Hyperplane Separation Technique for Multidimensional Mean-Payoff Games*

Krishnendu Chatterjee[1] and Yaron Velner[2]

[1] IST Austria (Institute of Science and Technology Austria)
[2] Tel Aviv University, Israel

Abstract. Two-player games on graphs are central in many problems in formal verification and program analysis such as synthesis and verification of open systems. In this work, we consider both finite-state game graphs, and recursive game graphs (or pushdown game graphs) that model the control flow of sequential programs with recursion. The objectives we study are multidimensional mean-payoff objectives, where the goal of player 1 is to ensure that the mean-payoff is non-negative in all dimensions. In pushdown games two types of strategies are relevant: (1) global strategies, that depend on the entire global history; and (2) modular strategies, that have only local memory and thus do not depend on the context of invocation. Our main contributions are as follows: (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 the weights are fixed; whereas if the number of dimensions is arbitrary, then the problem is known to be coNP-complete. (2) We show that pushdown graphs with multidimensional mean-payoff objectives can be solved in polynomial time. For both (1) and (2) our algorithms are based on hyperplane separation technique. (3) For pushdown games under global strategies both one and multidimensional mean-payoff objectives problems are known to be undecidable, and we show that under modular strategies the multidimensional problem is also undecidable; under modular strategies the one-dimensional problem is NP-complete. We show that if the number of modules, the number of exits, and the maximal absolute value of the weights are fixed, then pushdown games under modular strategies with one-dimensional mean-payoff objectives can be solved in polynomial time, and if either the number of exits or the number of modules is unbounded, then the problem is NP-hard. (4) Finally we show that a fixed parameter tractable algorithm for finite-state multidimensional mean-payoff games or pushdown games under modular strategies with one-dimensional mean-payoff objectives would imply the fixed parameter tractability of parity games.

1 Introduction

In this work we present a hyperplane separation 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.

* 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.

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 [8,24,23]. 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 (resp. player-2 vertex), then player 1 (resp. player 2) chooses an outgoing edge. 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. Mean-payoff games and the special case of graphs (with only one player) with mean-payoff objectives have been extensively studied over the last three decades; e.g. [20,14,27,17]. Graphs with mean-payoff objectives can be solved in polynomial time [20], whereas mean-payoff games can be decided in NP $\cap$ coNP [14,27]. The mean-payoff games problem is an intriguing and rare combinatorial problem that lie in NP $\cap$ coNP, but no polynomial time algorithm is known. However, pseudo-polynomial time algorithms exist [27,7]; if the weights are constants 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. Pushdown games have been studied widely with applications in verification, synthesis, and program analysis in [26,1]. In applications of verification and synthesis, the quantitative objectives that typically arise are multidimensional quantitative 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 [9,5,13,4]. 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. 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 [3,6].

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. 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 [1]. 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 with mean-payoff objectives can be solved in polynomial time [20], and finite-state graphs with multidimensional mean-payoff objectives can also be solved in polynomial time [25] using the techniques to detect zero-circuits in graphs of [21].

2. (Finite-state games). Finite-state games with a one-dimensional mean-payoff objective can be decided in NP $\cap$ coNP [27,14], and pseudo-polynomial time algorithms exist for mean-payoff games [27,7]: 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 [7]. Finite-state games with multidimensional mean-payoff objectives are coNP-complete with weights in $\{-1, 0, 1\}$ but with arbitrary dimensions [10], 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 [12]. 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 single exit for every module and weights in $\{-1, 0, 1\}$ are NP-hard, and pushdown games with any number of exits and general weight function are in NP [12].

Many fundamental algorithmic questions have remained open for analysis of finite-state and pushdown graphs and games with multidimensional mean-payoff objectives, such as: (A) Can finite-state game graphs with multidimensional mean-payoff objectives with 2 or 3 dimensions and constant weights be solved in polynomial time?; (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?

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

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 3); the algorithm is polynomial for general weight function and arbitrary 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 hyperplane and reduce the multidimensional problem to one-dimensional problem by multiplying the 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 4). 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 [12]. It was already shown in [12] 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 5). 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 questions (C) and (D).

3. (Hardness for fixed parameter tractability). 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). We show the hardness of fixed parameter tractability problem by reducing the long-standing open problem of fixed parameter tractability of parity games to both the problems (Section 2: Theorem 2 and Section 4: Theorem 7), i.e., fixed parameter tractability of any of the above problems would imply fixed parameter tractability of parity games.

2 Finite-State Multidimensional Mean-Payoff Games

In this section we will present two results: (1) 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; (2) a reduction of finite-state parity games to finite-state multidimensional mean-payoff games with polynomial weights and arbitrary dimensions that shows that fixed parameter tractability of multidimensional mean-payoff games would imply the fixed parameter tractability of parity games.

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 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. 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 $u \in V$.

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 take 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 τ: $V^* \cdot V_1 \to V$ 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 τ: $V_1 \to V$ 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 one 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 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 $\mathsf{LimInfAvg}(\pi) \geq \mathbf{0}$, where $\mathbf{0}$ denotes the vector of all zeros. 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 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 WLOG the comparison is with $\mathbf{0}$. We will present all the results for $\mathsf{LimInfAvg}$ objectives and the results for $\mathsf{LimSupAvg}$ objectives are simpler. In sequel we will write LimAvg for $\mathsf{LimInfAvg}$. Also all the results we will present would hold if we replace the non-strict inequality ($\geq \mathbf{0}$) with a strict inequality ($> \mathbf{0}$).

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 $\mathsf{LimAvg}(\pi(v, \tau, \sigma)) \geq \mathbf{0}$. 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 $\mathsf{LimAvg}(\pi(v, \tau, \sigma)) \geq \mathbf{0}$. The winning region for a player is the largest set U such that the player has a winning strategy from U.

Intuition and Key Ideas. Our key insight to solve multidimensional mean-payoff games is to search for a hyperplane $\mathcal{H}$ such that player 2 can ensure a mean-payoff vector below $\mathcal{H}$. Intuitively, we show that if such a hyperplane exists, then any vector 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 $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots$ and $\boldsymbol{b}_\ell$ lies above the hyperplane that is normal to $\sum_{j=1}^{\ell-1} \boldsymbol{b}_j$, then $\liminf_{\ell\to\infty} \frac{1}{\ell} \cdot \sum_{j=1}^{\ell} \boldsymbol{b}_j = \mathbf{0}$.

Multiple Dimensions to One Dimension. Given a multidimensional weight function w and a vector $\boldsymbol{\lambda}$, we denote by $w \cdot \boldsymbol{\lambda}$ the one-dimensional weight function that assigns every edge e the weight value $w(e)^T \cdot \boldsymbol{\lambda}$, 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 multi-dimensional mean-payoff objective to the same game with a one-dimensional mean-payoff objective. A vector $\boldsymbol{b}$ lies above a hyperplane $\mathcal{H}$ if $\boldsymbol{\lambda}$ is the normal vector of $\mathcal{H}$ and $\boldsymbol{b}^T \cdot \boldsymbol{\lambda} \geq 0$. 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 $w(e) \cdot \boldsymbol{\lambda}$.

Examples. Consider the game graph G_1 (Figure 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 Figure 3). All the vectors reside below the hyperplane $y = -x$, and consider the normal vector $\boldsymbol{\lambda} = (1, 1)$ to the hyperplane $y = -x$.

All the cycles in G_1 with weight function $w_1 \cdot \boldsymbol{\lambda}$ have negative weights. Therefore player 1 loses in the one-dimensional mean-payoff objective. Consider the game graph G_2 (Figure 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 Figure 4). Every two-dimensional hyperplane that passes through the origin intersects with the feasible region. Thus, no separating hyperplane exists.

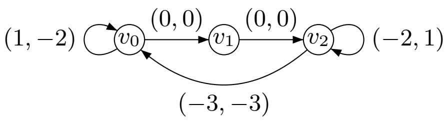

Fig. 1. Game graph G_1

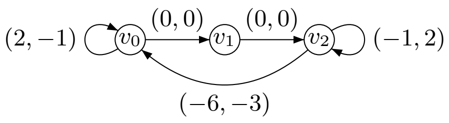

Fig. 2. Game graph G_2

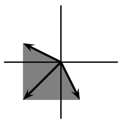

Fig. 3. Feasible vectors for G_1

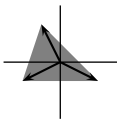

Fig. 4. Feasible vectors for G_2

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 (we refer 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 (we refer Assumption 2) is that every player-1 vertex has k self-loop edges $e_1, \ldots, e_k$ such that $w_i(e_j) = 0$ if $i \neq j$ and $w_i(e_i) = -1$. Let us denote by Win^2 the player-2 winning region in the multidimensional mean-payoff game with weight function w, and by $\mathsf{Win}^2_{\boldsymbol{\lambda}}$ the player-2 winning region in the one-dimensional mean-payoff game with the weight function $w \cdot \boldsymbol{\lambda}$. Lemma 1 shows that if $\mathsf{Win}^2_{\boldsymbol{\lambda}} \neq \emptyset$, then $\mathsf{Win}^2 \neq \emptyset$; thus gives a sufficient condition for non-emptiness of Win^2. Lemma 2 complements Lemma 1, and does not require Assumption 1 or Assumption 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 $\boldsymbol{\lambda} \in \mathbb{R}^k$ we have* $\mathsf{Win}^2_{\boldsymbol{\lambda}} \subseteq \mathsf{Win}^2$*; (hence, if* $\mathsf{Win}^2_{\boldsymbol{\lambda}} \neq \emptyset$*, then* $\mathsf{Win}^2 \neq \emptyset$*).*

Proof. Let σ be a player-2 winning strategy in G from an initial vertex v_0 for the mean-payoff objective with weight function $w \cdot \boldsymbol{\lambda}$. Due to Assumption 1 and Assumption 2 it follows that $\boldsymbol{\lambda} \in (0,\infty)^k$. We show σ is also a player-2 winning strategy wrt the multidimensional mean-payoff objective. 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 $w \cdot \boldsymbol{\lambda}$, there exists a constant $c > 0$ such that there are infinitely many prefixes of ρ with average weight (according to $w \cdot \boldsymbol{\lambda}$) at most $-c$. Let $\lambda_{\min} = \min\{\boldsymbol{\lambda}_i \mid 1 \leq i \leq k\}$ be the minimum value of $\boldsymbol{\lambda}$ among its dimension. Since $\boldsymbol{\lambda} \in (0,\infty)^k$, it follows that $\lambda_{\min} > 0$. There must be a dimension i with infinitely many prefixes of ρ with average weight at most $-\frac{c \cdot \lambda_{\min}}{k} < 0$. Hence the multidimensional objective is violated. $\square$

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

Proof. Since $\mathsf{Win}^2_{\boldsymbol{\lambda}} = \emptyset$ for every $\boldsymbol{\lambda} \in \mathbb{R}^k$, it follows by the determinacy of one-dimensional mean-payoff games [14] that for all $\boldsymbol{\lambda} \in \mathbb{R}^k$, player 1 can ensure the one-dimensional mean-payoff objective with weight function $w \cdot \boldsymbol{\lambda}$ by a memoryless strategy $\tau_{\boldsymbol{\lambda}}$ 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. We construct a player-1 winning strategy τ for the multidimensional objective in the following way:

- Initially, set $\boldsymbol{b}_0 := (1, 1, \dots, 1)$.
- For $i = 1, 2, \dots, \infty$, in iteration i play as follows:
  - Set $\boldsymbol{\lambda}_{b_i} := -\boldsymbol{b}_{i-1}$. In τ, player 1 plays according to $\tau_{\boldsymbol{\lambda}_{b_i}}$ for i rounds.
  - Let ρ_i be the play suffix that was formed in the last i rounds (or steps) of the play. 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.
  - Set $\boldsymbol{b}_i := \boldsymbol{b}_{i-1} + w(\rho_i^2)$; 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 $\boldsymbol{b}_i$. For this purpose we observe that $\tau_{\boldsymbol{\lambda}_{b_i}}$ is a memoryless winning strategy for the one-dimensional mean-payoff game with weight function $w \cdot \boldsymbol{\lambda}_{b_i}$; and hence it follows that for every cycle C in the graph $G^{\tau_{\boldsymbol{\lambda}_{b_i}}}$ the sum of the weights of C according to $w \cdot \boldsymbol{\lambda}_{b_i}$ is non-negative. Since ρ_i^2 is composed of cyclic paths, we must have $w(\rho_i^2)^T \cdot \boldsymbol{\lambda}_{b_i} \geq 0$; and hence, we have $w(\rho_i^2)^T \cdot \boldsymbol{b}_{i-1} \leq 0$. Thus

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

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 we get that $|\boldsymbol{b}_i| \leq \sqrt{|\boldsymbol{b}_{i-1}|^2 + k \cdot W^2 \cdot i^2} \leq \sqrt{k \cdot W^2 \cdot i^3}$. We are now ready to compute the the Euclidean norm of the play after the i-th iteration. We denote the weight vector after the i-th iteration by $\boldsymbol{x}_i$ and observe that $\boldsymbol{x}_i = \boldsymbol{b}_i + \sum_{j=1}^{i} w(\rho_j^1)$ and by the Triangle inequality we get that $|\boldsymbol{x}_i| \leq |\boldsymbol{b}_i| + \sum_{j=1}^{i} |w(\rho_j^1)|$. Since the length of ρ_i^1 is at most n and by the bound we obtained over $\boldsymbol{b}_i$ we get that $|\boldsymbol{x}_i| \leq \sqrt{k \cdot W^2 \cdot i^3} + i \cdot n \cdot W \cdot \sqrt{k}$. For a position j of the play between iteration i and iteration $i+1$, let us denote by $\boldsymbol{y}_j$ the weight vector after the play prefix at position j. Since there are i steps played in iteration i we have $|\boldsymbol{y}_j| \leq |\boldsymbol{x}_i| + i \cdot W \cdot \sqrt{k}$. 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, $|\frac{\boldsymbol{y}_j}{j}| \leq \frac{|\boldsymbol{y}_j|}{i \cdot (i-1)/2}$, tends to zero as i tends to infinity. It follows that the limit average of the weight vectors is zero. □

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) $\boldsymbol{\lambda} \in \mathbb{R}^k$ and check whether player-2 winning region is non-empty in the one-dimensional mean-payoff game with weight function $w \cdot \boldsymbol{\lambda}$. Lemma 3 shows that we need to consider only finitely many vectors.

Notations. For the rest of this section, we denote $M = (k \cdot n \cdot W)^{k+1}$, where W is the maximal absolute value of the weight function. For a positive integer ℓ, we will denote by $\mathbb{Z}_\ell^\pm = \{i \mid -\ell \le i \le \ell\}$ (resp. $\mathbb{Z}_\ell^+ = \{i \mid 1 \le i \le \ell\}$) 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 and a weight function w. There exists $\boldsymbol{\lambda}_0 \in \mathbb{R}^k$ for which player-2 winning region is non-empty in G for the one-dimensional mean-payoff objective with weight function $w \cdot \boldsymbol{\lambda}_0$ if and only if there exists $\boldsymbol{\lambda} \in (\mathbb{Z}_M^\pm)^k$ such that the player-2 winning region is non-empty in G for the one-dimensional mean-payoff objective with weight function $w \cdot \boldsymbol{\lambda}$.*

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 $w \cdot \boldsymbol{\lambda}_0$. Let $C_1, \dots, C_m$ 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 $w(C_i)^T \cdot \boldsymbol{\lambda}_0 < 0$ for every $i \in \{1, \dots, m\}$. We note that for all $1 \le i \le m$ we have $w(C_i) \in (\mathbb{Z}_{n \cdot W}^\pm)^k$ (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 [22, Lemma 2, items c and d] it follows that there is a vector of integers $\boldsymbol{\lambda}$ such that $w(C_i)^T \cdot \boldsymbol{\lambda} \le -1 < 0$, for all $1 \le i \le m$; and $\boldsymbol{\lambda} \in (\mathbb{Z}_M^\pm)^k$. Since all the reachable cycles from v_0 in G^σ are negative according to $w \cdot \boldsymbol{\lambda}$, we get that σ is a winning strategy for the one-dimensional mean-payoff game with weight function $w \cdot \boldsymbol{\lambda}$; and hence the proof for the direction from left to right follows. The converse direction is trivial. □

Lemma 4 removes the two assumptions of Lemma 1; the technical proof (in [11]) requires Lemma 1, Lemma 2, and Lemma 3.

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

Attractor Removal. 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 $i \ge 0$ 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 $Attr_2(U) = \bigcup_{i \ge 0} U_i$. 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 [18,2]. 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 $V \setminus Attr_2(U)$). 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 $Attr_2(U)$.

Algorithm. We now present our iterative algorithm that is based on Lemma 4 and attractor removal. In the current iteration i of the game graph execute the following steps:

sequentially iterate over vectors $\boldsymbol{\lambda} \in (\mathbb{Z}_M^+)^k$; and if for some $\boldsymbol{\lambda}$ we obtain a non-empty set U of winning vertices for player 2 for the one-dimensional mean-payoff objective with weight function $w \cdot \boldsymbol{\lambda}$ in the current game graph, remove $Attr_2(U)$ from the current game graph and proceed to iteration $i+1$. Otherwise if for all $\boldsymbol{\lambda} \in (\mathbb{Z}_M^+)^k$, player 1 wins from all vertices for the one-dimensional mean-payoff objective with weight function $w \cdot \boldsymbol{\lambda}$, 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 attractor removal. Since one-dimensional mean-payoff games with n vertices, m edges, and maximal weight W can be solved in time $O(n \cdot m \cdot W)$ [7], 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 $O(n^2 \cdot m \cdot k \cdot W \cdot (k \cdot n \cdot W)^{k^2+2\cdot k+1})$.*

Hardness for Fixed Parameter Tractability. We reduce finite-state parity games to finite-state multidimensional mean-payoff games with weights bounded linearly by the number of vertices. Note that our reduction is different from the standard reduction of parity games to one-dimensional mean-payoff games where exponential weights are necessary [19]. A parity game consists of a finite-state game graph G along with a priority function $p : E \to \{1, \ldots, k\}$ that maps every edge to a natural number (the priority). The objective of player 1 is to ensure that the *minimal* priority that occurs infinitely often in a play is *even*, and the goal of player 2 is the complement.

The Reduction. Given a game graph G with priority function p we construct a multidimensional mean-payoff objective with weight function w of k dimensions on G as follows: for every $i \in \{1, \ldots, k\}$ we assign $w_i(e)$ as follows: (i) 0 if $p(e) > i$; (ii) -1 if $p(e) \le i$ and $p(e)$ is odd; and (iii) n if $p(e) \le i$ and $p(e)$ is even. From a vertex v, player 1 wins the parity game iff she wins the multidimensional mean-payoff game.

Theorem 2. *Let G be a game graph with a parity objective defined by a priority function of k-priorities. We can construct in linear time a k-dimensional weight function w, with maximal weight W bounded by n, such that a vertex v is winning for player 1 in the parity game iff v is winning for player 1 in the multidimensional mean-payoff game.*

3 Multidimensional Mean-Payoff Pushdown Graphs

We consider pushdown graphs (pushdown systems) with multidimensional mean-payoff objectives, and give an algorithm that determines if there exists a path that satisfies a multidimensional objective. Our algorithm runs in polynomial time even for arbitrary number of dimensions and for arbitrary weight function. We again use the hyperplane separation technique to reduce the problem to one-dimensional pushdown graphs, and a polynomial solution for the latter is known [12].

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 value vector 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 [15] (a special case of Farkas' Lemma) and in Lemma 6 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 $\boldsymbol{\lambda} \cdot A < \mathbf{0}$. However, in general the matrix A can be of exponential size. Thus we need to use advanced linear-programing 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 $\boldsymbol{\lambda}$ 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 [12]), 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, $push(z)$ 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 (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 $(\bot, q_0)$.

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 $1 \leq i \leq n$ 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. A path can equivalently be defined as a sequence $\langle c_1 e_1 e_2 \ldots e_n \rangle$, where c_1 is the initial configuration and e_i are valid transitions. We will present an algorithm that given a WPS $\mathcal{A}$ decides if there exists an infinite path π in $\mathcal{A}$ from q_0 such that $\mathsf{LimAvg}(\pi) \geq \mathbf{0}$.

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.

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: $(p_1, p_2) = (e_{i_1} e_{i_1+1} \ldots e_{i_1+n_1}, e_{i_2} e_{i_2+1} \ldots e_{i_2+n_2})$, for $n_1, n_2 \geq 0, i_1 \geq 0$ and $i_2 > i_1 + n_1$ such that for every $j \geq 0$ 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.

Local Minimum 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 minimum* if for every $j \geq i$ we have $\alpha_i \sqsubseteq \alpha_j$ (i.e., the stack string α_i is a prefix string of α_j). One basic fact: Every infinite path has infinitely many local minimum.

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 α-path* if $(\alpha\gamma, q_1)$ is a local minimum. Note that if π is a non-decreasing α-path for some $\alpha \in \Gamma^*$, then the same sequence of transitions leads to a non-decreasing β-path for every $\beta \in \Gamma^*$. Hence we say that π is a non-decreasing path if there exists $\alpha \in \Gamma^*$ such that π is a non-decreasing α-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 denote $\mathbb{R}_+ = [0, +\infty)$. 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 $\mathsf{PPS}(\pi) = \{P_1 = (p^1_1, p^1_2), P_2 = (p^2_1, p^2_2), \ldots, P_j = (p^j_1, p^j_2)\}$, 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 $w(p^i_1) + w(p^i_2)$ (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, $\mathsf{PCone}(\pi) = \{\mathsf{PumpMat}(\pi) \cdot \boldsymbol{x} \mid \boldsymbol{x} \in (\mathbb{R}^k_+ \setminus \{\mathbf{0}\})\}$.

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 pair 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 $\mathsf{PumpMat}$ and PCone. In Proposition 1 we establish 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* $\mathsf{LimInfAvg}(\pi) \geq \mathbf{0}$ *if and only if there exists a (reachable) non-decreasing cycle π such that* $\mathbb{R}^k_+ \cap \mathsf{PCone}(\pi) \neq \emptyset$.

By Proposition 1, we can decide whether there is an infinite path π for which $\mathsf{LimAvg}(\pi) \geq \mathbf{0}$ by checking if there exist a tuple $(\gamma, q) \in \Gamma \times Q$ for which there is a non-negative (and non-trivial) solution for the equation $\mathsf{PumpMat}((\gamma, q)) \cdot \boldsymbol{x} \geq 0$. 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 $\mathsf{PumpMat}((\gamma, q)) \cdot \boldsymbol{x} = 0$. 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 $\boldsymbol{\lambda}$ we denote by $w \cdot \boldsymbol{\lambda}$ the one-dimensional weight function obtained by multiplying the weight vectors with $\boldsymbol{\lambda}$. The reduction to one-dimensional objective (Lemma 6) requires the use of Gordan's lemma (Lemma 5).

Lemma 5 ([15]). *For a matrix A, either $A \cdot \boldsymbol{x} = \mathbf{0}$ has a non-trivial non-negative solution for $\boldsymbol{x}$, or there exists a vector $\boldsymbol{y}$ such that $\boldsymbol{y} \cdot A^T$ is negative in every dimension.*

Lemma 6. *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* $\mathsf{PumpMat}((\gamma, q)) \cdot \boldsymbol{x} = 0$ *if and only if for every $\boldsymbol{\lambda} \in \mathbb{R}^k$ there is a non-decreasing path from (γ, q) to (γ, q) that contains a pumpable pair $P = (p_1, p_2)$ such that $(w \cdot \boldsymbol{\lambda})(P) \geq 0$.*

Proposition 2. *There is a polynomial time algorithm that given WPS $\mathcal{A}$ with k-dimensional weight function w, $(\gamma, q) \in \Gamma \times Q$, a vector $\boldsymbol{\lambda} \in \mathbb{Q}^k$, 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 $\frac{(w \cdot \boldsymbol{\lambda})(P)}{|P|} > r$ 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 now show how to use the result of the proposition and a result from linear programming to solve the problem.

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 [16]. We use this fact 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* $\mathsf{PumpMat}((\gamma, q)) \cdot \boldsymbol{x} = 0$.

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 $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_k)$, the objective function is to minimize r subject to the constraints below: (i) $\boldsymbol{\lambda} \cdot A^T \leq \boldsymbol{r}$ where $\boldsymbol{r} = (r, r, \dots, r)^T$; (ii) $\sum_{i=1}^{k} \lambda_i = 1$. Once the minimal r is computed, by Lemma 6, there exists a solution for $\mathsf{PumpMat}((\gamma, q)) \cdot \boldsymbol{x} = 0$ 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 $\boldsymbol{\lambda}$ 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. $\square$

Theorem 3. *Given a WPS $\mathcal{A}$ with multidimensional weight function w, we can decide in polynomial time whether there exists a path π such that* $\mathsf{LimAvg}(\pi) \geq \mathbf{0}$.

4 Recursive Mean-Payoff Games with Modular Strategies

In this section, we will consider recursive games (which are equivalent to pushdown games) with modular strategies.

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: (i) A finite nonempty set of *nodes* N_i. (ii) A nonempty set of *entry* nodes $En_i \subseteq N_i$ and a nonempty set of *exit* nodes $Ex_i \subseteq N_i$. (iii) A set of *boxes* B_i. (iv) 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. (v) A labeling $Y_i : B_i \to \{1, \ldots, n\}$ that assigns to every box an index of the game modules $A_1 \ldots A_n$. (vi) 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 [1] for detailed description and illustration with figures of recursive game graphs. 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). The module A_0 is the initial module, and its entry node the starting node of the game.

Modular Strategies. Intuitively, a modular strategy only depends on the local history, and not on the context of invocation of the module. 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 $\mathsf{LimAvg}(\pi) \geq \mathbf{0}$ (note that the counter strategy of player 2 is a general strategy).

Undecidability. We show the following undecidability result, and in view of it will focus on WRGs under modular strategies for one-dimensional mean-payoff objectives.

Theorem 4. *The problem of deciding the existence of a modular winning strategy in WRGs with multidimensional mean-payoff objectives is undecidable, with six dimensions, three modules and with at most one exit for each module.*

NP-hardness. We consider WRGs under modular strategies with one-dimensional mean-payoff objectives. It was already shown in [12] that if the number of modules is unbounded, then even if all modules have at most one exit, the problem is NP-hard even for one player game with weights restricted to $\{-1, 0, 1\}$. We present a similar hardness result by a reduction from 3SAT.

Theorem 5. *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* $\{0, -1\}$.

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.

Description of the Key Steps of the Algorithm. One key result that we use is the fact that if there is a winning modular strategy, then there is a memoryless one [12]. The intuitive idea of our algorithm is to consider a *signature* function for a strategy that assigns to every exit of a module the weight of the *worst* sub-play from the entrance of the module to the exit. We show that for a memoryless modular winning strategy the range of the signature function is bounded (in absolute value) by $(n \cdot \mathsf{M})^{\mathsf{M}\cdot\mathsf{E}+1} \cdot W$. The final step is to show that for a given a signature function, there is a memoryless modular strategy to satisfy the signature function. The verification is achieved by solving a finite-state mean-payoff game. The detailed formal description is presented in [11].

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.*

Hardness for Fixed Parameter Tractability. Given Theorem 6 (algorithm to solve in polynomial time when M and E are fixed) an interesting question is whether it is possible to show that WRGs under modular strategies is fixed parameter tractable (i.e., to obtain an algorithm that runs in time $O(f(\mathsf{M}, \mathsf{E}) \cdot \text{poly}(n, m, W))$). We show the hardness of fixed parameter tractability, again by a reduction from parity games, implying that fixed parameter tractability would imply the solution of the long-standing open problem of fixed parameter tractability of parity games (details presented in [11]).

Theorem 7. *Given a finite-state parity game G with n vertices and priority function of k-priorities, we can construct in polynomial time a WRG $\mathcal{A}$ with $2 \cdot k + 1$ modules, $O(k \cdot n)$ nodes, and weights in $\{-1, 0, +1\}$ such that a vertex v is winning for player 1 in the parity game iff there is a modular winning strategy in $\mathcal{A}$ with v as the initial node.*

References

1. Alur, R., La Torre, S., Madhusudan, P.: Modular strategies for recursive game graphs. Theor. Comput. Sci. 354(2), 230–249 (2006)

2. Beeri, C.: On the membership problem for functional and multivalued dependencies in relational databases. ACM Trans. on Database Systems 5, 241–259 (1980)
3. Bloem, R., Chatterjee, K., Henzinger, T.A., Jobstmann, B.: Better quality in synthesis through quantitative objectives. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 140–156. Springer, Heidelberg (2009)
4. Bohy, A., Bruyère, V., Filiot, E., Raskin, J.-F.: Synthesis from ltl specifications with mean-payoff objectives. In: Piterman, N., Smolka, S.A. (eds.) TACAS 2013. LNCS, vol. 7795, pp. 169–184. Springer, Heidelberg (2013)
5. Boker, U., Chatterjee, K., Henzinger, T.A., Kupferman, O.: Temporal specifications with accumulative values. In: LICS, pp. 43–52 (2011)
6. Brázdil, T., Chatterjee, K., Kučera, A., Novotný, P.: Efficient controller synthesis for consumption games with multiple resource types. In: Madhusudan, P., Seshia, S.A. (eds.) CAV 2012. LNCS, vol. 7358, pp. 23–38. Springer, Heidelberg (2012)
7. Brim, L., Chaloupka, J., Doyen, L., Gentilini, R., Raskin, J.-F.: Faster algorithms for mean-payoff games. Formal Methods in System Design 38(2), 97–118 (2011)
8. Büchi, J.R., Landweber, L.H.: Solving sequential conditions by finite-state strategies. Transactions of the AMS 138, 295–311 (1969)
9. Chatterjee, K., Doyen, L., Henzinger, T.A.: Quantitative languages. ACM ToCL (2010)
10. Chatterjee, K., Doyen, L., Henzinger, T.A., Raskin, J.-F.: Generalized mean-payoff and energy games. In: FSTTCS, pp. 505–516 (2010)
11. Chatterjee, K., Velner, Y.: Finite-state and pushdown games with multi-dimensional mean-payoff objectives. CoRR, abs/1210.3141 (2012)
12. Chatterjee, K., Velner, Y.: Mean-payoff pushdown games. In: LICS (2012)
13. Droste, M., Meinecke, I.: Describing average- and longtime-behavior by weighted MSO logics. In: Hliněný, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 537–548. Springer, Heidelberg (2010)
14. Ehrenfeucht, A., Mycielski, J.: Positional strategies for mean payoff games. IJGT 8, 109–113 (1979)
15. Gordan, P.: Ueber die auflosung linearer gleichungen mit reellen coeffizienten. Mathematische Annalen 6, 23–28 (1873)
16. Grötschel, M., Lovász, L., Schrijver, A.: The ellipsoid method and its consequences in combinatorial optimization. Combinatorica 1(2), 169–197 (1981)
17. Gurvich, V.A., Karzanov, A.V., Khachiyan, L.G.: Cyclic games and an algorithm to find minimax cycle means in directed graphs. USSR Comp. Math. Phys. 28(5), 85–91 (1990)
18. Immerman, N.: Number of quantifiers is better than number of tape cells. Journal of Computer and System Sciences 22, 384–406 (1981)
19. Jurdzinski, M.: Deciding the winner in parity games is in UP $\cap$ co-UP. IPL 68 (1998)
20. Karp, R.M.: A characterization of the minimum cycle mean in a digraph. Discrete Mathematics 23, 309–311 (1978)
21. Kosaraju, S.R., Sullivan, G.F.: Detecting cycles in dynamic graphs in polynomial time. In: STOC, pp. 398–406 (1988)
22. Papadimitriou, C.H.: On the complexity of integer programming. JACM 28, 765–768 (1981)
23. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: POPL, pp. 179–190. ACM Press (1989)
24. Ramadge, P.J., Wonham, W.M.: Supervisory control of a class of discrete-event processes. SIAM Journal of Control and Optimization 25(1), 206–230 (1987)
25. Velner, Y., Rabinovich, A.: Church synthesis problem for noisy input. In: Hofmann, M. (ed.) FOSSACS 2011. LNCS, vol. 6604, pp. 275–289. Springer, Heidelberg (2011)
26. Walukiewicz, I.: Pushdown processes: Games and model-checking. Inf. Comput. 164(2), 234–263 (2001)
27. Zwick, U., Paterson, M.: The complexity of mean payoff games on graphs. Theoretical Computer Science 158, 343–359 (1996)