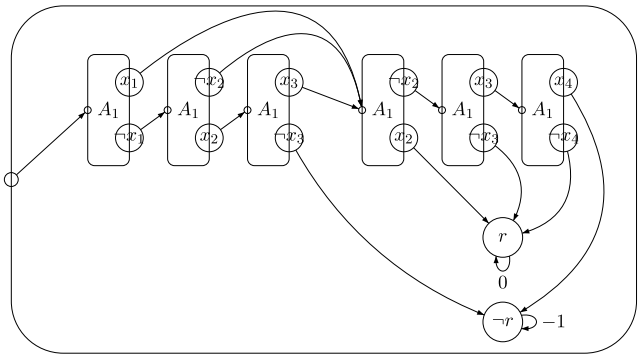

Fig. 15. Module A_0 for $(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee \neg x_4)$.

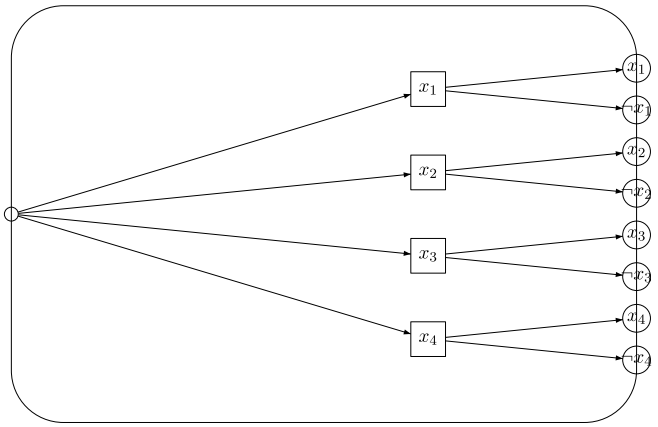

Fig. 16. Module A_1.

Theorem 3. *The problem of deciding the existence of a modular winning strategy in WRGs with multidimensional mean-payoff objectives is undecidable, even for hierarchical games (i.e., games without recursive calls), with six dimensions, three modules and with at most single exit for each module.*

In view of Theorem 3 we will focus on complexity and algorithms for WRGs under modular strategies for one-dimensional mean-payoff objectives.

4.2. NP-hardness

We consider WRGs under modular strategies with one-dimensional mean-payoff objectives. It was already shown in [17] that if the number of modules is not bounded, then even if all modules have at most one exit, the problem is NP-hard even when there is only player 1 and weights are restricted to $\{-1, 0, 1\}$. We present a similar hardness result when the number of modules is restricted to only two, but the number of exits is not bounded (we recall that the general problem is in NP [17], hence the lower bound is tight). We present a simple log-space reduction from 3SAT to WRGs with two modules. The objective we will consider is the reachability objective, where the mean-payoff objective is satisfied once a vertex r is reached (i.e., r has a self-loop with weight 0 and all other transitions have negative weights).

The reduction For a 3SAT formula $\varphi(x_1, \ldots, x_n) = \bigwedge_{i=1}^{m} C_i$ we construct a WRG with two modules, namely A_0 and A_1.

- Module A_1: The module has $2n$ exits namely, $Ex_{x_1}, Ex_{\neg x_1}, \ldots, Ex_{x_n}, Ex_{\neg x_n}$, an entrance node that is owned by player 2, and n player-1 nodes $x_1, \ldots, x_n$. From the entrance node there is a transition (En, x_i), for $i = 1, \ldots, n$; and from every node x_i there is one transition to Ex_{x_i} and one transition to $Ex_{\neg x_i}$. Intuitively, a modular strategy for player 1 is to decide on a True/False value for every x_i (since the strategy is modular, the same decision is always taken).
- Module A_0: This is the initial module; it consists of m gadgets $C_1, \ldots, C_m$ (note that these are gadgets and not modules), and two sink states, namely r and $\neg r$, where r is the reachability objective. A gadget $C_i = y_i^1 \vee y_i^2 \vee y_i^3$ consists of three sub-gadgets, namely, y_i^1, y_i^2, y_i^3; gadget y_i^j invokes module A_1 and the exits $(\{Ex_{x_1}, Ex_{\neg x_1}, \ldots, Ex_{x_n}, Ex_{\neg x_n}\} \setminus \{y_i^j, \neg y_i^j\})$ of A_1 leads to the good sink node r, the exit y_i^j leads to gadget C_{i+1} (or to node r if $i = m$), and the exit $\neg y_i^j$ leads to sub-gadget y_i^{j+1} (or to the bad sink node $\neg r$ if $j = 3$).

The reduction is illustrated in Fig. 15 and Fig. 16. It is an easy observation that player 1 has a modular winning strategy iff the formula φ is satisfiable.

Theorem 4. *The decision problem of existence of modular winning strategies in WRG's with one-dimensional mean-payoff objectives is NP-hard even for WRG's with two modules and weights restricted to* $\{0, -1\}$.

4.3. Algorithm for one-dimensional mean-payoff objectives

Given the undecidability result, we focus on WRGs with one-dimensional mean-payoff objectives, and given the hardness results for either unbounded number of modules or unbounded number of exits, our goal is to present an algorithm that runs in polynomial time if both the number of modules and the number of exits are bounded. For the rest of this section we denote the number of game modules by M, the number of exits and boxes (in the entire graph) by E and B, respectively, and by n and m the maximal size of $|V_i|$ and $|\delta_i|$ (number of vertices and transitions) respectively that a module has. If M, E and W (the maximal absolute weight) are bounded, then our algorithm runs in polynomial time. We first present a theorem from [17] that will be useful in our result and then present the notion of cycle-free memoryless modular strategy.

Theorem 5 *([17]). Given a WRG $\mathcal{A}$ with a one-dimensional weight function, if there is a modular winning strategy for the objective* LimAvg, *then there is a memoryless modular winning strategy.*

Negative-cycle-free memoryless modular strategy. A player-1 memoryless modular strategy τ is called *negative-cycle-free memoryless modular strategy* if in the recursive graph $\mathcal{A}^\tau$ there are no proper cycles C with negative weights, i.e., $w(C) < 0$.

Signature of a negative-cycle-free memoryless modular strategy. The *signature* of a negative-cycle-free memoryless modular strategy $\tau = \{\tau_i\}_{i=1}^{\mathsf{M}}$ is an M-tuple of function $\mathsf{Sig}(\tau) = \{\mathsf{Sig}_i : Ex_i \to \mathbb{Z} \cup \{-\omega, +\infty\}\}_{i=1}^{\mathsf{M}}$ such that for an exit node x in module A_i we have $\mathsf{Sig}_i(x) = z$ if

- $z \in \mathbb{Z}$ and any non-decreasing path with minimal weight in $\mathcal{A}^\tau$ from En_i to x (in the same stack height) has weight z.
- $z = +\infty$ and there is no non-decreasing path in $\mathcal{A}^\tau$ from En_i to x.
- $z = -\omega$ and for every integer j there is a non-decreasing path in $\mathcal{A}^\tau$ from En_i to x (at the same stack height), with weight at most j.

The next lemma demonstrates an important property of signature functions.

Lemma 17. *Let $\ell = (\mathsf{M} \cdot n)^{\mathsf{M}\cdot\mathsf{E}+1}$; let τ be a negative-cycle-free memoryless modular strategy; and let W denote the maximal weight (in absolute value) that occur in $\mathcal{A}$. Then* $\mathsf{Sig}(\tau)$ *has the following property:*

For every $i \in \{1, \ldots, \mathsf{M}\}$, the image (range) of Sig_i *is* $\{-\omega, +\infty\} \cup (\mathbb{Z} \cap [-W \cdot \ell, W \cdot \ell])$

Proof. We fix the strategy τ in $\mathcal{A}$, and obtain the player-2 recursive game graph $\mathcal{A}^\tau$. To show the result we need to prove that if there is a path (in $\mathcal{A}^\tau$) from En_i (the entrance of A_i) to $x \in Ex_i$ with weight less than $-W \cdot \ell$, then for every $r \in \mathbb{Z}$ there exists a path from En_i to x, consistent with τ, and with weight less than r; and that it is impossible that the path with the minimal weight from En_i to x has weight at least $W \cdot \ell + 1$.

The proof is as follows: let π be the shortest path in $\mathcal{A}^\tau$ from En_i to x with weight $w(\pi) < -W \cdot \ell$ (note that π corresponds to a play consistent with τ). Since $w(\pi) < -W \cdot \ell$, it must be that $|\pi| > \ell$, therefore π must have a pumpable pair of paths, and since π is the shortest path from En_i to x with such weight, the weight of the pumpable pair must be strictly negative, and thus we can construct paths from En_i to x with arbitrary small weights.

Similarly, we show that if there is a path from En_i to x, then there is a path with weight at most $W \cdot \ell - 1$. Towards contradiction, let π be the path with minimal weight between En_i and x and $w(\pi) \geq W \cdot \ell$ and π is the shortest path with minimal weight. As $|\pi| \geq \ell$ it follows that it has a pumpable pair of paths P. If $w(P) > 0$ or $w(P) < 0$, then we get a contradiction to the fact that π has minimal weight (either by omitting P if $w(P) < 0$ or pumping P arbitrarily if $w(P) > 0$). If $W(P) = 0$, then we get a contradiction to the assumption that π is the shortest path by simply omitting P. The desired result follows. □

Feasibility of signature We say that a function $\mathsf{Sig} : Ex \to \{-\omega, +\infty\} \cup \mathbb{Z}$ is *feasible* if there is a negative-cycle-free memoryless modular strategy τ such that $\mathsf{Sig}(\tau) = \mathsf{Sig}$.

Lemma 18. *Given a threshold vector $\vec{\nu} \in (\{-\omega, +\infty\} \cup \mathbb{Z})^{\mathsf{E}}$, we can verify in $(\mathsf{M} \cdot n)^{O(\mathsf{M}\cdot\mathsf{E}^2)} \cdot W^{O(\mathsf{E})}$ time if there exists a feasible signature function $\mathsf{Sig} : Ex \to \{-\omega, +\infty\} \cup \mathbb{Z}$ such that $\mathsf{Sig} \geq \vec{\nu}$ (i.e., for every $x \in Ex$ we have $\mathsf{Sig}(x) \geq \nu_x$).*

Proof. By Lemma 17 we may assume that the input is restricted for $\vec{\nu} \in (\{-\omega, +\infty\} \cup (\mathbb{Z} \cap [-W\ell, +W\ell]))^{\mathsf{E}}$ and $\mathsf{Sig} : Ex \to \{-\omega, +\infty\} \cup (\mathbb{Z} \cap [-W\ell, +W\ell])$. The proof of the lemma will use the idea of signature verification games.

The signature verification games. For a recursive game $\mathcal{A}$ and a function $\mathsf{Sig} : Ex \to \{-\omega, +\infty\} \cup \mathbb{Z}$ we construct M game modules $G_1, \ldots, G_{\mathsf{M}}$, such that G_i is formed from the module A_i by replacing every box b, that invokes module A_j and its k-th return node leads to node v_k, with a player-2 node v_b and edges (v_b, v_k) with weight $\mathsf{Sig}_j(Ex_k)$. Intuitively every game module is like a finite-state game with thresholds for exit vertices. We first prove a claim related to signature verification games.

Claim. For every game module G_i there exists a strategy τ_i that satisfies Sig, i.e., it assures:

- every path in $G_i^{\tau_i}$ from En_i to Ex_j has weight at least $\mathsf{Sig}_i(Ex_j)$; and
- there are no cycles with negative weight in $G_i^{\tau_i}$;

if and only if there exists a feasible signature function Sig' such that $\mathsf{Sig}' \geq \mathsf{Sig}$.

Proof of claim. We prove both directions of the claim. We start with the left to the right direction. By Theorem 5 such strategies $\{\tau_i\}_{i=1}^{\mathsf{M}}$ exist iff there exist memoryless strategies $\{\tau_i^*\}_{i=1}^{\mathsf{M}}$ that satisfy the above. Clearly, τ^* is also a modular strategy. In addition, for every path π in $\mathcal{A}^{\tau^*}$, the path does not contain negative proper cycles (and hence, τ^* is a negative-cycle-free strategy), and the path does not violate the constraints according to Sig. The proof is by a straightforward induction on the additional stack height of π. Hence we have $\mathsf{Sig}(\tau^*) \geq \mathsf{Sig}$. The other direction is simpler. Clearly if there exists a negative-cycle-free modular strategy τ such that $\mathsf{Sig}(\tau) \geq \mathsf{Sig}$, then τ_i satisfies both items for every game module G_i. This proves the desired claim.

The results of [18, Lemma 31] provide an algorithm that decides if for a given function $f : Ex \to \{-\omega, +\infty\} \cup (\mathbb{Z} \cap [-W\ell, +W\ell])$ and a game module G_i there is a memoryless strategy that satisfies f; this is done by solving a (finite-state) mean-payoff game with one-dimensional objective with weights at most $2 \cdot n \cdot W \cdot \ell$. Hence, we can sequentially go over all the functions $f : Ex \to \{-\omega, +\infty\} \cup (\mathbb{Z} \cap [-W\ell, +W\ell])$ such that $f \geq \vec{\nu}$ and check if f is satisfiable. By the claim a signature $\mathsf{Sig} \geq \vec{\nu}$ exists if and only if such f was found.

Complexity. The complexity analysis is as follows: there are $(2 \cdot W \cdot \ell + 2)^{\mathsf{E}}$ functions to verify; in the verification process we solve M mean-payoff games with weights at most $2 \cdot n \cdot W \cdot \ell$ and at most n vertices and m edges; and every mean-payoff game can be solved in $O(m \cdot n^2 \cdot W \cdot \ell)$ time [10]. Thus the overall complexity is

$$O(n^2 \cdot m \cdot (W \cdot \ell)^{\mathsf{E}+1} \cdot \mathsf{M}) = O(n^{\mathsf{M}\cdot\mathsf{E}^2+\mathsf{M}\cdot\mathsf{E}+\mathsf{E}+3} \cdot m \cdot \mathsf{M}^{\mathsf{M}\cdot\mathsf{E}^2+\mathsf{M}\cdot\mathsf{E}+\mathsf{E}+2} \cdot W^{\mathsf{E}+1}) = (\mathsf{M} \cdot n)^{O(\mathsf{M}\cdot\mathsf{E}^2)} \cdot W^{O(\mathsf{E})}$$

The desired result follows. □

Reduction from modular games to signature problem. Intuitively, for a given WRG $\mathcal{A}$, we would like to construct a new WRG $\mathcal{A}'$, such that player 1 is the winner in $\mathcal{A}$ iff there exists a feasible signature in $\mathcal{A}'$ with certain properties. We construct $\mathcal{A}'$ in the following way: Let $(A_1, \ldots, A_{\mathsf{M}})$ be the modules of $\mathcal{A}$, then we construct the modules $(A'_1, \ldots, A'_{\mathsf{M}})$ from $(A_1, \ldots, A_{\mathsf{M}})$ as follows:

- Add M exit nodes $x_1, \ldots, x_{\mathsf{M}}$ for every module.
- For every box node b, in module A_j, if b invokes module A_i, then for all $i \in \{1, \ldots, \mathsf{M}\}$, the auxiliary exit node x_i is connected (by an edge with weight 0) to the exit x_i in the module A_j.
- W.l.o.g. we assume that all the entrances are player-2 nodes, and we add edge $En_i \to x_i$ with zero weight for every i (i.e., we add an edge from the entry node of every module A'_i to the auxiliary exit node x_i).

We note that the number of exits E' in $\mathcal{A}'$ is $\mathsf{E} + \mathsf{M}^2$. The following lemma establishes a connection between winning in $\mathcal{A}$ and properties of signature function in $\mathcal{A}'$.

Lemma 19. *Player 1 has a memoryless modular winning strategy in $\mathcal{A}$ iff there is a feasible signature* Sig *in $\mathcal{A}'$ such that for every module A'_i we have* $\mathsf{Sig}_i(x_i) \geq 0$.

Proof. We first prove the direction from left to right. Let τ be a memoryless modular winning strategy (and therefore also negative-cycle-free) in $\mathcal{A}$. We note that τ is a modular negative-cycle free strategy also for $\mathcal{A}'$. We claim that (the feasible signature function) $\mathsf{Sig} = \mathsf{Sig}(\tau)$ satisfies $\mathsf{Sig}_i(x_i) \geq 0$. Indeed, if $\mathsf{Sig}_i(x_i) < 0$, then by the construction of $\mathcal{A}'$, there is a play ρ from En_i to En_i (at an higher stack height) with negative weight, that is consistent with τ. Since τ is a modular strategy we get that ρ^ω is a play with a negative mean-payoff that is consistent with τ, which contradicts the assumption that τ is a winning strategy.

To prove the converse direction, let τ be a memoryless negative-cycle-free strategy in $\mathcal{A}'$ such that $\mathsf{Sig}(\tau) = \mathsf{Sig}$. We note that τ is a modular strategy also for $\mathcal{A}$ and we claim that it is a winning strategy for $\mathcal{A}$. Indeed, let $\mathcal{A}^\tau$ be the player-2 game according to τ; if in $\mathcal{A}^\tau$ there is a path with negative mean-payoff then either

- there is a proper cycle in $\mathcal{A}^\tau$ with negative weight, which contradicts the assumption that τ is negative-cycle-free strategy; or

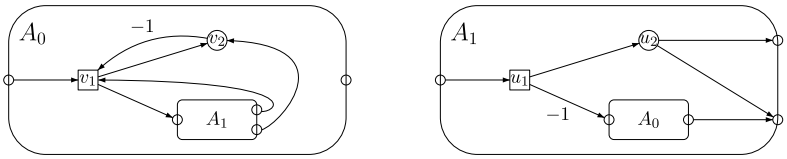

Fig. 17. WRG $\mathcal{A} = (A_0, A_1)$. Player 1 controls the square vertices and the rest of the vertices are controlled by player 2. The weight of all non-labeled edges is 0.

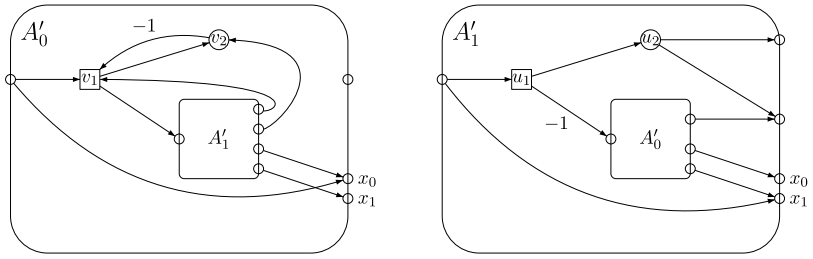

Fig. 18. WRG $\mathcal{A}' = (A'_0, A'_1)$. Player 1 controls the square vertices and the rest of the vertices are controlled by player 2. The weight of all non-labeled edges is 0.

- there is a non-decreasing cycle $\mathcal{A}^\tau$ with negative weight. If this is the case then for some module A_i there is a non-decreasing path in $\mathcal{A}^\tau$ from En_i to En_i with negative weight, and thus in $\mathcal{A}'^\tau$ there is a path with negative weight from En_i to x_i and therefore $\mathsf{Sig}_i(x_i) < 0$, in contradiction to the assumption.

The desired result follows. □

Example 2. Consider the WRG $\mathcal{A}$ depict in Fig. 17. The reduction from modular games to signature games would yield the WRG $\mathcal{A}'$ depict in Fig. 18. We claim that in $\mathcal{A}'$, any feasible strategy would have $\mathsf{Sig}_0(x_0) < 0$ (and thus player 1 cannot win the game). Indeed, in $\mathcal{A}'$ a memoryless player-1 strategy is deciding the next moves in nodes v_1 and u_1. A strategy that chooses to move from v_1 to u_1 would not yield a feasible signature as the negative cycle $v_1 \to u_1 \to v_1$ is consistent with the strategy. Hence, in v_1 player 1 must invoke the module A'_1. For node u_1 we consider two distinct cases:

- In the first case, player 1 invokes A'_0. In such case, the path $En_0 \to v_1 \to En_1 \to u_1 \to En_0 \to A_{0'_{x_0}} \to A_{1'_{x_0}} \to A_{0'_{x_0}}$ is consistent with the strategy and has negative weight. Hence, $\mathsf{Sig}_0(x_0) < 0$.
- In the second case, player 1 chose to move from u_1 to u_2. In this case, when player 2 proceeds to the second exit node from u_2 a negative cycle (that begins and ends in v_1) is formed, and the signature is not feasible.

Hence, for any feasible signature we have $\mathsf{Sig}_0(x_0) < 0$ and player 1 lose the game.

Theorem 6. *Given a WRG $\mathcal{A}$ with a one-dimensional mean-payoff objective, whether player 1 has a modular winning strategy can be decided in $(n \cdot \mathsf{M})^{O(\mathsf{M}^5 + \mathsf{M} \cdot \mathsf{E}^2)} \cdot W^{O(\mathsf{M}^2 + \mathsf{E})}$ time.*

Proof. We first construct the modular game graph $\mathcal{A}'$ and then we check if there is a signature function Sig such that $\mathsf{Sig}_i(x_i) \geq 0$ for every $i \in \{1, \ldots, \mathsf{M}\}$. The correctness and complexity follow from Lemma 19 and Lemma 18. □

5. Conclusion

In this work we considered the fundamental algorithmic questions related to multidimensional mean-payoff objectives in finite-state games, pushdown graphs, and pushdown games. We presented algorithms that precisely characterize the parameters that need to be constant for polynomial-time algorithms. Some interesting directions of future works are as follows. In [16] it has been shown that several interesting problems for interprocedural analysis can be modeled as pushdown systems with mean-payoff objectives. Extension of the above problems to multi-dimensional mean-payoff objectives is an interesting direction. The second interesting direction is to obtain algorithms with better computational complexity (i.e., better dependency in terms of the number of vertices, edges, weights, and dimensions) for the pseudo-polynomial time algorithms we present.

Acknowledgments

The research was supported by Austrian Science Fund (FWF) Grant No. P 23499-N23, FWF NFN Grant No. S11407-N23 (RiSE), ERC Start grant (279307: Graph Games), Microsoft faculty fellows award, the RICH Model Toolkit (ICT COST Action IC0901), and was carried out in partial fulfillment of the requirements for the Ph.D. degree of the second author.

References

[1] R. Alur, A. Degorre, O. Maler, G. Weiss, On omega-languages defined by mean-payoff conditions, in: FOSSACS, 2009, pp. 333–347.
[2] R. Alur, S.L. Torre, P. Madhusudan, Modular strategies for recursive game graphs, Theor. Comput. Sci. 354 (2) (2006) 230–249.
[3] C. Beeri, On the membership problem for functional and multivalued dependencies in relational databases, ACM Trans. Database Syst. 5 (1980) 241–259.
[4] R. Bloem, K. Chatterjee, T.A. Henzinger, B. Jobstmann, Better quality in synthesis through quantitative objectives, in: CAV, 2009, pp. 140–156.
[5] A. Bohy, V. Bruyére, E. Filiot, J.-F. Raskin, Synthesis from LTL specifications with mean-payoff objectives, in: TACAS, 2013.
[6] U. Boker, K. Chatterjee, T.A. Henzinger, O. Kupferman, Temporal specifications with accumulative values, ACM Trans. Comput. Log. 15 (4) (2014) 27:1–27:25.
[7] T. Brázdil, V. Brozek, V. Forejt, A. Kucera, Reachability in recursive Markov decision processes, Inf. Comput. 206 (5) (2008) 520–537.
[8] T. Brázdil, V. Brozek, A. Kucera, J. Obdrzálek, Qualitative reachability in stochastic BPA games, Inf. Comput. 209 (8) (2011) 1160–1183.
[9] T. Brázdil, K. Chatterjee, A. Kucera, P. Novotný, Efficient controller synthesis for consumption games with multiple resource types, in: CAV, 2012, pp. 23–38.
[10] L. Brim, J. Chaloupka, L. Doyen, R. Gentilini, J.-F. Raskin, Faster algorithms for mean-payoff games, Form. Methods Syst. Des. 38 (2) (2011) 97–118.
[11] J. Büchi, L. Landweber, Solving sequential conditions by finite-state strategies, Trans. Am. Math. Soc. 138 (1969) 295–311.
[12] C.S. Calude, S. Jain, B. Khoussainov, W. Li, F. Stephan, Deciding parity games in quasipolynomial time, in: STOC, 2017, in press.
[13] K. Chatterjee, L. Doyen, H. Edelsbrunner, T.A. Henzinger, P. Rannou, Mean-payoff automaton expressions, in: CONCUR, 2010, pp. 269–283.
[14] K. Chatterjee, L. Doyen, T.A. Henzinger, Quantitative languages, ACM Trans. Comput. Log. 11 (4) (2010).
[15] K. Chatterjee, L. Doyen, T.A. Henzinger, J.-F. Raskin, Generalized mean-payoff and energy games, in: FSTTCS, 2010, pp. 505–516.
[16] K. Chatterjee, A. Pavlogiannis, Y. Velner, Quantitative interprocedural analysis, in: POPL, 2015, pp. 539–551.
[17] K. Chatterjee, Y. Velner, Mean-payoff pushdown games, in: LICS, 2012, pp. 195–204.
[18] K. Chatterjee, Y. Velner, Mean-payoff pushdown games, CoRR, arXiv:1201.2829, 2012.
[19] K. Chatterjee, Y. Velner, Hyperplane separation technique for multidimensional mean-payoff games, in: CONCUR, in: LNCS, vol. 8052, Springer, 2013, pp. 500–515.
[20] E. Cohen, N. Megiddo, Strongly polynomial-time and NC algorithms for detecting cycles in periodic graphs, J. ACM 40 (4) (1993) 791–830.
[21] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, The MIT Press, 2001.
[22] M. Droste, I. Meinecke, Describing average- and longtime-behavior by weighted MSO logics, in: MFCS, 2010, pp. 537–548.
[23] A. Ehrenfeucht, J. Mycielski, Positional strategies for mean payoff games, Int. J. Game Theory 8 (2) (1979) 109–113.
[24] K. Etessami, M. Yannakakis, Recursive Markov decision processes and recursive stochastic games, in: ICALP'05, in: LNCS, vol. 3580, Springer, 2005, pp. 891–903.
[25] K. Etessami, M. Yannakakis, Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations, J. ACM 56 (1) (2009).
[26] P. Gordan, Ueber die auflosung linearer gleichungen mit reellen coeffizienten, Math. Ann. 6 (1873) 23–28.
[27] M. Grötschel, L. Lovász, A. Schrijver, The ellipsoid method and its consequences in combinatorial optimization, Combinatorica 1 (2) (1981) 169–197.
[28] V.A. Gurvich, A.V. Karzanov, L.G. Khachiyan, Cyclic games and an algorithm to find minimax cycle means in directed graphs, USSR Comput. Math. Math. Phys. 28 (5) (Apr. 1990) 85–91.
[29] N. Immerman, Number of quantifiers is better than number of tape cells, J. Comput. Syst. Sci. 22 (1981) 384–406.
[30] M. Jurdzinski, R. Lazic, S. Schmitz, Fixed-dimensional energy games are in pseudo-polynomial time, in: ICALP, 2015, pp. 260–272.
[31] R. Karp, A characterization of the minimum cycle mean in a digraph, Discrete Math. 23 (1978) 309–311.
[32] S.R. Kosaraju, G.F. Sullivan, Detecting cycles in dynamic graphs in polynomial time, in: STOC, 1988, pp. 398–406.
[33] D.A. Martin, Borel determinacy, in: Annals of Mathematics, 1975, pp. 363–371.
[34] R. Niskanen, I. Potapov, J. Reichert, Undecidability of two-dimensional robot games, in: MFCS, 2016, pp. 73:1–73:13.
[35] C.H. Papadimitriou, On the complexity of integer programming, J. ACM 28 (4) (1981) 765–768.
[36] A. Pnueli, R. Rosner, On the synthesis of a reactive module, in: POPL, ACM Press, 1989, pp. 179–190.
[37] P.J. Ramadge, W.M. Wonham, Supervisory control of a class of discrete-event processes, SIAM J. Control Optim. 25 (1) (1987) 206–230.
[38] P. Černý, S. Gopi, T. Henzinger, A. Radhakrishna, N. Totla, Synthesis from incompatible specifications, in: EMSOFT'12, ACM, 2012, pp. 53–62.
[39] Y. Velner, The complexity of mean-payoff automaton expression, in: ICALP, 2012, pp. 390–402.
[40] Y. Velner, K. Chatterjee, L. Doyen, T.A. Henzinger, A.M. Rabinovich, J. Raskin, The complexity of multi-mean-payoff and multi-energy games, Inf. Comput. 241 (2015) 177–196.
[41] Y. Velner, A. Rabinovich, Church synthesis problem for noisy input, in: FOSSACS, 2011, pp. 275–289.
[42] I. Walukiewicz, Model checking CTL properties of pushdown systems, in: FSTTCS, 2000, pp. 127–138.
[43] I. Walukiewicz, Pushdown processes: games and model-checking, Inf. Comput. 164 (2) (2001) 234–263.
[44] U. Zwick, M. Paterson, The complexity of mean payoff games on graphs, Theor. Comput. Sci. 158 (1996) 343–359.