Deterministic and Probabilistic Complexity of Mean Payoff Games

Abstract. In this thesis, we consider three computational problems on mean payoff games on weighted digraphs: (1) finding winning positions, (2) computing characteristic values, and (3) determining optimal strategies. We give a detailed analysis of the relations between (1), (2), and (3). These three problems are discussed from multiple perspectives following several strategies to develop faster methods, one of which selects candidate positional strategies and provides additional edges by means of reachability between cycles. We provide limiting examples for all methods. Additionally, for a fixed number of vertices, we prove that Zwick and Paterson's algorithm to determine the winning positions runs in $O(\log M)$ time with error probability going to zero as M goes to infinity, where M is the maximum of the absolute values of the weights.

Keywords — mean payoff games, reachability games, P vs. NP, parity games, two-player games

University of Groningen
Master Thesis Mathematics
July 13, 2022
Student: Robert Modderman
First supervisor: prof. dr. O. Lorscheid
Second supervisor: prof. dr. D. Grossi

Acknowledgements

A full year of doing a master's project has come to an end, and finalizing this work also marks the end of eight wonderful years I've had as a student at the University of Groningen. An era which I started off as a first year student Applied Physics I am now closing off with a master thesis in Mathematics. A vast amount of people are co-responsible for the completion of this work. Some directly, others more indirectly. Here, I'd like to take a moment to thank all of those people.

First and foremost, I wish to thank my supervisor, Oliver. I'd like to thank him for coming up with this fascinating problem that has very intensely kept me busy for the past year. I'd also like to thank him for both seeing potential in ideas I came up with myself and, at the same time, for seeing the necessity to get other tasks done as well. Most of my gratitude towards Oliver, however, is due to his overall guidance of the process of the past year and his patience therein.

I would also like to thank my second supervisor, Davide, for his interest in the topic, for philosophizing about complexity classes several subproblems might or might not lie in, and for coming up with useful tools to analyze complexity of algorithms.

Second, I'd like to thank a couple of study friends with whom I've been working together in the final project room for Mathematics on the second floor of the Bernoulliborg for the greatest deal of time in the year '21–'22. Specifically, I would like to thank Emile, Jeroen, Manoy, Oscar, Robbert, Ruben, and Sven. For the interesting discussions we've had, for occasionally doing some mathematics on the whiteboard that sometimes was very serious and sometimes very recreational, for a spare game of chess, and the like. A special shout-out to Jeroen for coming up with a very practical tool to typeset graphs in LaTeX– you have saved me a lot of time. Another shout-out is to Robbert, for his willingness to proofread my draft version – even though in the end I never came to send my draft version to him.

Next, I'd like to thank a couple of friends I've met at this university, ranging from those whom I've known since the dawn of time in 2014 up to those whom I've only met in the past year. Specifically, I'd like to thank Robbert (once again), Gijsbert, Hans, Pieter, Aoibhin, Vladimir, Oluchi, Diana, and Luca. Thank you for having made my days of the past years extra bright, and for occasionally – without knowing – giving me that extra bit of motivation in life in general as well as for completing this thesis.

Furthermore, a sincere thank you to all people who were present at the final presentation of my thesis. This event marked the end of my time as a student, and I feel grateful towards those who took the effort to be present and to share that moment.

I wish to thank my family for always being there. Specifically, I wish to thank my parents and my sister. Finally, I'd like to thank two close friends of mine whom I've met outside academic contexts – Yannick, and Ziad. I'd like to thank both for being good friends.

Dankjulliewel — Robert Modderman, Groningen, Wednesday July 13, 2022

Contents

Notation

Below, we introduce the most important notational conventions.

Logic	
$[\varphi]$	for a propositional formula φ, $[\varphi] = 0$ if φ is false and $[\varphi] = 1$ if φ is true
$[\varphi, A, B]$	A if φ is true and B if φ is false, for any objects A, B
$\neg\varphi$	the negation of a propositional formula φ
Sets and maps	
$\subset$, $\subsetneq$, $\sqcup$	subset, strict subset, and disjoint union, respectively
$\lvert X\rvert$	the cardinality of a set X
$\lvert\boldsymbol{x}\rvert$	the length of a vector $\boldsymbol{x}$
$A \times B$	the Cartesian product of two sets A and B
X^n	the n-fold Cartesian product of the set X with itself
$x_1 \cdots x_n$	short-hand for $(x_1, \ldots, x_n) \in X^n$ where X is any set
Y^X	the set of maps $f : X \to Y$ where X, Y are sets
$f \sqcup g$	for $f : X \to Y$ and $g : Z \to W$ with $f\lvert_{X\cap Z} = g\lvert_{X\cap Z}$, the map sending $a \in X \cup Z$ to $f(a)$ if $a \in X$ and to $g(a)$ if $a \in Z$
$\overline{A}$	the set $A \cup \{\pm\infty\}$
$\mathbf{R}$	the set of real numbers
$\mathbf{N}$	the set of positive integers
$[n]$	the set $\{i \in \mathbf{N} : 1 \leqslant i \leqslant n\}$ where $n \in \mathbf{N}$
Real analysis	
$\liminf$, $\limsup$	limit inferior and limit superior, respectively
inf, sup	infimum and supremum, respectively
$\lfloor x\rfloor$, $\lceil x\rceil$	for $x \in \mathbf{R}$, $\max\{n \in \mathbf{Z} : n \leqslant x\}$ and $\min\{n \in \mathbf{Z} : n \geqslant x\}$, respectively
$\lVert\boldsymbol{x}\rVert_p$	the p-norm of a vector $\boldsymbol{x} \in \mathbf{C}^n$: $\lVert\boldsymbol{x}\rVert_p := (\sum_{i=1}^n \lvert x_i\rvert^p)^{1/p}$, where $p \in [1, \infty)$
$\lVert\boldsymbol{x}\rVert_\infty$	the infinity norm or maximum norm of $\boldsymbol{x} \in \mathbf{C}^n$: $\max_{1\leqslant i\leqslant n} \lvert x_i\rvert$
Graphs	
$V(G), E(G)$	the vertex- and edge set of a graph G
$t(e)$, $h(e)$	the tail and head of an edge e of a graph, respectively
vS, Sv	the sets $\{vw : w \in S\}$ and $\{uv : u \in S\} \subset V \times V$, where $v \in V$ and $S \subset V$
$N^-(v)$, $d^-(v)$	the set of in neighbours and the in degree of $v \in V(G)$, respectively
$N^+(v)$, $d^+(v)$	the set of out neighbours and the out degree of $v \in V(G)$, respectively
S^*	for $S \subseteq V(G)$ for a graph G, the set $\{v \in S : d^+(v) > 0\}$
R	the least nonzero distance to zero of the mean weight of a directed cycle
Δm	the least nonzero distance between mean weights of directed cycles

We set a few other conventions. To the *empty sum* we assign the value zero, and to the *empty product* the value one. The maximum of the empty set is $\max \varnothing = -\infty$, and the minimum of the empty set is $\min \varnothing = +\infty$.

Introduction

A mean payoff game on an edge-weighted, finite directed graph (G, μ) where $\mu : E(G) \to \mathbf{R}$, with partition[1] $V(G) = V_- \sqcup V_+$, is a two-player game on (G, μ) starting at a particular vertex $v \in V(G)$. Two players, Min and Max[2], construct a path, where each next edge is determined by the player who owns the vertex the most recent edge ended in. Here, V_- is Min's collection of vertices and V_+ is Max's collection of vertices. As a function of these weights, to such paths we can assign scores in a certain way. Min's objective is to minimize this score, whereas it is Max's to maximize.

To any path $\boldsymbol{\sigma}$, we assign values $\chi_-(\boldsymbol{\sigma})$ and $\chi_+(\boldsymbol{\sigma})$ as follows:

- if $\boldsymbol{\sigma}$ is finite, then we set $\chi_-(\boldsymbol{\sigma}) = \chi_+(\boldsymbol{\sigma}) = -p \cdot \infty$ if player p owns the final vertex of $\boldsymbol{\sigma}$, where either $p = -$ (Min) or $p = +$ (Max);
- if $\boldsymbol{\sigma}$ is infinite, then we define

$$\chi_-(\boldsymbol{\sigma}) := \liminf_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \mu(\sigma_i) \text{ and } \chi_+(\boldsymbol{\sigma}) := \limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \mu(\sigma_i)$$

  where σ_i is the i-th edge of $\boldsymbol{\sigma}$, for $i \in \mathbf{N}$.

If a game starts at a vertex v from which a path $\boldsymbol{\sigma}$ is constructed, then we say that Min wins if $\chi_+(\boldsymbol{\sigma}) < 0$, that Max wins if $\chi_-(\boldsymbol{\sigma}) > 0$, and that Min and Max achieve a draw if $\chi_-(\boldsymbol{\sigma}) \leqslant 0 \leqslant \chi_+(\boldsymbol{\sigma})$. In this thesis, along with two fundamental papers [8] and [10], we restrict our attention to digraphs with integer weights.

It has been shown in [8] that for every vertex v, there exists a value $\chi(v) \in \overline{\mathbf{R}} := \mathbf{R} \cup \{\pm\infty\}$ such that Min has a strategy ensuring $\chi_+(\boldsymbol{\sigma}) \leqslant \chi(v)$ for every path $\boldsymbol{\sigma}$ following that strategy, irrespective of Max's choices of moves, and that Max has a strategy ensuring $\chi_-(\boldsymbol{\sigma}) \geqslant \chi(v)$ in a similar fashion. From here onward, we always assume that Min and Max play optimally, and hence that $\chi_-(\boldsymbol{\sigma}) = \chi(v) = \chi_+(\boldsymbol{\sigma})$ for every optimal path $\boldsymbol{\sigma}$ starting at v. It was furthermore shown in [8] that Min and Max can ensure these inequalities using *positional strategies*: that is, independent of the starting vertex, each next move is predetermined in a memoryless fashion and hence only dependent on the graph data. Positional strategies ensuring these inequalities are said to be *optimal*.

In this setting, three main computational problems on mean payoff games have been formulated and studied. We list them in ascending order of apparent difficulty:

(1) deciding who wins at which starting vertices: "The Decision Problem",

(2) computing the values $\chi(v)$ for $v \in V(G)$: "The Value Finding Problem",

(3) determining optimal positional strategies for both players.

The fact that a solution to the second problem immediately gives a solution to the first is apparent from the fact that the situations that Min wins at v, Min and Max achieve a draw at v, and Max wins at v, align with the possible outcomes $-, 0, +$ of the sign of $\chi(v)$, respectively. Furthermore, the values $\chi(v)$

[1] The graph need not be bipartite with respect to this partition

[2] Alternatively, these players can be called players minus $(-)$ and plus $(+)$, respectively

can be efficiently computed from optimal positional strategies. This fact was implicitly agreed upon but is formally proven in this thesis.

What makes mean payoff games interesting from a complexity-theoretic point of view is that the problem of finding the winning positions resides in NP$\cap$co $-$ NP [12, Thm 4.2] but is not known to be in P. Hence, this problem is a candidate to break the P vs. NP conjecture, which asserts that the complexity classes P and NP coincide: P = NP. This fact has been more than enough reason for many mathematicians and computer scientists to try to find efficient algorithms to the third problem or to prove that efficient algorithms to one of the first two (or three) problems do not exist.

The complexity of mean payoff games can be made more precise. There are two main parameters of interest, namely the number of vertices n and the integer M, which is taken to be the maximum among the absolute values of the weights and is called the *weight size*. Worst-case running times are given in big-O notation $O(f(n, M))$, where $f : \mathbf{N} \times \mathbf{N} \to \mathbf{R}_{\geqslant 0}$. Occasionally, the number $|E|$ of edges is taken into consideration in running time analyses. However, as $|E| = O(n^2)$ for digraphs for which between two vertices there are at most two edges (one in the one and one in the other direction, at the very most), the number $|E|$ is often discarded. The reason lies in the fact that an algorithm on mean payoff games is seen as *polynomial-time* whenever it is polynomial in n and $\log M$, and is seen as *pseudo-polynomial* (in M) if the running time dependence on M is polynomial in M.

One of the best deterministic algorithm sfor the decision problem stems from [12, Thm. 2.4], and runs in $O(n^4 \cdot M)$ time, and is fully incorporated in Algorithm 4. In Section 5 we prove that for fixed n, any function $h(M)$ of $M \in \mathbf{N}$ satisfying[3] $h(M) = \omega(1)$ has the property that Algorithm 4 terminates in $O(h(M))$ time with probability approaching one as $M \to \infty$. Here, this probability is taken with respect to selecting mean payoff games on n vertices uniformly at random. A similar result potentially holds for one of the best known algorithms that compute the values of mean payoff games, as argued in Section 6.

The core observation responsible for this probabilistic result is the following. We let R denote the minimum of the mean weights of all directed cycles with nonzero weights in a weighted digraph. We prove in Section 2 that Algorithm 4 terminates with absolute certainty in $O(h(M))$ steps for a mean payoff game played on a weighted graph whose value of R is at least $M/h(M)$. The observation then is that upon selecting weighted digraphs on n vertices uniformly at random, the probability that a weighted digraph has a value of R of at least $M/h(M)$ goes to one as $M \to \infty$. In Section 6 we argue that a similar probabilistic speed-up potentially holds for the algorithm described in [12, Thm. 2.3] that solves the value finding problem. For this problem, not the quantity R but the quantity Δm will likely be relevant, where Δm is defined to be the least nonzero difference between mean weights of directed cycles.

The hardest problem however, that of finding optimal positional strategies, did not allow for such probabilistic speed-up. The reason that we were able to do these speed-ups for problems (1) and (2) lies in the fact that Algorithm 4 is an approximation algorithm. Since on a mean payoff game all characteristic

[3] That is, $h(M) \to \infty$ as $M \to \infty$

values are in a set of the form $\overline{A} = A \cup \{\pm\infty\}$ where $A \subset \mathbf{Q}$ is finite, [12, Thm. 2.2] (or, phrased slightly differently, Theorem 2.14) implies that the approximation in the algorithm involved terminates to succesfully find the exact values. For the problem of finding optimal strategies, which is unlike the other two problems even exponential in n (but polynomial in $\log M$), no algorithm that might terminate earlier with some (small) error rate is known.

In Section 4 we discuss some deterministic methods that might have potential in some instances. Specifically, we utilize the concept of *derived mean payoff games* as defined in Definition 4.13 as a means to reduce the complexity of the mean payoff game in question (specifically, the numbers n and M) while keeping winning positions invariant.

Throughout this thesis, the adjective "efficient" is often used to classify algorithms with respect to running time. This is not a formal definition, yet an algorithm being efficient can be understood as the algorithm terminating in polynomial running time with respect to all complexity parameters involved.

1 Preliminaries

In this section we start with a review of the most important concepts and tools in the analysis of sequences and real numbers. Next, we briefly review the fundamentals of algorithmic complexity theory. At the end of this section, since mean payoff games are two-player games on finite weighted directed graphs, we review the basics of graph theory as well as those of general two-player games on directed graphs.

1.1 Sequences

A *sequence* is an infinite list of objects $\boldsymbol{x} = (x_1, x_2, x_3, \ldots)$, and a *vector* of length $n \in \mathbf{Z}_{\geqslant 0}$ is a finite list of objects $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$. We also define the *empty vector* $()$, which is taken to be the unique vector of length zero. Typically, a sequence will be indexed by the set of positive integers $\mathbf{N}$ or by a subset thereof of the form $\{r, r+1, \ldots\}$ where $r \in \mathbf{N}$. A sequence is often seen as function from its indexing set to (a superset of) the set of its members. Similarly, a vector of length n is often seen as a function from $[n] := \{1, \ldots, n\}$ to (a superset of) the set of its members.

Given a sequence $\boldsymbol{x}$ and vectors $\boldsymbol{y}$ and $\boldsymbol{z}$ of lengths $n, m \in \mathbf{N}$, respectively, we can form the *concatenations* $\boldsymbol{yx}$ and $\boldsymbol{yz}$, where $\boldsymbol{yx}$ is the sequence defined by $(y_1, y_2, \ldots, y_n, x_1, x_2, x_3, \ldots)$ and $\boldsymbol{yz}$ is the vector of length $n + m$ defined by $(y_1, y_2, \ldots, y_n, z_1, z_2, \ldots, z_m)$. Note that we can always *pre-concatenate* a vector with a sequence, but not *post-concatenate*. Furthermore, two vectors can always be concatenated with one other in any order, and two sequences cannot be concatenated with one another at all.

Next, we are in place to take a look at two very special types of sequences, called *periodic sequences* and *eventually periodic sequences*. A sequence $\boldsymbol{x}$ is said to be *periodic* if it is of the form $\boldsymbol{yyy}\cdots$ where $\boldsymbol{y}$ is a vector of length L. The smallest such L is called the *period length* of $\boldsymbol{x}$ and the vector $\boldsymbol{y}$ is called the *period* of x. A sequence is said to be *eventually periodic* if it is of the form $\boldsymbol{yz}$ where $\boldsymbol{y}$ is a vector and $\boldsymbol{z}$ is a periodic sequence. The shortest such $\boldsymbol{y}$ is called the *tail* of $\boldsymbol{yz}$. The *eventual period length* and the *eventual period* of $\boldsymbol{yz}$ are defined to be the period length and the period of the sequence $\boldsymbol{z}$, respectively. Note that a periodic sequence $\boldsymbol{x}$ can be seen as an eventually periodic sequence $()\boldsymbol{x}$.

Finally, we introduce notation regarding to erasing members of vectors and sequences. Given $A \subset \mathbf{N}$, a vector or sequence $\boldsymbol{x}$ can be transformed into another vector or sequence $\boldsymbol{x}_A$ by erasing all members indexed by $\mathbf{N} \backslash A$ from $\boldsymbol{x}$ whilst keeping the members indexed by A in order. For sets of the form $[n]$ ($n \in \mathbf{N}$) we set $\boldsymbol{x}_{\leqslant n} := \boldsymbol{x}_{[n]}$ and for sets of the form $A = \{i \in \mathbf{N} : i \geqslant n\}$ ($n \in \mathbf{N}$) we set $\boldsymbol{x}_{\geqslant n} := x_A$.

1.2 Real Analysis

Here, we study sequences of real numbers $\boldsymbol{x} = (x_1, x_2, x_3, \ldots)$ as well as their associated *sequences of mean values* $\overline{\boldsymbol{x}}$ where for $n \in \mathbf{N}$ we define $\overline{x}_n := \frac{1}{n}\sum_{i=1}^{n} x_i$. For a *converging* sequence of real numbers $\boldsymbol{x}$ we write $x = \lim \boldsymbol{x} = \lim_{n\to\infty} x_n$.

Take a sequence of real numbers $\boldsymbol{x}$. The *limit inferior* of $\boldsymbol{x}$, denoted by $\liminf_{n\to\infty} x_n$, is defined to be the limit of the sequence whose n-th term is given by the infimum of the set $\{x_m : m \geqslant n\}$.

Similarly, the *limit superior* of $\boldsymbol{x}$, denoted by $\limsup_{n\to\infty} x_n$, is defined to be the limit of the sequence whose n-th term is given by the supremum of the set $\{x_m : m \geqslant n\}$. In more concise notation, $\liminf_{n\to\infty} x_n := \lim_{n\to\infty} \inf_{m\geqslant n} x_m$ and $\limsup_{n\to\infty} x_n := \lim_{n\to\infty} \sup_{m\geqslant n} x_m$. We occasionally write $\liminf \boldsymbol{x} = \liminf_{n\to\infty} x_n$ and $\limsup \boldsymbol{x} = \limsup_{n\to\infty} x_n$.

We say that $\boldsymbol{x}$ is *bounded from above* if $\boldsymbol{x}$ viewed as a subset of $\mathbf{R}$ is bounded from above. That is, if there exists a constant $c \in \mathbf{R}$ such that $x_n \leqslant c$ for all $n \in \mathbf{N}$. Similarly, $\boldsymbol{x}$ is said to be *bounded from below* if there exists a constant $c \in \mathbf{R}$ such that $x_n \geqslant c$ for all $n \in \mathbf{N}$. A sequence of real numbers is said to be *bounded* if it is both bounded from above and from below.

An important observation at this point is that a bounded sequence $\boldsymbol{x}$ has both its limit inferior and limit superior. Indeed: the sequence whose n-th term is given by $\inf_{m\geqslant n} x_m$ is a sequence of real numbers (as $\boldsymbol{x}$ is bounded from below), is monotonically increasing, and is bounded from above by $\sup \boldsymbol{x} \in \mathbf{R}$. Hence, this sequence convergences (to $\liminf_{n\to\infty} x_n$) by Theorem 1.1. A similar argument shows that $\limsup_{n\to\infty} x_n$ exists.

Theorem 1.1 (Monotone Convergence Theorem)**.** *Let $\boldsymbol{x}$ be a* monotonically increasing *sequence. I.e., $x_n \leqslant x_{n+1}$ for every $n \geqslant 1$. If $\boldsymbol{x}$ is bounded from above, then* $\lim \boldsymbol{x}$ *exists and equals* $\sup \boldsymbol{x}$*. Similarly, if $\boldsymbol{y}$ is a* monotonically decreasing *sequence (i.e., $y_n \geqslant y_{n+1}$ for every $n \geqslant 1$) and is bounded from below then* $\lim \boldsymbol{y}$ *exists and equals* $\inf \boldsymbol{y}$*.*

Proof. The first part is proven in [1, Thm. 2.4.2]. The second part can be proven by applying the first part to the sequence $-\boldsymbol{y}$ given by $(-y_1, -y_2, -y_3, \ldots)$. □

An elementary example shows that the limit inferior and limit superior of a real sequence might both exist but not coincide. Let $\boldsymbol{x} = (-1, 1, -1, 1, \ldots)$. It is clear that $\inf_{m\geqslant n} x_m = -1$ and $\sup_{m\geqslant n} x_m = 1$ for all $n \in \mathbf{N}$. Hence, $\liminf \boldsymbol{x} = -1 < 1 = \limsup \boldsymbol{x}$. We also note that $\boldsymbol{x}$ does not converge. It turns out that a sequence converges if and only if both its limits inferior and superior exist and coincide.

Proposition 1.2. *Let $\boldsymbol{x}$ be a sequence of real numbers. If $\boldsymbol{x}$ converges, then* $\liminf \boldsymbol{x} = \lim \boldsymbol{x} = \limsup \boldsymbol{x}$*. If* $\liminf \boldsymbol{x} = \limsup \boldsymbol{x}$*, then $\boldsymbol{x}$ converges to* $\liminf \boldsymbol{x} = \limsup \boldsymbol{x}$*.*

Proof. If $\boldsymbol{x}$ converges, then $x = \lim \boldsymbol{x}$ exists. Let $\varepsilon > 0$. Since $\boldsymbol{x}$ converges to x, there exists an $N \in \mathbf{N}$ such that $|x - x_n| < \varepsilon/2$ for all integers $n \geqslant N$. Hence, $\{x_n : n \geqslant N\} \subset (x - \varepsilon/2, x + \varepsilon/2)$. It follows that $\sup_{m\geqslant n} x_m \in (x - \varepsilon, x + \varepsilon)$ for all $n \geqslant N$. Since $\varepsilon > 0$ was arbitrary, it must be that the sequence $(\sup_{m\geqslant n} x_m)_{n=1}^{\infty}$ converges to x and hence $\limsup \boldsymbol{x} = x$. The fact that $\liminf \boldsymbol{x} = x$ is proven similarly. Now, assume that $\liminf \boldsymbol{x} = \limsup \boldsymbol{x}$ exist. Given $n \in \mathbf{N}$, we note that $\inf_{m\geqslant n} x_m \leqslant x_n \leqslant \sup_{m\geqslant n} x_m$. Since the sequences $(\inf_{m\geqslant n} x_m)_{n=1}^{\infty}$ and $(\sup_{m\geqslant n} x_m)_{n=1}^{\infty}$ are known to converge to the same limit $\liminf \boldsymbol{x} = \limsup \boldsymbol{x}$, it follows from lemma 1.3 that $\boldsymbol{x}$ converges with $\liminf \boldsymbol{x} = \lim \boldsymbol{x} = \limsup \boldsymbol{x}$. □

Lemma 1.3 (Squeeze Theorem)**.** *Let $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ be sequences of real numbers such that $x_n \leqslant y_n \leqslant z_n$ for all $n \in \mathbf{N}$. If $\boldsymbol{x}$ and $\boldsymbol{z}$ converge to the same limit $a \in \mathbf{R}$, then $\boldsymbol{y}$ also converges to a.*

Proof. Let $\varepsilon > 0$. Then there exist $N, M \in \mathbf{N}$ such that $|a - x_n| < \varepsilon$ for all $n \geqslant N$ and $|a - z_n| < \varepsilon$ for all $n \geqslant M$. Setting $K := \max(N, M)$, it follows that for all $n \geqslant K$ we have $a - \varepsilon < x_n \leqslant y_n \leqslant z_n < a + \varepsilon$ and hence $|a - y_n| < \varepsilon$ for all $n \geqslant K$. Conclude that $\boldsymbol{y}$ converges to a as well. □

We now wish to investigate the behaviour of the sequence of mean values $\overline{\boldsymbol{x}}$ of a sequence $\boldsymbol{x}$ in terms of its limits. Before we derive the most important results, we first note that $\overline{\boldsymbol{x}}$ is bounded from above if $\boldsymbol{x}$ is, and that $\overline{\boldsymbol{x}}$ is bounded from below if $\boldsymbol{x}$ is.

Proposition 1.4. *Consider the sequence $\boldsymbol{z} = \boldsymbol{xyyy}\cdots$ where $\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors of real numbers. Let $y := \frac{1}{|\boldsymbol{y}|}\sum_{a\in\boldsymbol{y}} a$. I.e., the number y is taken to be the* mean weight *of the vector y. Then the sequence of mean values $\overline{\boldsymbol{z}}$ of $\boldsymbol{z}$ converges to y.*

Proof. Write $L := |\boldsymbol{y}|$. We note that, for $n > |\boldsymbol{x}|$, we have

$$n\overline{z}_n = x_1 + \ldots + x_{|\boldsymbol{x}|} + \lfloor (n-|\boldsymbol{x}|)/L \rfloor (y_1 + \ldots + y_L) + (y_1 + \ldots + y_{n-|\boldsymbol{x}|-L\lfloor (n-|\boldsymbol{x}|)/L \rfloor}).$$

Note that $n - |\boldsymbol{x}| - L\lfloor \frac{n-|\boldsymbol{x}|}{L} \rfloor < L$ because $\lfloor a \rfloor > a - 1$ for every $a \in \mathbf{R}$ (here, $a = \frac{n-|\boldsymbol{x}|}{L}$). Hence, the sequences with n-th terms $\frac{1}{n}(x_1 + \ldots + x_{|\boldsymbol{x}|})$ and $\frac{1}{n}(y_1 + \ldots + y_{n-|\boldsymbol{x}|-L\lfloor (n-|\boldsymbol{x}|)/L \rfloor})$ converge to zero. Since $y_1 + \ldots + y_L = Ly$, it follows that

$$\lim \overline{\boldsymbol{z}} = \lim_{n\to\infty} \lfloor (n-|\boldsymbol{x}|)/L \rfloor (y_1 + \ldots + y_L)/n = \lim_{n\to\infty} \frac{L}{n} \lfloor (n-|\boldsymbol{x}|)/L \rfloor y$$

so we are done if we can prove that $\lim_{n\to\infty} \frac{L}{n} \lfloor \frac{n-|\boldsymbol{x}|}{L} \rfloor = 1$. Since every $a \in \mathbf{R}$ satisfies $a - 1 < \lfloor a \rfloor \leqslant a$, we have $1 - \frac{1}{n}(|\boldsymbol{x}| + L) < \frac{L}{n} \lfloor \frac{n-|\boldsymbol{x}|}{L} \rfloor \leqslant 1 - \frac{|\boldsymbol{x}|}{n}$ and the sequences formed by the first and the third numbers in the latter inequality both converge to one. By lemma 1.3, the result follows. □

We often use other basic tools from real analysis that are not defined here. For those notions, we refer to [1].
We conclude the part on real analysis by an elementary lemma.

Lemma 1.5. *Let $n \in \mathbf{N}$ and let $k \in [n, \infty)$. Then the maximum value of $\prod_{i=1}^n a_i$ when taking numbers $a_i \in [1, \infty)$ satisfying $\sum_{i=1}^n a_i = k$ is given by k^n/n^n and is attained precisely when $a_i = k/n$ for each i.*

Proof. Suppose that a legitimate choice of $a_1, \ldots, a_n$ yields maximal $\prod_{i=1}^n a_i$. Suppose furthermore that two of these n numbers are distinct: say, $a_1 \neq a_2$ without loss of generality. Then the choice of $b_1 = b_2 = (a_1 + a_2)/2$ and $b_i = a_i$ for $i \geqslant 3$ yields another choice with $\sum_{i=1}^n b_i = k$ but satisfies $\prod_{i=1}^n a_i < \prod_{i=1}^n b_i$. To show the latter, it is sufficient to show that $a_1 a_2 < b_1 b_2$, or, equivalently, that $4a_1 a_2 < (a_1 + a_2)^2$. But this is seen through

$$(a_1 + a_2)^2 = a_1^2 + 2a_1 a_2 + a_2^2 = (a_1^2 - 2a_1 a_2 + a_2^2) + 4a_1 a_2 = (a_1 - a_2)^2 + 4a_1 a_2 > 4a_1 a_2$$

as $(a_1 - a_2)^2 > 0$ because $a_1 \neq a_2$. So we have shown that $\prod_{i=1}^n a_i < \prod_{i=1}^n b_i$, which contradicts maximality of $\prod_{i=1}^n a_i$. Conclude that no pair of the numbers $a_1, \ldots, a_n$ is a pair of distinct numbers. Hence, $a_i = k/n$ for all i and the maximum value of the product we are seeking equals k^n/n^n. □

1.3 Complexity Analysis

In this thesis, the speed analyses and in particular counting the running time of elementary operations in all algorithms are based upon [6]. Since running times are given in terms of function classes consisting of real-valued functions in terms of finitely many variables $n_1, \ldots, n_k$ ranging over $\mathbf{N}$, we will formally define

the most important ones.

Let $f : \mathbf{N}^k \to \mathbf{R}_{\geqslant 0}$. We define a class of functions $O(f(n_1, \ldots, n_k))$ as follows: we say that g is in $O(f(n_1, \ldots, n_k))$, notation $g = O(f(n_1, \ldots, n_k))$, if there exist $C > 0$ and $N \in \mathbf{N}$ such that $N \leqslant \max_i n_i$ implies $g(n_1, \ldots, n_k) \leqslant Cf(n_1, \ldots, n_k)$. Similarly, we say that g is in $\Omega(f(n_1, \ldots, n_k))$ if there exist $C > 0$ and $N \in \mathbf{N}$ such that $N \leqslant \max_i n_i$ implies $g(n_1, \ldots, n_k) \geqslant Cf(n_1, \ldots, n_k)$. Finally, the class $\Theta(f(n_1, \ldots, n_k))$ is defined as the intersection of $O(f(n_1, \ldots, n_k))$ and $\Omega(f(n_1, \ldots, n_k))$: we say that $g(n_1, \ldots, n_k) = \Theta(f(n_1, \ldots, n_k))$ if and only if $g(n_1, \ldots, n_k) = O(f(n_1, \ldots, n_k))$ and $g(n_1, \ldots, n_k) = \Omega(f(n_1, \ldots, n_k))$.

For a more thorough background on complexity analysis (of algorithms), we refer to [6, Ch. 3].

1.4 Graph Theory

In our review of graph theory, we first revisit the definition of directed graphs and fom there onward we address the basic properties and invariants of directed graphs.

A *directed graph*, or *digraph* for short, is a nonempty collection of *vertices* with a collection of (*directed*) *edges* between these vertices. Formally, a digraph G is a pair $G = (V, E)$ with V a nonempty set called the *vertex set* of G, and $E \subset V \times V$ is a set constituting the *edges* of G. We say that there is an edge from $v \in V$ to $w \in V$ precisely when $vw \in E$, where vw is short-hand notation for (v, w). Given an edge vw, we call v the *tail* of vw and w the *head* of vw, and the tail and head of an edge e are denoted by $t(e)$ and $h(e)$, respectively. For a graph G with possibly unspecified vertex- and edge sets, we write $V(G)$ and $E(G)$ for these two sets, respectively. The graph G is called *finite* if V (and hence E) is (are) finite.

In all definitions that follow, let $G = (V, E)$ be a digraph.

Definition 1.6 (Subgraphs)**.** *For a nonempty vertex subset $S \subset V$, we can look at the class $G(S) := \{(S, F) : F \subset E \cap (S \times S)\}$ of graphs with vertex set S and edges of G between vertices in S. A graph in $G(S)$ for some nonempty $S \subset V$ is called a* subgraph *of G. Any $G(S)$ has a maximal element with respect to set inclusion on edge sets, namely $G[S] := (S, E \cap (S \times S))$, and the graph $G[S]$ is called the* subgraph of G induced by S.

Definition 1.7 (Graph Connectivity I: vertex neighbours)**.** *For $v, w \in V$ we call v an* in neighbour *of w and w an* out neighbour *of v if $vw \in E$. We also say that v* precedes *or is a* predecessor of *w and that w* succeeds *or is a* successor of *v. We let $N^-(v)$ be the collection of all $u \in V$ with $uv \in E$, and we let $N^+(v)$ be the collection of all $w \in V$ with $vw \in E$. Furthermore, the set cardinalities $d^-(v) := |N^-(v)|$ and $d^+(v) := |N^+(v)|$ are said to be the* in degree *and* out degree *of v, respectively. Finally, v is called a* sink *if $d^+(v) = 0$, and given a vertex subset $S \subset V$ we let $S^* := \{v \in V : d^+(v) > 0\}$ be the collection of non-sinks in V.*

Definition 1.8 (Paths in Graphs)**.** *A path in G is a vector or sequence $\boldsymbol{\sigma}$ of edges in G such that for every two consecutive edges σ_i, σ_{i+1} in $\boldsymbol{\sigma}$ the head of the first edge coincides with the tail of the second:*

$h(\sigma_i) = t(\sigma_{i+1})$. *A finite path* $\boldsymbol{\sigma}$ *is said to be* closed *if the tail of the first edge coincides with the head of the last edge:* $t(\sigma_1) = h(\sigma_{|\boldsymbol{\sigma}|})$. *A closed path* $\boldsymbol{\sigma}$ *is said to be a* cycle *if it passes through exactly* $|\boldsymbol{\sigma}|$ *distinct vertices* (*or, equivalently, if it does not strictly contain other closed paths*). *An edge* $e \in E$ *can be seen as a path* (e) *of length one, and* e *is said to be a* loop *if* (e) *is a cycle* (*or, equivalently, if* $t(e) = h(e)$). *Finally, two paths* $\boldsymbol{\sigma}, \boldsymbol{\tau}$ *can be concatenated to another path* $\boldsymbol{\sigma\tau}$, *provided* $\boldsymbol{\sigma}$ *is finite, if the head of the final edge of the first path* $\boldsymbol{\sigma}$ *coincides with the tail of the first edge of the second path* $\boldsymbol{\tau}$.

Remark 1.9. *Alternatively, paths can be characterized as a sequence of vertices. That is, instead of writing* $\boldsymbol{\sigma} = \sigma_1\sigma_2\sigma_3\cdots$ *with each* $\sigma_i \in E$ *we could write* $\boldsymbol{\sigma} = v_1v_2v_3\cdots$ *with each* $v_iv_{i+1} \in E$. *The length of* $\boldsymbol{\sigma}$ *is always taken to be the number of edges, however, which is the number of vertices minus one.*

Definition 1.10 (Graph Connectivity II: strong connectivity)**.** *Two vertices* $v, w \in V$ *are said to be* strongly connected *if* $v = w$ *or there are paths from* v *to* w *and from* w *to* v *in* G. *The binary relation of* strong connectivity *is an equivalence relation on* V, *and its corresponding equivalence classes are called the* strongly connected components *of* G. *We call* G strongly connected *if all vertices of* G *are pairwise connected, or, equivalently, if* V *itself is the only strongly connected component of* G.

Definition 1.11. *Let* $G = (V, E)$ *be a graph, and* $x \in V$. *We define* G_x *as the largest subgraph of* G *containing* x *for which there is a path in* G_x *from* v *to* x *for every* $v \in V(G_x)$.

The graph G_x can be efficiently computed as follows, where we recall that the maximum and minimum of the empty set are taken as $-\infty$ and ∞, respectively:

Algorithm 1: Computing G_x from $G = (V, E)$ and $x \in V$

Data: a graph $G = (V, E)$, $x \in V$
Result: G_x

```
1  Function graphSubx(G = (V, E), x)
2      n ← |V|
3      for v ∈ V do
4          b_0(v) ← [v = x, 0, −1]
5      end
6      for k = 1, ..., n do
7          for v ∈ V do
8              b_k(v) ← max{b_{k−1}(w) : w ∈ N^+(v)}
9          end
10     end
11     V_x ← {v ∈ V : max_{0⩽k⩽n} b_k(x) = 0}
12     return (V_x, E ∩ (V_x × V_x))
13 end
```

Remark 1.12. *We can weaken the notion of strong connectivity as follows. Two vertices* $v, w \in V$ *are said to be* (weakly) connected *if* $v = w$ *or there is a nonempty path from* v *to* w *in* G', *where* G' *is the undirected graph with vertex set* V *for which there is an edge between two vertices if and only if there is an edge from one to the other in* G (*or in both directions in* G). *The binary relation of* (*weak*) *connectivity is again an equivalence relation, and we obtain notions as* (weakly) connected components *of* G (*or simply*

components) *and graphs being* (weakly) connected (*or not*). *Note that every component of a digraph is a disjoint union of strongly connected components.*

Remark 1.13. *All digraphs on which mean payoff games are played are taken to be finite. Hence, when using the word "graph" it is implicitly understood that we take a finite directed graph.*

Finally, we define the notion of positional strategies below, as the concept of positional strategies plays a central role in reachability games and hence mean payoff games on finite digraphs.

Definition 1.14 (Positional Strategies)**.** *Let $G = (V, E)$ be a graph and let $S \subset V$ be a vertex subset. Then a map of the form $f : S^* \to V$ from the set of non-sinks in S into the vertex set of G satisfying $vf(v) \in E$ for all $v \in S^*$ is said to be a* positional strategy *on S. When $S = V$ we call f a* positional strategy*.*

Remark 1.15. *A key feature of positional strategies is that they define paths, given a start vertex $v \in V$. That is, if we pick $v \in V$ and a positional strategy $f : V^* \to V$ then a path $\boldsymbol{\sigma}(f, v) := vf(v)f^2(v)\cdots$ is formed, where for $k \in \mathbf{N}$ the map $f^k = f \circ \ldots \circ f$ is the k-th iterate of f. Note that $\boldsymbol{\sigma}(f, v)$ is infinite if $f^k(v) \in V^*$ for all $k \in \mathbf{N} \cup \{0\}$ and is finite of length* $\min\{k \in \mathbf{N} \cup \{0\} : f^k(v) \in V \backslash V^*\}$ *otherwise. We also note that $\boldsymbol{\sigma}(f, v)$ either ends in a sink or in a cycle hence either forms a vector of edges or an (eventually) periodic sequence of edges. The path $\boldsymbol{\sigma}(f, v)$ is said to be the path starting at v* following *the positional strategy f.*

1.5 Reachability

Since the concept of reachability plays an important role in mean payoff games, let us briefly define its notion by describing the winning criterion on mean payoff games in the case the play is finite. Given a graph $G = (V, E)$, we can partition V into two sets $V = V_0 \sqcup V_1$ to our liking and let two players, player 0 and player 1, play a game on G. We let $\varepsilon : V \to \{0, 1\}$ be such that $V_0 = \varepsilon^{-1}(0)$ and $V_1 = \varepsilon^{-1}(1)$. All vertices in V_0 are said to belong to player 0, and all vertices in V_1 are said to belong to player 1.

The setting of the game is as follows. We choose a start vertex $v \in V$ and let player $\varepsilon(v)$ decide the next move. That is, player $\varepsilon(v)$ picks a vertex $w \in V$ such that $vw \in E$, and next it is the turn of player $\varepsilon(w)$ to move. In this way, a path $\boldsymbol{\sigma}$ is formed starting at v and $\boldsymbol{\sigma}$ is said to be a *play* starting at v.

If during the construction of $\boldsymbol{\sigma}$ a sink $s \in V \backslash V^*$ is hit, then $\boldsymbol{\sigma}$ will be finite with end vertex s and player $\varepsilon(s)$ is said to lose the game. If no sink is hit during the construction of $\boldsymbol{\sigma}$ then $\boldsymbol{\sigma}$ is infinite. In that case, we need another criterion to decide which player wins. In the case of *parity games*, to each vertex an integer was assigned and the parity of the largest integer occurring infinitely often would determine the winner: player 0 would win if that parity is even, and player 1 would win if that parity is odd. For mean payoff games we introduce another winning criterion for plays of infinite length, and we do so in definition 2.4.

Remark 1.16. *In most literature, graphs on which reachability games are played are taken such that $V = V^*$, i.e. such that the graph in question does not have sinks. In this thesis, however, we consider any finite directed graph, possibly with sinks, as will be apparent in definition 4.1.*

The concept that is relevant for *all* two-player games on digraphs, however, is the notion of *reachability* and we define the most fundamental construct of reachability below.

Definition 1.17 (Reachability)**.** *Let (G, ε) be the data of a two-player game on a finite digraph. That is, we let $G = (V, E)$ be a finite digraph and we let $\varepsilon : V \to \{0, 1\}$ define the partition $V = \varepsilon^{-1}(0) \sqcup \varepsilon^{-1}(1)$ of V. Let $v \in V$. For $p \in \{0, 1\}$, we let $R_p(v) \subset V$ be the set of all $w \in V$ for which, if a play starts at w, player p can force the play to pass through v irrespective of the decisions of the other player along the way. It is often said that $R_p(v)$ consists of the vertices from which player p can* reach *the vertex v.*

Note that if $s \in V \backslash V^*$ is a sink then player $1 - \varepsilon(s)$ can force a win if we let a game start at any vertex in $R_{1-\varepsilon(s)}(s)$.

Remark 1.18. *We can generalize the notion of $R_p(v)$ as follows. Given $S \subset V$ and $p \in \{0, 1\}$, we let $R_p(S) = \bigcup\{R_p(v) : v \in S\}$ be the set of all $w \in V$ from which player p can reach a vertex in S.*

Algorithms running in time $O(|E|)$ are known to compute the sets of the form $R_p(S)$. We give one below, based on [2, Algorithm 5], and we implement it in Listing 1. We recall that for a propositional formula φ and objects A and B, the object $[\varphi, A, B]$ has value A if φ is true and B otherwise. We furthermore recall that for a graph $G = (V, E)$ and a vertex $v \in V$, the set $N^-(v)$ is the collection of all $u \in V$ with $uv \in E$ and the set $N^+(v)$ is the collection of all $w \in V$ with $vw \in E$. For $v \in V$ and $S \subset V$ we write $vS := \{vw : w \in S\}$ and $Sv := \{uv : u \in S\}$ which are subsets of $V \times V$. The routine `ReachOne` only finds the vertices that are within a path of length one from S, but `Reach` repeatedly utilizes `ReachOne` to determine the full set $R_p(S)$.

Algorithm 2: Computing the set from which a player can reach a given vertex subset

Data: a graph G, $\varepsilon : V(G) \to \{0, 1\}$, $S \subset V(G)$, $p \in \{0, 1\}$
Result: $R = R_p(S)$

```
1 Function Reach(G, ε, S, p)
2 |   R ← S, R′ ← S
3 |   R ← ReachOne (G, ε, R′, p)
4 |   while R′ ⊊ R do
5 |   |   R′ ← R, R ← ReachOne (G, ε, R′, p)
6 |   end
7 |   return R
8 end
```

```
1 Function ReachOne(G, ε, S, p)
2 |   T ← S
3 |   for v ∈ S do
4 |   |   for u ∈ N⁻(v)\S do
5 |   |   |   T ← [ε(u) = p or N⁺(u) ⊂ S, T ∪ {u}, T]
6 |   |   end
7 |   end
8 |   return T
9 end
```

2 Mean Payoff Games

Mean Payoff Games have been introduced by Ehrenfeucht and Mycielski in 1979 in [8]. However, only mean payoff games on bipartite graphs were studied in [8]. Many other variants and generalizations have been studied, of which [12] is an important work taking a central position in this thesis. We shall, in this thesis, only lay the focus on mean payoff games on finite directed graphs where the edges have *integer weights*.

Other notable works on mean payoff games include [4], [5], and [7].

2.1 Winning Criterion for Mean Payoff Games

Definition 2.1 (Mean Payoff Game)**.** *The data of a* mean payoff game *is as follows. We choose a triple* (G, ε, μ)*. Here* $G = (V, E)$ *is a graph, and* $\varepsilon : V \to \{\pm\}$ *is a map defining a partition of* V *into sets* $V = V_- \sqcup V_+$ *where* $V_- := \varepsilon^{-1}(-)$ *and* $V_+ := \varepsilon^{-1}(+)$*. The vertices of* V_- *are said to belong to* player P_- *or* player Min*, and the vertices of* V_+ *are said to belong to* player P_+ *or* player Max*. The function* $\mu : E \to \mathbf{Z}$ *is a* weight function *attaching to every edge of the graph* G *an integer* weight*.*

Remark 2.2. *In equations, for convenience we often identify* $+$ *with* 1 *and* $-$ *with* -1*.*

The setting of a mean payoff game (G, ε, μ) is partially inherited by the reachability game (G, ε) for two players. This means that if a play ends in a sink, the player who does *not* own that sink wins. We still have not decided yet who wins in case a path of infinite length is formed, but we do so in terms of the weight function μ and with that we automatically introduce the concept of *mean payoff*.

Definition 2.3 (Mean Payoff)**.** *Suppose that* $\boldsymbol{\sigma}$ *is an infinite play on a mean payoff game* (G, ε, μ)*. This gives rise to a sequence of integers* $\mu(\boldsymbol{\sigma}) = (\mu(\sigma_1), \mu(\sigma_2), \mu(\sigma_3), \ldots)$*. In particular,* $\mu(\boldsymbol{\sigma})$ *has an associated sequence of mean values* $\overline{\mu(\boldsymbol{\sigma})}$ *whose* n*-th term is given by* $\overline{\mu(\boldsymbol{\sigma})}_n = \frac{1}{n}\sum_{i=1}^{n} \mu(\sigma_i)$*. Then we can define two* characteristic values $\chi_-(\boldsymbol{\sigma})$ *and* $\chi_+(\boldsymbol{\sigma})$ *of the play* $\boldsymbol{\sigma}$ *by taking*

$$\chi_-(\boldsymbol{\sigma}) := \liminf \overline{\mu(\boldsymbol{\sigma})} \quad \textit{and} \quad \chi_+(\boldsymbol{\sigma}) := \limsup \overline{\mu(\boldsymbol{\sigma})}.$$

The values $\chi_-(\boldsymbol{\sigma})$ and $\chi_+(\boldsymbol{\sigma})$ are guaranteed to exist, because both are limits of monotone sequences that are bounded (bounded from above by $\|\mu\|_\infty$ and from below by $-\|\mu\|_\infty$ if we regard μ as a vector indexed by the finite set E with values in $\mathbf{R}$). Note, furthermore, that $\chi_-(\boldsymbol{\sigma}) \leqslant \chi_+(\boldsymbol{\sigma})$.

Definition 2.4 (Winning Criterion for Infinite Plays)**.** *In the context of Definition 2.1, an infinite play* $\boldsymbol{\sigma}$ *is said to be* won *by player Min if* $\chi_+(\boldsymbol{\sigma}) < 0$*, is said to be* won *by player Max if* $\chi_-(\boldsymbol{\sigma}) > 0$*, and is said to be a* draw *otherwise (i.e., if* $\chi_-(\boldsymbol{\sigma}) \leqslant 0 \leqslant \chi_+(\boldsymbol{\sigma})$*).*

2.2 Positional Strategies

In this thesis, we thoroughly study mean payoff games with the additional requirement that each game is played along positional strategies. That is, player Min chooses a *positional strategy* $f_- : V_-^* \to V$ and player Max chooses a *positional strategy* $f_+ : V_+^* \to V$. Then choosing a start vertex $v \in V$ and

forming the function $f := f_- \sqcup f_+ : V^* \to V$ we obtain the play $\boldsymbol{\sigma}(f, v)$ where $\boldsymbol{\sigma}(f, v)$ is defined in Remark 1.15. It has been the subject of many studies to construct efficient algorithms to find *optimal* positional strategies for both players, as outlined in [12]. The reason lies in the fact that it was shown in [8] that positional strategies f_-, f_+ exist, independent of starting vertex, such that for any $v \in V$ we have $\chi_-(\boldsymbol{\sigma}(f, v)) \leqslant \chi(v) \leqslant \chi_+(\boldsymbol{\sigma}(f, v))$. In the argumentation in [8], each mean payoff game (G, ε, μ) was assumed to be bipartite with respect to the partition $V = V_- \sqcup V_+$. The authors of [12], however, pointed out that the additional requirement that the mean payoff game in question is bipartite is not necessary to derive these results.

For completeness, we define the subclass of bipartite mean payoff games.

Definition 2.5. *A mean payoff game (G, ε, μ) is said to be* bipartite *if the graph on which the game is played is bipartite with respect to the partition $V = V_- \sqcup V_+$: if $E \cap ((V_- \times V_+) \cup (V_+ \times V_-)) = \varnothing$.*

Next, we outline in more detail how values of a mean payoff game can always be achieved using positional strategies. Ehrenfeucht and Mycielski showed the following in [8, Thm. 1]. Choose a (bipartite) mean payoff game (G, ε, μ), and a starting vertex $v \in V$. Then there exists a value $\chi(v)$ such that:

(−) Min has an *optimal* positional strategy f_- ensuring that any path $\boldsymbol{\sigma}$ start in v and following f_- has the property that $\chi_-(\boldsymbol{\sigma}) \leqslant \chi(v)$ independent of Max's choices of moves (Max may play via a positional strategy or not), and

(+) Max has an *optimal* positional strategy f_+ ensuring that any path $\boldsymbol{\sigma}$ starting in v and following f_+ has the property that $\chi_+(\boldsymbol{\sigma}) \geqslant \chi(v)$ independent of Min's choices of moves.

Next, since we know that infinite paths following positional strategies are eventually periodic (as outlined in Remark 1.15), it must be that the sequence of mean values $\overline{\mu(\boldsymbol{\sigma}(f, v))}$ is eventually periodic as well if f_- and f_+ are positional strategies for Min and Max, respectively. It follows, upon combining Propositions 1.2 and 1.4, that $\chi_-(\boldsymbol{\sigma}) = \chi_+(\boldsymbol{\sigma})$ for all paths $\boldsymbol{\sigma}$ following positional strategies. Hence, given a start vertex $v \in V$, we have $\chi(v) = \chi_-(\boldsymbol{\sigma}(f, v)) = \chi_+(\boldsymbol{\sigma}(f, v))$ (whenever f_-, f_+ are optimal) and $\chi(v)$ either equals the mean weight[4] $m(C)$ of the terminal cycle C a play following optimal positional strategies will end up in or equals $-\varepsilon(s) \cdot \infty$ if $\boldsymbol{\sigma}(f, v)$ ends in a sink $s \in V \backslash V^*$.

Remark 2.6. *At this point it deserves to be noted that we have been taking values $\chi_-(\boldsymbol{\sigma}), \chi_+(\boldsymbol{\sigma})$ of infinite paths $\boldsymbol{\sigma}$. This is because in [8] and [12] only graphs without sinks are considered. We, however, do not exclude the possible occurence of sinks. If a finite play $\boldsymbol{\sigma}$ ends in a sink belonging to Min, then we consider this play optimal for Max and we set $\chi(\boldsymbol{\sigma}) = +\infty$. Likewise, if a finite play $\boldsymbol{\sigma}$ ends in a sink belonging to Max, then we set $\chi(\boldsymbol{\sigma}) = -\infty$. In this way, the results of Ehrenfeucht and Mycielski as outlined in [8] do not only extend to non-bipartite mean payoff games but also to mean payoff games with sinks.*

It deserves to be noted that χ can be thought of as a function $V \to \overline{\mathbf{R}} := \mathbf{R} \cup \{\pm\infty\}$. On a graph without sinks we obtain a function $\chi : V \to \mathbf{R}$. Given $v \in V$, we call $\chi(v)$ the *characteristic value* of v. The information of the function χ can be reduced to a function $\gamma : V \xrightarrow{\chi} \overline{\mathbf{R}} \xrightarrow{\text{sign}} \{-, 0, +\}$ which assigns to

[4] The mean weight of a cycle C in a weighted digraph is taken as the sum of the weights of the edges of C altogether divided by the length of C

each vertex v the sign of $\chi(v) \in \overline{\mathbf{R}}$. The function γ is said to be the *decision function* associated with the mean payoff game (G, ε, μ).

2.3 The Three Main Problems on Mean Payoff Games

In essence, the decision problem of mean payoff games is a computational problem and is not known to be polynomial [12]. We also consider two seemingly harder problems: computing characteristic values and finding optimal positional strategies. The first problem is known to be in $\mathrm{NP} \cap \mathrm{co}-\mathrm{NP}$ [12, Thm. 4.2] hence is a candidate to break the conjecture that the complexity classes P and NP coincide.

Definition 2.7 (Decision Problem of Mean Payoff Games)**.** *Let (G, ε, μ) be the data of a mean payoff game. The* decision problem *on (G, ε, μ) is the computational task to, for each $v \in V$, determine the value $\gamma(v) := \mathrm{sign}(\chi(v))$. That is, for each $v \in V$ it is required to determine whether Min wins at v ($\gamma(v) = -$), whether Max wins at v ($\gamma(v) = +$), or v is a* draw position *($\gamma(v) = 0$).*

In other words, given a mean payoff game, we assume that the players play according to optimal positional strategies and we wish to determine which starting vertices fall into the set $\gamma^{-1}(-)$ of *winning vertices for Min*, which fall into the set $\gamma^{-1}(+)$ of *winning vertices for Max*, and which fall into the set $\gamma^{-1}(0)$ of *draw positions*. The process of determining these winning positions is called *solving the mean payoff game*. Once again, we stress that the mean payoff game in question does not have to be bipartite with respect to the partition $V = V_- \sqcup V_+$ and additionally is allowed to contain sinks.

Remark 2.8. *To say that a player wins at a vertex $v \in V$ might be confusing at first hand in the following sense. Namely, an optimal play starting at v does not necessarily end in v or end up in a cycle containing v. All it means for a player to win at a vertex v or to draw against the other player is that the play* starting *at v will be winning for a player or will lead to a draw.*

Remark 2.9. *Three fundamental problems on mean payoff games have been studied:*

(1) *the* decision problem *on mean payoff games (definition 2.7);*

(2) *the computational problem of* finding values $\chi(v)$*;*

(3) *the computational problem of* finding optimal strategies f_-, f_+.

A solution to (3) *would immediately be a solution to problem* (2) *as argued in Corollary 4.6. Additionally, a solution to problem* (2) *would immediately be a solution to problem* (1)*: given that $v \in V$ has value $\chi(v)$, we at once decide who wins at v by taking $\gamma(v) := \mathrm{sign}(\chi(v))$. If follows that problems* (1)*,* (2) *and* (3) *are successive generalizations. As outlined in [12], it is an open problem whether problem* (3) *is deducible from problem* (2).

2.4 Two Classical Approaches

Since the first complexity classification of the problem of mean payoff games in 1996 by Zwick and Paterson [12], three parameters have been of highest priority in determining the speed of (decision) algorithms on mean payoff games: the numbers n, M, and $|E|$. Here, $n := |V|$ is reserved to denote the number of vertices of the graph $G = (V, E)$ the game is played on, M is the *size* of the weight function $\mu : E \to \mathbf{Z}$

defined by $M := \|\mu\|_\infty = \max_{e\in E} |\mu(e)|$ and $|E|$ is the number of edges of the graph G. In [12] two algorithms were presented to solve mean payoff games, be it that they do not take into account the existence of sinks. One runs in $O(n^3 \cdot |E| \cdot M)$ time, and one runs in $n \cdot 2^{O(|E|)}$ time or in $2^{O(|E|)}$ time if we store positional strategies efficiently. Both can be generalized to mean payoff games in which sinks may occur. In fact, the first algorithm was exhibited in [2] and generalized there. We start by exhibiting the *second* algorithm, as it is more simplistic in nature whence it will be able to serve as a proper algorithmic introduction.

2.4.1 Approach I: $n \cdot 2^{O(|E|)}$ time

As usual, let (G, ε, μ) be the data of a mean payoff game. The observation which led to the algorithm to solve (2) by solving (3) we present here has been the following. As outlined in subsection 2.2, both players have optimal positional strategies at their disposal independent of starting vertex. This observation, together with [9, Equation (1)] implies that the collection of optimal strategies $f = f_- \sqcup f_+$ is precisely given by all pairs f_-, f_+ such that $\chi(v) = \chi(\boldsymbol{\sigma}(f, v))$ and furthermore that whenever f_-, f_+ achieve the optimizations

$$\chi(v) = \max_{g_+} \min_{g_-} \chi(\boldsymbol{\sigma}(g, v)) = \min_{g_-} \max_{g_+} \chi(\boldsymbol{\sigma}(g, v)) \tag{2.1}$$

for all $v \in V$ (where we use the notation $g = g_- \sqcup g_+$), both strategies f_-, f_+ are optimal. In principle, (2.1) can already be seen as an algorithm on itself to find the characteristic values — we go over all positional strategies g_-, g_+ and for each one we determine the value $\chi(\boldsymbol{\sigma}(g, v))$. We, however, wish to present one on a lower level, in such a way that the running time will become apparent. We give a recursive variation, with the urgent disclaimer that V needs to be ordered in such a way that the signs of the instances a vertex is taken on line 15 do not alternate; i.e., when listing these signs as the algorithm progresses, one should obtain a list of the form $p, \ldots, p, -p, \ldots, -p$ where $p \in \{\pm\}$.

Algorithm 3: Solving a mean payoff game by exhausting positional strategies

Data: a graph G, $\varepsilon : V(G) \to \{\pm\}$, $\mu : E(G) \to \mathbf{R}$
Result: $\gamma : V \to \{-, 0, +\}$

```
1  Function solveFromPosStr(G, ε, μ)
2      (V, E) ← G
3      for v ∈ V do
4          γ(v) ← sign(solveVertex(G, ε, μ, v))
5      end
6      return γ
7  end
8  Function solveVertex(G, ε, μ, v)
9      (V, E) ← G
10     V* ← {v ∈ V : d⁺(v) ⩾ 1}
11     V₂ ← {v ∈ V : d⁺(v) ⩾ 2}
12     if |V₂| = 0 then
13         χ ← χ(σ(V* ∋ v* ↦ w ∈! N⁺(v*), v))
14     else
15         Take v₂ ∈ V₂
16         for w ∈ N⁺(v₂) do
17             E_w ← E\(v₂N⁺(v₂)\{v₂w})
18         end
19         if ε(v₂) = 1 then
20             χ ← max{solveVertex((V, E_w), ε, μ|_{E_w}, v) : w ∈ N⁺(v₂)}
21         else
22             χ ← min{solveVertex((V, E_w), ε, μ|_{E_w}, v) : w ∈ N⁺(v₂)}
23         end
24     end
25     return χ
26 end
```

Here, on line 13 there is only one positional strategy $V^* \to V$ as no vertex with respect to the inputted graph on line 8 has multiple out neighbours once we are past the if-condition on line 12. The path is uniquely determined by the graph G and start vertex v. On line 17, once an out neighbour w of v_2 is selected, we obtain E_w by removing all edges from E with tail v_2 except for the edge v_2w.

The worst-case running time of Algorithm 3 is $n \cdot 2^{O(|E|)}$. This is seen as line 4 calls n times the exact same number of positional strategies, of which there are $\prod_{v \in V} \max(1, d^+(v))$. The fact that the running time of Algorithm 3 is in $n \cdot 2^{O(|E|)}$ is established once we verify that $\sum_{v \in V} \log(\max(1, d^+(v))) = O(|E|)$,

which is seen through

$$\begin{aligned}\frac{1}{|E|}\sum_{v\in V}\log\big(\max(1,d^+(v))\big) &= \frac{1}{|E|}\sum_{v\in V^*}\log\big(d^+(v)\big)\\ &= \frac{1}{|E|}\log\left(\prod_{v\in V^*}d^+(v)\right)\\ &\leqslant \frac{1}{|E|}\log\left(\left(\frac{\sum_{v\in V^*}d^+(v)}{|V^*|}\right)^{|V^*|}\right) \quad \text{(by Lemma 1.5)}\\ &= \frac{1}{|E|}\log\left(\left(\frac{|E|}{|V^*|}\right)^{|V^*|}\right)\\ &= \frac{\log\left(\frac{|E|}{|V^*|}\right)}{\left(\frac{|E|}{|V^*|}\right)}\end{aligned}$$

which has an absolute upper bound given by the global maximum[5] of the positive-valued function $x \mapsto \frac{1}{x}\log x$ on $(1,\infty)$, which is applicable as $|E| = \sum_{v\in V^*} d^+(v) \geqslant \sum_{v\in V^*} 1 = |V^*|$. We can also show that $n\cdot 2^{O(|E|)}$ is the sharpest upper running time bound of Algorithm 3. We take the following family of graphs $G = (V_n, E_n)$ indexed by and given in terms of increasing $n \in \mathbf{N}$: let $V_n = \mathbf{Z}/n\mathbf{Z}$ and $E_n = \bigcup_{v\in V_n}\{(v, v+1),(v, v+2)\}$. Then $d^+(v) = 2$ for every $v \in V_n$ and $|E_n| = 2n$, so $\log\left(\prod_{v\in V}\max(1,d^+(v))\right) = n = \Theta(2n) = \Theta(|E_n|)$.

Remark 2.10. *Algorithm 3 could be made significantly faster by the following observation, which has not been pointed out in the original paper [12]. Namely, it was established in [8] that positional strategies exist that simultaneously serve as optimal for every start vertex. This implies in particular that, once the value $\chi(v)$ of a first vertex $v \in V$ is found, we can keep track of all positional strategies f for which $\chi(\boldsymbol{\sigma}(f, v)) \neq \chi(v)$ and exclude those when we take another vertex $v' \in V\backslash\{v\}$ to determine the value $\chi(v')$ of.*

2.4.2 Approach II: $O(n^3\cdot|E|\cdot M)$ time

It has been the same mathematicians Zwick and Paterson in the very same paper [12] who managed to construct a mean payoff game solving algorithm[6] whose running time only depends polynomially on n, but does linearly depend on M: their Algorithm [12, Thm. 2.3] runs in $O(n^3\cdot|E|\cdot M)$ time. Since $|E| = O(n^2)$ we are guaranteed that [12, Thm. 2.3] runs in $O(n^5M)$ time which has polynomial dependence on n but has linear dependence on M — or, exponential dependence on $\log M$. The algorithm has already been implemented at lower level in [2, Algorithm 4] which is in addition able to solve on (smaller) strongly connected components of the graph. Here, we give a simplification of [2, Algorithm 4] as to reflect [12, Thm. 2.3] in its purest and most classical form.
Algorithm 4 utilizes the fact that for each $v \in V$ we have $\lim_{k\to\infty}\eta_k(v)/k = \chi(v)$, as will be apparent in Theorem 2.14. We prove correctness of Algorithm 4 in Proposition 2.15.

Remark 2.11. *Essentially, Algorithm 4 runs in $O(n^3\cdot|E|\cdot M)$ time, but the decision problem whether $\chi(v) > 0$, $\chi(v) = 0$, or $\chi(v) < 0$ for all vertices $v \in V$ can be solved in $O(n^2\cdot|E|\cdot M)$ [12, Thm. 2.4].*

[5] One computes that this maximum is $1/(e\cdot\ln 2) = 0.5307\ldots$
[6] As to solve problem *(2)*

Algorithm 4: Solving a mean payoff game by computing finite values

Data: a graph G, $\varepsilon : V(G) \to \{\pm\}$, $\mu : E(G) \to \mathbf{Z}$
Result: $\gamma : V \to \{-, 0, +\}$

```
Function solveFromFinVals(G, ε, μ)
    (V, E) ← G
    n ← |V|, M ← max_{e∈E} |μ(e)|
    for v ∈ V do
        η_0(v) ← 0, γ(v) ← 0
    end
    for k = 1, ..., 4n²M + 1 do
        for v ∈ V do
            if ε(v) = 1 then
                η_k(v) ← max{μ(v, w) + η_{k−1}(w) : w ∈ N⁺(v)}
            else
                η_k(v) ← min{μ(v, w) + η_{k−1}(w) : w ∈ N⁺(v)}
            end
            γ(v) ← [η_k(v) > 2nM, 1, [η_k(v) < −2nM, −1, γ(v)]]
        end
    end
    return γ
end
```

This slight polynomial improvement in n has not been incorporated in Algorithm 4. This is because our greatest concern with Algorithm 4 is the linear dependence on M.

Remark 2.12. *Algorithm 4 is limited to graphs without sinks, since it was assumed in [12, p. 1] that every vertex has at least one out neighbour. In case sinks are present, we initialize each value $\eta_0(v)$ as zero if v is not a sink and as $-\varepsilon(v) \cdot \infty$ if v is a sink. Furthermore, we replace E by $V \times V$ and extend μ to $V \times V$ by $\mu(v, w) = -\varepsilon(v) \cdot \infty$ for each $vw \in V \times V$ that was originally not an edge.*

2.5 Finite Values

The values $\eta_k(v)$ as computed in Algorithm 4 can appear in a broader context than just this algorithm, as they are solely induced by the mean payoff game data (G, ε, μ). For $v \in V$ and $k \in \mathbf{N}$ the number $\eta_k(v)/k$ is said to be the *finite value of (G, ε, μ) at vertex v for plays of length k*. This nomenclature is justified by the following: if a play starts at a vertex but is beforehand given to end whenever it has reached length k instead of being infinite, then the average value of the k weights encountered when an optimal path is formed is equal to $\eta_k(v)/k$. This fact has already been established in [12], but easily extends to mean payoff games on graphs with sinks. In the same paper, it is shown that $\chi(v) = \lim_{k\to\infty} \eta_k(v)/k$, but $\eta_k(v)/k$ oftentimes approaches $\chi(v)$ at a disadvantageous rate. Due to the importance of finite values we redefine them, along with stating a fundamental result on finite values.

Definition 2.13. *Let (G, ε, μ) be the data of a mean payoff game, with $G = (V, E)$. For all $v \in V$ and $k \in \mathbf{Z}_{\geqslant 0}$, we define a value $\eta_k(v)$ as follows. We set $\eta_0(v) = 0$ for all $v \in V$. Then for $k = 1, 2, 3, \ldots$ we*

inductively set

$$\eta_k(v) := \begin{cases} \max\{\mu(v,w) + \eta_{k-1}(w) : w \in N^+(v)\}, & \text{if } \varepsilon(v) = +, \\ \min\{\mu(v,w) + \eta_{k-1}(w) : w \in N^+(v)\}, & \text{if } \varepsilon(v) = -, \end{cases} \tag{2.2}$$

with the conventions $\min \varnothing = +\infty$ *and* $\max \varnothing = -\infty$.

Next, we restate a fundamental result on finite values.

Theorem 2.14. *For $v \in V$ and $k \in \mathbf{N}$, we have the following estimate for $\chi(v)$ around $\eta_k(v)/k$:*

$$\left|\chi(v) - \frac{\eta_k(v)}{k}\right| \leqslant \frac{2nM}{k}. \tag{2.3}$$

Proof. This is [12, Thm. 2.2]. □

The fact that $\lim_{k\to\infty} \eta_k(v)/k = \chi(v)$ is an immediate consequence of Theorem 2.14.

Proposition 2.15. *Algorithm 4 terminates and outputs the correct $\gamma : V \to \{-, 0, +\}$.*

Proof. The fact that algoritm 4 terminates is immediate from the facts that $|V|$ is a finite set and the number $4n^2M+1$ is finite. Now, considering line 14 in Algorithm 4 we readily see that if $\eta_k(v) > 2nM$ then $\eta_k(v)/k > 2nM/k$ so that (2.3) implies that $\chi(v) > 0$, and if $\eta_k(v) < -2nM$ then $\eta_k(v)/k < -2nM/k$ so that (2.3) implies that $\chi(v) < 0$. If $\chi(v) = 0$ then (2.3) implies that $|\eta_k(v)/k| \leqslant 2nM/k$ for all $k \in \mathbf{N}$ and hence $\gamma(v)$ will never be assigned a nonzero value on line 14. We are left to show that the sign of a nonzero $\chi(v)$ will never be falsely detected as zero or as $-\text{sign}(\chi(v))$. First, suppose that we have a vertex $v \in V$ for which $\chi(v) > 0$ and such that $\chi(v)$ is not detected as positive in Algorithm 4. Then for all $k = 1, \ldots, 4n^2M + 1$ we have $\eta_k(v)/k \leqslant 2nM/k$. In particular, this holds if we take $k = 4n^2M + 1$. For this k we have $k > 4n^2M$ and hence $\eta_k(v)/k \leqslant 2nM/k < 2nM/(4n^2M) = 1/(2n)$. However, $2nM/k < 1/(2n)$ and (2.3) imply for this k that $|\chi(v) - \eta_k(v)/k| < 1/(2n)$ as well. Invoking the triangle inequality yields

$$|\chi(v)| \leqslant \left|\chi(v) - \frac{\eta_k(v)}{k}\right| + \left|\frac{\eta_k(v)}{k}\right| < \frac{1}{2n} + \frac{1}{2n} = \frac{1}{n}$$

and hence, as $\chi(v)$ was assumed positive, we have $0 < \chi(v) < 1/n$. In particular, $\chi(v)$ is finite and hence equals the mean weight of a cycle of length l with $1 \leqslant l \leqslant n$. But since all edges have integer weights, the denominator of the fraction $\chi(v)$ in lowest terms is a positive integer not bigger than n. Since $\chi(v)$ is positive, it must be that $\chi(v) \geqslant 1/n$ which contradicts the relation $0 < \chi(v) < 1/n$ we have derived earlier. Similarly, the sign of no negative characteristic value will be falsely detected as zero or as $+$, and the proof is complete. □

Theorem 2.16. *Algorithm 4 successfully terminates if k runs from 1 to $\lfloor 4nM/R \rfloor + 1$ where R is the minimum among the absolute values of the mean weights of all directed cycles that have nonzero mean weight.*

Proof. Whenever the main loop of Algorithm 4 runs from $k = 1$ to a value of k bigger than $4nM/R$, then for this k we would have $2nM/k < R/2$ so that a nonzero $\chi(v)$ falsely detected as zero or of the opposite sign would satisfy

$$|\chi(v)| \leqslant \left|\chi(v) - \frac{\eta_k(v)}{k}\right| + \left|\frac{\eta_k(v)}{k}\right| < \frac{R}{2} + \frac{R}{2} = R,$$

which is impossible because if $\chi(v) \in \mathbf{R}$ is nonzero then $\chi(v) = m(C)$ for some cycle C of nonzero mean weight which by definition of R satisfies $|m(C)| \geqslant R$. □

Now that we have introduced mean payoff games along with the two first significant contributions in solving them, we introduce a tool to efficiently compute the minimum cycle mean of a weighted directed graph in $O(n \cdot |E|)$ time. This algorithm has been of major importance in the contributions to the problem in [2], but was initially used in Zwick's and Paterson's paper [12] in 1996 to prove that the decision problem of mean payoff games lies in both NP and in co $-$ NP. The first method to efficiently (polynomial time in n) to compute the minimum cycle mean of a weighted digraph was presented by Richard Karp in [10] in 1978 — just one year before Ehrenfeucht's and Mycielski's introduction of mean payoff games in 1979 — and is often referred to as *Karp's algorithm*. We introduce their methods in Section 3.

3 Karp's Algorithm

Before we state a proper version of Karp's Algorithm [10], we first formulate the minimum cycle mean problem on weighted directed graphs along with two generalizations. One of those generalizations will be responsible for a big class of mean payoff games of which the decision problem can be solved in purely polynomial time. At the end we state Karp's algorithm and we mention why problem (1) resides in $\mathrm{NP} \cap \mathrm{co} - \mathrm{NP}$.

3.1 Three Problems on Cycle Means

Given a weighted digraph $(G = (V, E), \mu : E \to \mathbf{R})$ and a cycle C in G, we again write $m(C)$ for the mean weight of the cycle C. That is, $m(C)$ is the sum of the weights given by the weight function μ, altogether divided by the length of the cycle C. Occasionally, we use the term *cycle mean* of a cycle to refer to its mean weight. We are in place to pose three computational problems on mean weights of cycles in weighted digraphs. Similarly to the three problems as enumerated in Remark 2.9 we can order these three problems with respect to successive generality, and we do so before formally defining them: on weighted digraphs, we consider the problems of

(C1) finding the minimum or maximum cycle mean;

(C2) finding the minimum among the absolute values of the nonzero cycle means;

(C3) finding the minimum positive cycle mean or the maximum negative cycle mean.

Definition 3.1 (Minimum Cycle Mean Problem)**.** *The* minimum cycle mean problem *on* (G, μ) *is the computational problem to determine the associated* minimum cycle mean $\min_C m(C)$ *where* C *runs over all directed cycles in* G*.*

Remark 3.2. *Completely analogous to the problem statement in definition 3.1, one can compute the* maximum cycle mean $\max_C m(C)$ *associated to* (G, μ)*. Both problems have exactly the same computational complexity, as either is immediately derived from the other by additively inverting the weight function and switching from minimizing to maximizing or the other way around. For example, the maximum cycle mean of* (G, μ) *would be the additive inverse of the minimum cycle mean of* $(G, -\mu)$*.*

Now that we have defined problem (C1), we can formulate problem (C2)

Definition 3.3 (Minimum Absolute Nonzero Cycle Mean Problem)**.** *The* minimum absolute nonzero cycle mean problem *on* (G, μ) *is the computational problem to determine the associated* minimum absolute nonzero cycle mean $R := \min\{|m(C)| : m(C) \neq 0\}$ *where* C *runs over all directed cycles in* G *with nonzero mean weight.*

The relevance of (C2), namely that of computing the value R as defined in Definition 3.3, is immediate from Theorem 2.16.

Remark 3.4. *The fact that a solution to problem* (C2) *is also a solution to problem* (C1) *is seen as follows. Along with* (G, μ)*, consider the weighted digraph* $(G, \mu + \delta)$ *obtained from* (G, μ) *by shifting the weight function* μ *by the constant* $\delta := 1 - \min_{e \in E} \mu(e)$*. Then* $\mu + \delta : V \to [1, \infty)$ *so the minimum absolute nonzero cycle mean* R *of* $(G, \mu + \delta)$ *coincides with the minimum cycle mean* m' *of* $(G, \mu + \delta)$*. But*

shifting a weight function by a constant will shift the mean weights of all cycles by that same constant, so the minimum cycle mean m of (G, μ) is related to m' and R by $m = m' - \delta = R - \delta$.

Next, we formulate problem (C3).

Definition 3.5 (Minimum Positive Cycle Mean Problem)**.** *The* minimum positive cycle mean problem *on* (G, μ) *is the computational problem to determine the associated* minimum positive cycle mean $\min\{m(C) : m(C) > 0\}$ *where C runs over all directed cycles in G with positive mean weight.*

Remark 3.6. *Similar to the fact that the problem statement in definition 3.1 has an additive dual version as pointed out in Remark 3.2, so has the problem statement in definition 3.5. This dual version would be to compute the* maximum negative cycle mean $\max\{m(C) : m(C) < 0\}$ *associated to* (G, μ)*. One sees that the minimum positive cycle mean of (G, μ) equals the additive inverse of the maximum negative cycle mean of $(G, -\mu)$, and that the maximum negative cycle mean of (G, μ) equals the additive inverse of the minimum positive cycle mean of $(G, -\mu)$.*

At this point, we can show how problem (C3) is a generalization of problem (C2) We observe that on a weighted digraph (G, μ), we have the relation

$$\min_{C:m(C)\neq 0} |m(C)| = \min\left\{ \min_{C:m(C)>0} m(C), -\max_{C:m(C)<0} m(C) \right\}.$$

It follows that the left-hand side, which is the quantity we look for in (C2), is deduced at once from two quantities on the right-hand side and those in turn are computed by a solution to (C3)

Until now, no efficient methods to solve problems (C2) and (C3) are known. The first problem, (C1), however, does admit a known polynomial time solution which was proposed by Richard Karp in 1978 in [10]. It turns out that a polynomial time solution to problem (C1) is sufficent to determine that (1) lies fully in $\mathrm{NP} \cap \mathrm{co} - \mathrm{NP}$ [12].

3.2 Karp's Algorithm

In 1978, Richard Karp proposed an efficient method in [10] to compute the minimum cycle mean of a weighted digraph (G, μ). We state his algorithm in its most classical form. We note however that Karp assumes in [10] each graph to be strongly connected. However, Karp also notes in [10] that both determining whether a graph is strongly connected and determining the (multiple) strongly connected components of a graph in case the graph is not strongly connected can be done in $O(n+|E|)$ steps. Then determining the minimum cycle mean would be reduced to determining the minimum cycle means of the individual strongly connected components and then taking the minimum of those. This stems from the graph-theoretic fact that cycles are always fully contained within one strongly connected component.

Karp's proves in [10] that the minimum cycle mean of a weighted graph (G, μ) is given by

$$\min_{v\in V} \max_{0\leqslant k<n} \frac{F_n(v) - F_k(v)}{n-k} \tag{3.1}$$

where the quantities $F_k(v)$ for $k = 0, \ldots, n$ and $v \in V$ are defined through $F_0(s) = 0$ and $F_0(v) \neq 0$ for all $v \neq s$, and for $k \geqslant 1$ through $F_k(v) = \min_{u \in N^-(v)} F_{k-1}(u) + \mu(u, v)$ where s is any start vertex whose exact choice is irrelevant for the correctness of (3.1).

Algorithm 5: Computing the minimum cycle mean of a strongly connected digraph

Data: a strongly connected graph $G = (V, E)$, $\mu : E \to \mathbf{R}$, a vertex $a \in V$
Result: the minimum cycle mean m of (G, μ)

```
1  Function Karp(G, μ, a)
2      n ← |V|
3      for v ∈ V do
4          F_0(v) ← [v = a, 0, ∞]
5      end
6      for k = 1, ..., n do
7          for v ∈ V do
8              F_k(v) ← min{F_{k−1}(u) + μ(u, v) : u ∈ N^−(v)}
9          end
10     end
11     for k = 0, ..., n − 1 do
12         for v ∈ V do
13             m_k(v) ← (F_n(v) − F_k(v))/(n − k)
14         end
15     end
16     for v ∈ V do
17         m(v) ← max{m_k(v) : k = 0, ..., n − 1}
18     end
19     m ← min{m(v) : v ∈ V}
20     return m
21 end
```

Karp argued in [10] that his algorithm determines the minimum cycle mean independent of the start vertex $a \in V$. The primary reason why Algorithm 5 is this efficient is the fact that, given any two vertices $v, w \in V$, any path $\boldsymbol{\sigma}$ with mean weight $m(\boldsymbol{\sigma})$ strictly bigger than the minimum mean weight of a path from v to w does not have to be considered. This guarantees that problem (C1) is solved efficiently by Karp's algorithm. For problem (C2), for example, we cannot leave paths that are minimal in weight out of consideration: a cycle C attaining the minimum absolute nonzero cycle mean R may contain paths $\boldsymbol{\sigma}$ that are not minimal over the collection of paths between its endpoints.

A crucial use of Karp's algorithm is notable in the argumentation of [12, Thm. 4.2] as to why the decision problem on mean payoff games resides in $\mathrm{NP} \cap \mathrm{co} - \mathrm{NP}$.

4 Complexity Reduction

One of the most natural ways to come up with faster methods than those exhibited in 2.4 is to think of methods that reduce the complexity[7] of the mean payoff game in question yet keeping the signs of the characteristic values and with that the winning positions invariant. An example would be to double all the edge weights: with this operation, all characteristic values will double as well but their signs will not change. From the perspective of Algorithm 4, however, this would be slowing down the running time as it would double the worst case running time of Algorithm 4.

In the ideal case, one would reduce the size $M = \max_{e \in E} |\mu(e)|$ of the weight function $\mu : E \to \mathbf{Z}$ of the graph $G = (V, E)$ in question. Since this problem is very hard without edge deletion (e.g., the integer weights might be pairwise coprime), from the perspective of Algorithm 3 it would be natural to think of ways to delete edges whilst avoiding the effect that $\gamma : V \to \{-, 0, +\}$ changes. We present two different types of edges that can be omitted from the graph without changing the winning positions; one type is due to [2], and one type is introduced in this document which has been implicitly understood in previous work but was nowhere formally stated or proven. Again, $n = |V|$ is the number of vertices.

4.1 Edge Deletion

Given a mean payoff game, we look for methods to reduce the edge set of the graph to a smaller set such that the decision function of the new mean payoff game coincides with that of the first mean payoff game. In this part, we consider two types $D \subset E$ and $S \subset E$ of edges that can all be removed whilst not changing the decision function. The first type, D, is due to [2]. The second type, S, is due to previous work.

Definition 4.1 (Dispensable Edges [2])**.** *Let $k \in \mathbf{N}$. Then an edge $vw \in E$ played by player $p = \varepsilon(v)$ is said to be k-*dispensable *in determining the decision function $\gamma : V \to \{-, 0, +\}$ if one or both of the following hold:*

1. *player $-p$ can, starting from w, reach a sink $s \in V_p \backslash V_p^*$ such that every path $\boldsymbol{\sigma}$ from w to s over which player $-p$ reaches s from w has length at most $k - 2$,* or
2. *player $-p$ can reach v from w such that every path $\boldsymbol{\sigma}$ from w to v over which player $-p$ reaches v from w has length at most $k - 1$ and satisfies the property that the mean weight of the formed closed path $(vw)\boldsymbol{\sigma}$ has sign $-p$.*

Furthermore, vw is called dispensable *if it is n-dispensable.*

Theorem 4.2. *Deleting any subset of D from E leaves γ invariant.*

Proof. This is [2, Prop. B]. □

A more natural type of edges whose potential removal would not change the winning positions is the collection of all edges that are not chosen in optimal positional strategies.

[7] That is, reducing n, $|E|$ and M

Definition 4.3 (Suboptimal Edges)**.** *An edge $vw \in E$ played by player $\varepsilon(v)$ is called* suboptimal *if there does not exist an optimal positional strategy $f_{\varepsilon(v)} : V^*_{\varepsilon(v)} \to V$ for player $\varepsilon(v)$ such that $w = f_{\varepsilon(v)}(v)$. Let S denote the set of such edges vw, associated to (G, ε, μ).*

In what follows, for a positional strategy $f : V^* \to V$ let E_f be all edges that are selected by f: $E_f := \{vf(v) : v \in V^*\}$, and let $\mathcal{F}$ be the collection of optimal positional strategies $V^* \to V$.

Theorem 4.4. *Let $f \in \mathcal{F}$. Then deleting any subset of $E \backslash E_f$ from E leaves χ invariant.*

Proof. Let $T \subseteq E \backslash E_f$, and let $v \in V$. Write $f = f_- \sqcup f_+$. Let $\mathcal{G}$ denote the old mean payoff game, and let $\mathcal{G}'$ denote the mean payoff game obtained from $\mathcal{G}$ after removing T from E. Since we keep all of E_f in $\mathcal{G}'$, it must be that f_- and f_+ are positional strategies in $\mathcal{G}'$ for Min and Max, respectively. Now, pick optimal positional strategies f'_- and f'_+ in $\mathcal{G}'$. Note that these are also positional strategies in $\mathcal{G}$, as the edge set of $\mathcal{G}$ contains the edge set of $\mathcal{G}'$. Hence, f_-, f_+, f'_-, f'_+ are positional strategies in both games. Denoting by $\chi(v)$ and $\chi'(v)$ the values of v in $\mathcal{G}$ and $\mathcal{G}'$, respectively, we derive

$$\begin{aligned}
\chi(v) &\leqslant \chi(\boldsymbol{\sigma}(f'_- \sqcup f_+, v)) \quad (f_+ \text{ is optimal for Max in } G) \\
&\leqslant \chi'(v) \quad (f'_- \text{ is optimal for Min in } G') \\
&\leqslant \chi(\boldsymbol{\sigma}(f_- \sqcup f'_+, v)) \quad (f'_+ \text{ is optimal for Max in } G') \\
&\leqslant \chi(v) \quad (f_- \text{ is optimal for Min in } G)
\end{aligned}$$

so $\chi(v) = \chi'(v)$, completing the proof. □

Corollary 4.5. *Deleting any subset of S from E leaves χ invariant.*

Proof. This is seen as $S = E \backslash \bigcup_{f \in \mathcal{F}} E_f$, so any subset of S is a subset of a set of the form $E \backslash E_f$ for some $f \in \mathcal{F}$. Invoking Theorem 4.4 completes the argument. □

Corollary 4.6. *Problem* (2) *can be derived from* (3) *in polynomial time.*

Proof. Consider our general mean payoff game $\mathcal{G}$ along with known optimal positional strategies $f = f_- \sqcup f_+$. Obtain $\mathcal{G}'$ from $\mathcal{G}$ by deleting $E \backslash E_f$ from E, or, equivalently, by only keeping the edges that are in E_f. By Theorem 4.4, the values of $\mathcal{G}'$ coincide with the values of $\mathcal{G}$, but are easily found: due to the nature of E_f, the vertex set of $\mathcal{G}'$ is partitioned into several components each of which has either one cycle or one sink. Within each component, the cycle or sink is efficiently found and the values of the vertices in that component coincide and equal the mean weight of the cycle if a cycle is present or they equal $-\varepsilon(s) \cdot \infty$ if a sink s is present. □

We visualize this result in the following example.

Example 4.7. *Consider the following mean payoff game:*

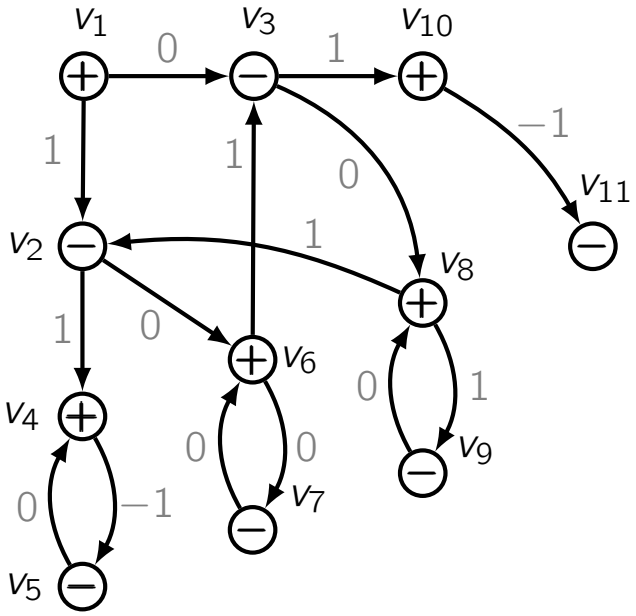


Figure 1: A mean payoff game on 11 vertices

In the above graph, there are precisely five vertices — namely, v_1, v_2, v_3, v_6, v_8 *— that have more than one out neighbor, and all of those have precisely two out neighbors. It turns out that each of those has a suboptimal outgoing edge. We visualize those suboptimal edges below, where all optimal edges are colored but all suboptimal edges are not.*

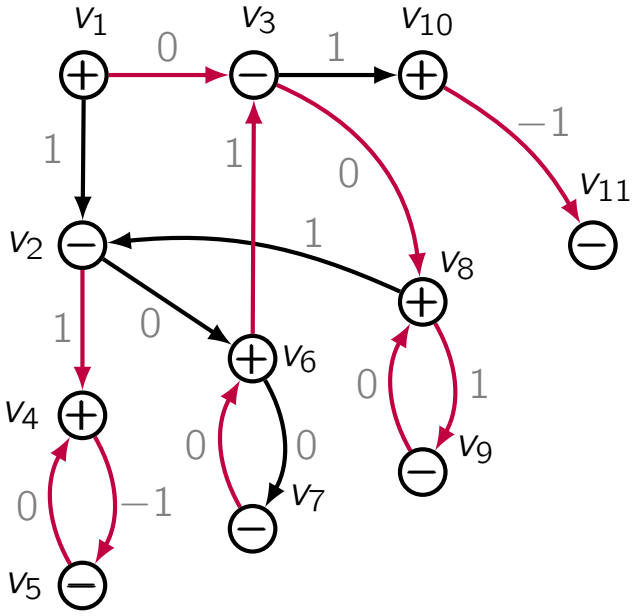


Figure 2: A mean payoff game on 11 vertices: optimal positional strategies visualized

As a final step, we discard all suboptimal edges, divide the graph into components, and read off the characteristic values per component based on the unique cycle or unique sink present:

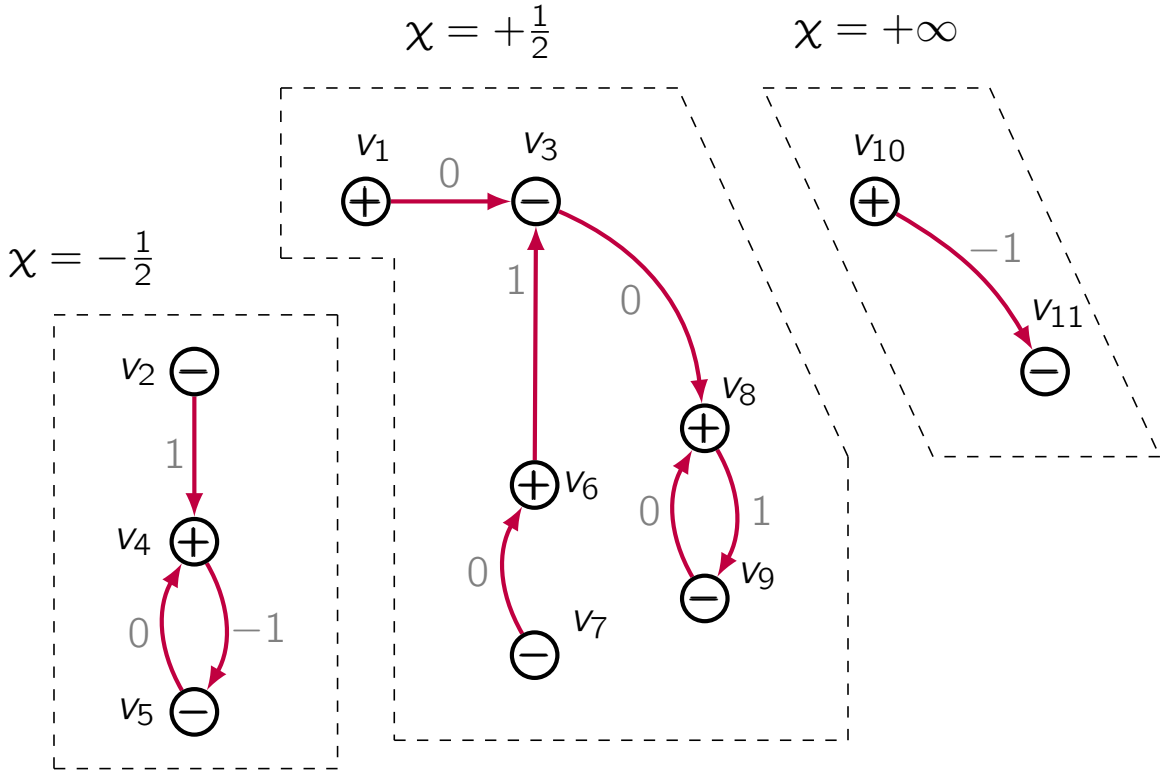


Figure 3: A mean payoff game on 11 vertices: reading off $\chi : V \to \overline{\mathbf{R}}$

Another way of seeing the result of the previous corollary is through he following result.

Corollary 4.8. *Optimal strategies preserve characteristic values. That is, given $f \in \mathcal{F}$ and $v \in V^*$, we have $\chi(v) = \chi(f(v))$.*

Proof. Reduce the mean payoff game $\mathcal{G}$ to $\mathcal{G}'$ by removing $E \backslash E_f$ from E. Then we have $\chi(v) = \chi'(v) = \chi'(f(v)) = \chi(f(v))$, where the first and third equality are due to Theorem 4.4 and the second is due to the fact that v and $f(v)$ reside within the same component of $\mathcal{G}'$. □

The converse of this statement, namely the statement that a positional strategy $f : V^* \to V$ with $\chi(v) = \chi(f(v))$ for all $v \in V^*$ is always optimal, is false, as the following example demonstrates.

Example 4.9. *Consider the following mean payoff game on* 8 *vertices:*

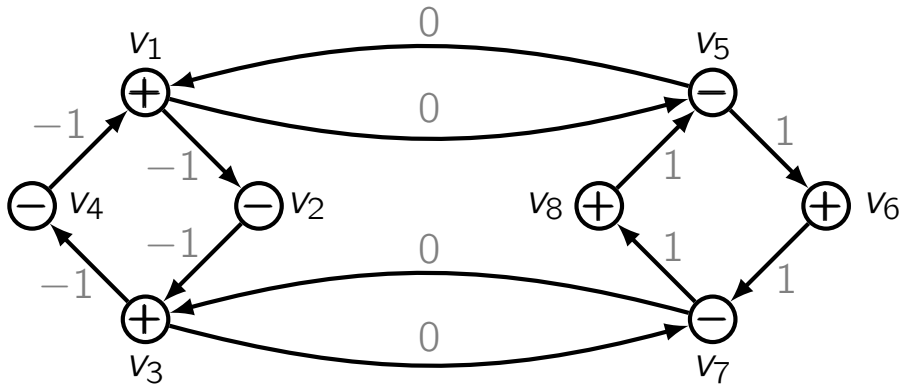


Figure 4: A bipartite mean payoff game on 8 vertices — "allValsAre0"

Since there are no sinks present, for each $i \in [8]$ *the value* $\chi(v_i)$ *equals the mean weight of a cycle. Max can ensure that each value is strictly greater than* -1 *by choosing a positional strategy* f_+ *for which* $f_+(v_1) = v_5$ *or* $f_+(v_3) = v_7$ *as to avoid ending up in the cycle* $v_1v_2v_3v_4v_1$*. Similarly, Min can avoid ending up in the cycle* $v_5v_6v_7v_8v_5$ *hence can ensure that each value is strictly less than* 1*. This means that* $\chi(v_i) = 0$ *for all* $i = 1, \ldots, 8$*. Yet, if Max chooses to play from* v_1 *to* v_2 *and from* v_3 *to* v_4*, then Max's strategy preserves characteristic values but will not be optimal as Min now can ensure all vertices to have value* -1*.*

There is more to observe about the mean payoff game depicted in Figure 4. The associated sets O and S of dispensable and suboptimal edges, respectively, are empty for instance. The path $\boldsymbol{\sigma} = v_1v_2v_3v_4$, however, would ensure Min to obtain value -1 on all vertices v_1, v_2, v_3, v_4 if Max would choose to play along $\boldsymbol{\sigma}$. Since Max is in full control of playing along $\boldsymbol{\sigma}$ — as all vertices belonging to Min on that path have out-degree one — one could say that $\boldsymbol{\sigma}$ as a *forced path* by Max is dispensable, as Min can force to close the cycle with negative payoff. This leads to the following generalization of Definition 4.1.

Definition 4.10 (Dispensable Forced Paths)**.** *Let* $k \in \mathbf{N}$*, let* $\boldsymbol{\sigma}$ *be a path of length* $l \leqslant k$ *starting at a vertex* $v \in V$ *and ending in a vertex* $w \in V$ *such that* $\boldsymbol{\sigma}$ *is forced by player* $p \in \{\pm\}$*. That is, for every vertex* v' *on* $\boldsymbol{\sigma}$ *strictly between the end points of* $\boldsymbol{\sigma}$ *that belongs to player* $-p$*, we have* $d^+(v') = 1$ *and the unique out-neighbour of* v' *is the next vertex in the path* $\boldsymbol{\sigma}$*. Then* $\boldsymbol{\sigma}$ *is said to be a* dispensable forced path *if one or both of the following hold:*

1. *player* $-p$ *can, starting from* w*, reach a sink* $s \in V_p \backslash V_p^*$ *such that every path* $\boldsymbol{\tau}$ *from* w *to* s *over which player* $-p$ *reaches* s *from* w *has length at most* $k - l - 1$*,* or
2. *player* $-p$ *can reach* v *from* w *such that every path* $\boldsymbol{\tau}$ *from* w *to* v *over which player* $-p$ *reaches* v *from* w *has length at most* $k - l$ *and satisfies the property that the mean weight of the formed closed path* $\boldsymbol{\sigma\tau}$ *has sign* $-p$*.*

Furthermore, $\boldsymbol{\sigma}$ *is called* dispensable *if it is* n*-dispensable.*

Indeed, a k-dispensable edge is a k-dispensable forced path of length one. However, the player for which an edge is dispensable is always specified, whereas this is not the case for general dispensable forced paths.

4.2 Finite Values Revisited

Consider Definition 2.13 in which the finite values $\eta_k(v)$ for mean payoff games are defined. Given $k \geqslant 1$ and $v \in V$, we consider the collection $N_k^+(v) = \{w \in N^+(v) : \eta_k(v) = \mu(v,w) + \eta_{k-1}(w)\}$ of out neighbours of v responsible for attaining the optimization (2.2). A natural quantity of v to consider along the way would be the size $d_k^+(v) = |N_k^+(v)|$ of the collection $N_k^+(v)$.

Given a mean payoff game (G, ε, μ), and given an integer $k \geqslant 1$, we consider the collection $\mathcal{F}_k := \{f : V^* \to V : f(v) \in N_k^+(v) \text{ for all } v \in V^*\}$. We notice that $\mathcal{F}_k$ has size $\prod_{v \in V^*} d_k^+(v)$ and hence has size 1 if and only if for every $v \in V^*$ there exists a unique $w \in V$ such that $vw \in E$ and $\eta_k(v) = \mu(v,w) + \eta_{k-1}(w)$.

Since we have $\lim_{k\to\infty} \eta_k(v)/k = \chi(v)$ for every $v \in V$, one would expect that the sequence $\mathcal{F}_1, \mathcal{F}_2, \ldots$ would stabilize into $\mathcal{F}$; that is, there exists $N \in \mathbf{N}$ such that for every integer $k \in N$ the collection $\mathcal{F}_k$ coincides with the collection $\mathcal{F}$ of optimal positional strategies. This turns out to be false, as the following example demonstrates.

Example 4.11. *Consider the following mean payoff game on the vertex set $V = \{v_1, v_2\}$:*

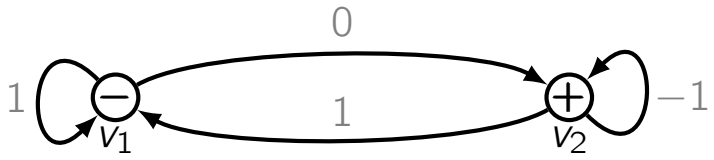


Figure 5: A mean payoff game on K_2

Using the very definition of finite values as given in Definition 2.13, we compute the first finite values of the game:

k	0	1	2	3	4	5
$\eta_k(v_1)$	0	0	1	1	2	2
$\eta_k(v_2)$	0	1	1	2	2	3

Table 1: The first 6 finite values of the mean payoff game in Figure 5

Clearly, Min has a positional strategy to avoid ending up in the cycle v_1v_1 which has (highest) mean weight 1, and Max has a positional strategy to avoid ending up in the cycle v_2v_2 which has (lowest) mean weight -1. As such, the only optimal positional strategy is the strategy that maps v_1 to v_2 and vice versa: $\mathcal{F} = \{v_i \mapsto v_{3-i}\}$. Yet, for even $k \geqslant 2$, we have $\eta_k(v_1) = \mu(v_1,v_1) + \eta_{k-1}(v_1) = \mu(v_1,v_2) + \eta_{k-1}(v_2)$ hence $N_k^+(v_1) = \{v_1, v_2\}$. One also establishes that $N_k^+(v_2) = \{v_1\}$ for all $k \geqslant 1$. Hence, for $k \geqslant 1$ we have

$$\mathcal{F}_k = \begin{cases} \{v_i \mapsto v_{3-i}\}, & k \text{ is odd}, \\ \{v_i \mapsto v_{3-i}, v_i \mapsto v_1\} & k \text{ is even}, \end{cases}$$

so for this mean payoff game it is not the case that $\mathcal{F} = \mathcal{F}_k$ for sufficiently large k. Namely, we see

that the sequence of $\mathcal{F}_k$ alternates between two fixed sets, and hence in every other instance contains strictly more than all optimal positional strategies. This example shows that, despite Theorem 2.14, local optimizations are not guaranteed to have a say in the determination of optimal positional strategies.

We expand the definition of finite values $\eta_k(v)$ (Definition 2.13) to that of finite values $\eta_k(v, w)$ that are taken along paths that are supposed to end in a chosen end vertex $w \in V$.

Definition 4.12 (Finite Values with Fixed End Points)**.** *Let (G, ε, μ) be the data of a mean payoff game, with $G = (V, E)$, and let $x \in V$ be a vertex which we will call the* end point*. For all $v \in V$ and $k \in \mathbf{Z}_{\geqslant 0}$, we define a value $\eta_k(v, x)$ as follows. We compute G_x as defined in Definition 1.11 and now take $G_x = (V_x, E_x)$ as ambient graph. We set $\eta_0(x, x) = 0$ and $\eta_0(v, x) = -\varepsilon(v) \cdot \infty$ for all $v \in V_x \backslash \{x\}$. Then for $k = 1, 2, 3, \ldots$ we inductively set*

$$\eta_k(v, x) := \begin{cases} \max\{\mu(v, w) + \eta_{k-1}(w, x) : w \in N^+(v)\}, & \text{if } \varepsilon(v) = +, \\ \min\{\mu(v, w) + \eta_{k-1}(w, x) : w \in N^+(v)\}, & \text{if } \varepsilon(v) = -, \end{cases}$$

where each out-neighborhood of v is taken with respect to G_x, and with the conventions $\max \varnothing = -\infty$ and $\min \varnothing = +\infty$.

4.3 Derived Mean Payoff Games

For a graph $G = (V, E)$ and positional strategy[8] $f : V^* \to V$, we set $E_f := \{vf(v) : v \in V^*\}$ as before. We look at the class of mean payoff games (G, ε, μ) on n vertices for which $|\mathcal{F}_n| = 1$. Then there exists a unique positional strategy $f : V \to V^*$ such that $\eta_n(v) = \mu(v, f(v)) + \eta_{n-1}(f(v))$ for all $v \in V^*$.

Taking the data of a mean payoff game (G, ε, μ) with the additional assumption that $|\mathcal{F}_n| = 1$, i.e., $\mathcal{F}_n = \{f\}$, we construct a graph $G^{(1)}$ as follows. We let $E_C^{(1)}$ be all edges of (V, E_f) that lie on a cycle in (V, E_f). We now let $E_R^{(1)}$ be the smallest subset of $E \backslash E_C^{(1)}$ such that, for vertices v, v' lying on distinct cycles C, C' in (V, E_f), we include an edge $vv' \in E_R^{(1)}$ if and only if in (G, ε, μ), $v \in R_{\varepsilon(v)}(v')$ ([9]) and $\varepsilon(v) \cdot (m(C') - m(C)) \geqslant 0$. We set $E^{(1)} := E_C^{(1)} \sqcup E_R^{(1)}$, let $V^{(1)}$ consist of all vertices in V that lie on edges in $E^{(1)}$, set $G^{(1)} := (V^{(1)}, E^{(1)})$, $\varepsilon^{(1)} := \varepsilon|_{V^{(1)}}$, and finally define $\mu^{(1)} := \mu|_{E_C^{(1)}} \sqcup (E_R^{(1)} \ni e \mapsto 0)$.

Definition 4.13 (Derived Mean Payoff Game)**.** *Given a mean payoff game $\mathcal{G} = (G, \varepsilon, \mu)$, the mean payoff game $\mathcal{G}^{(1)} = (G^{(1)}, \varepsilon^{(1)}, \mu^{(1)})$ is called the* first derived mean payoff game *of $\mathcal{G}$. We define the* derived series *$\mathcal{G}^{(0)}, \mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \ldots$ by $\mathcal{G}^{(0)} := \mathcal{G}$ and $\mathcal{G}^{(k)} := (\mathcal{G}^{(k-1)})^{(1)}$ for $k \in \mathbf{N}$.*

Calculating the first derived mean payoff game of a mean payoff game can be done efficiently, as the following algorithm shows.

[8] That is, a map $f : V^* \to V$ satisfying $vf(v) \in E$ for all $v \in V^*$

[9] That is, player $\varepsilon(v)$ can reach v' in G from v

Algorithm 6: Computing the derived mean payoff game

Data: a graph $G = (V, E)$, $\mu : E \to \mathbf{R}$, $\varepsilon : V \to \{\pm\}$, encoded in $\mathcal{G} = (G, \varepsilon, \mu)$, with $|\mathcal{F}_{|V|}| = 1$

Result: the first derived mean payoff game $\mathcal{G}^{(1)} = (G^{(1)}, \varepsilon^{(1)}, \mu^{(1)})$

Function `derivedMPG`(G, ε, μ)
  $n \leftarrow |V|$
  for $k = 1, \ldots, n$ **do**
    for $v \in V$ **do**
      if $\varepsilon(v) = 1$ **then**
        $\eta_k(v) \leftarrow \max\{\mu(v, w) + \eta_{k-1}(w) : w \in N^+(v)\}$
      else
        $\eta_k(v) \leftarrow \min\{\mu(v, w) + \eta_{k-1}(w) : w \in N^+(v)\}$
      end
    end
  end
  for $v \in V^*$ **do**
    for $w \in N^+(v)$ **do**
      if $\eta_n(v) = \mu(v, w) + \eta_{n-1}(w)$ **then**
        $f(v) \leftarrow w$
      end
    end
  end
  $E_f \leftarrow \{vf(v) : v \in V^*\}$
  for $v \in V$ **do**
    $l(v) \leftarrow \min\{i \in \{0, \ldots, n\} : f^i(v) \in V \backslash V^*\}$
    $c(v) = [v \in \{f^i(v) : 1 \leqslant i \leqslant l(v)\}, 1, 0]$
  end
  $V^{(1)} \leftarrow c^{-1}(1)$, $E_C^{(1)} \leftarrow E_f \cap (V^{(1)} \times V^{(1)})$
  for $v \in V^{(1)}$ **do**
    $L(v) \leftarrow \min\{i \in \{1, \ldots, n\} : f^i(v) = v\}$
    $m(v) \leftarrow \frac{1}{L(v)} \sum_{i=0}^{L(v)-1} \mu(f^i(v), f^{i+1}(v))$
  end
  $E_R^{(1)} \leftarrow \{\}$
  for $v, v' \in V^{(1)}$ *with* $v' \notin \{f^i(v) : i = 0, \ldots, L(v)\}$ **do**
    $E_R^{(1)} \leftarrow [v \in \mathtt{Reach}(G, \varepsilon, \{v'\}, \varepsilon(v))$ and $\varepsilon(v) \cdot (m(v') - m(v)) \geqslant 0, E_R^{(1)} \cup \{vv'\}, E_R^{(1)}]$
  end
  $\mu^{(1)} \leftarrow \mu|_{E_C^{(1)}} \sqcup (E_R^{(1)} \ni e \mapsto 0)$, $E^{(1)} \leftarrow E_C^{(1)} \cup E_R^{(1)}$
  $\varepsilon^{(1)} \leftarrow \varepsilon|_{V^{(1)}}$, $G^{(1)} \leftarrow (V^{(1)}, E^{(1)})$
  return $(G^{(1)}, \varepsilon^{(1)}, \mu^{(1)})$
end

Here, it is understood that the function `Reach` from Algorithm 2 is taken with the choices for p in $\{\pm\}$ instead of in $\{0, 1\}$. The following example indicates the procedure of deriving a mean payoff game in this fashion.

Example 4.14. *Consider the following mean payoff game on the vertex set $V = \{v_1, \ldots, v_6\}$, along with the effect of computing its first derived game:*

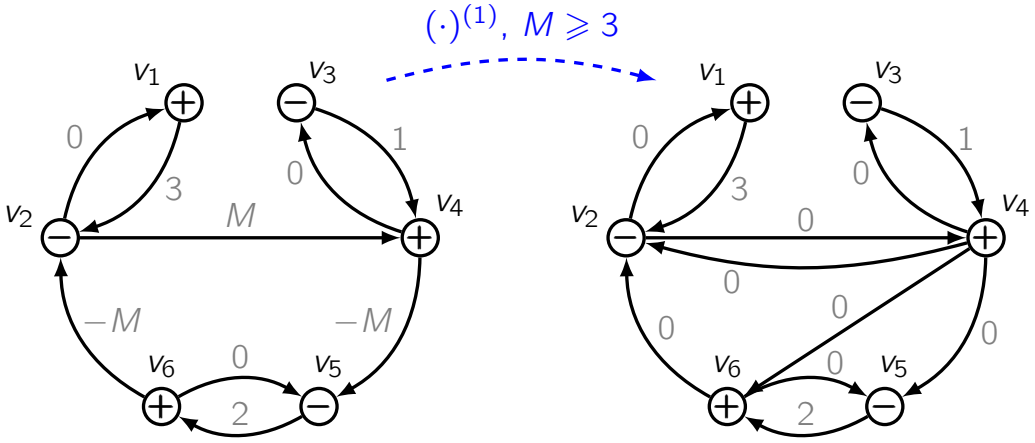


Figure 6: Computing the first derived game of a mean payoff game

As is apparent in the above figure, we have $E_C^{(1)} = \{v_1v_2, v_2v_1, v_3v_4, v_4v_3, v_5v_6, v_6v_5\}$. Writing C_1, C_2, C_3 for the cycles formed by the vertices $\{v_1, v_2\}$, $\{v_3, v_4\}$, and $\{v_5, v_6\}$, respectively, we see that the mean weights of these cycles satisfy $m(C_1) > m(C_3) > m(C_2)$. Now, player Min can force a play from $v_2 \in C_1$ to $v_4 \in C_2$, thus from a cycle to another cycle with mean weight that is better from the perspective of Min. Hence, an edge $v_2v_4 \in E_R^{(1)}$ of weight zero is included. Furthermore, Max can reach C_3 from a vertex in C_2, Max can reach C_1 from a vertex in C_3, and since Min does not have an escape from C_3 it follows that Max also can reach C_1 from C_2. Hence, more edges with weight zero are included in $E_R^{(1)}$, eventually leading to the graph on the right in figure 6.

The characteristic values of the game on the left in figure 6 are all equal to 1 as one can check — implying that every play following optimal positional strategies has terminal cycle C_3. The characteristic values on the right, however, are all equal to 1 as well. Yet, for the derived graph, the weight size is always equal to 3 and hence Algorithm 4 to compute the values of the mean payoff game on the left terminates in $O(1)$ time when applied to its derived game.

Given a mean payoff game $\mathcal{G}$, it is not always the case that the operation $(\cdot)^{(1)}$ keeps the winning positions invariant, even in the case none of the initial vertices becomes isolated. This is illustrated by the following example.

Example 4.15. *Consider the following mean payoff game on K_3, along with its derived graph:*

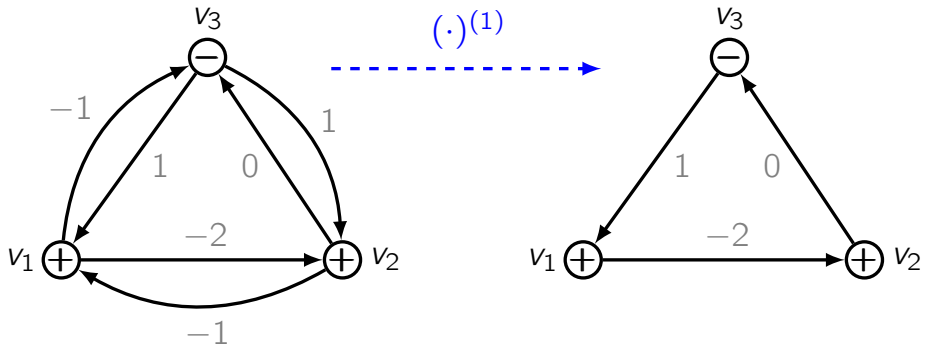


Figure 7: A mean payoff game on K_3

Max can avoid ending up in a cycle with negative weight, and Min can avoid ending up in a cycle with positive weight. As such, all three vertices in the mean payoff game on the left in Figure 7 are draw positions. Deriving the game, however, yields C_3 with negative weight and thus all positions are winning for Min in that game.

5 Probabilistic Complexity of Problem (1)

In this section, we prove that for every $n \in \mathbf{N}$ and every function $h(M)$ of $M \in \mathbf{N}$ satisfying $h(M) \to \infty$ as $M \to \infty$, the probability that Algorithm 4 operating on mean payoff games on n vertices with weight size M terminates correctly in $O(h(M))$ time[10] converges to 1 as $M \to \infty$. Here, the probability of success is taken with respect to selecting mean payoff games on n vertices with weight size M uniformly at random. The result follows by considering and properly estimating the number $R := \min_{C: m(C) \neq 0} |m(C)|$ where C runs over all directed cycles of the graph with nonzero mean weight. Before we state the result, we first classify how we count mean payoff games on n vertices with weight size M. Since the result does not depend on the partition $V = V_- \sqcup V_+$ of the vertices, it will be sufficient to only come up with a method to count weighted digraphs.

5.1 Counting Weighted Digraphs

Given $n \in \mathbf{N}$ and $M \in \mathbf{N}$, one can take a look at the class $\mathscr{G}(n, M)$ of all weighted graphs (G, μ) with $V(G) = [n]$ and $\|\mu\|_\infty = M$. One readily notes that the members of the class $\mathscr{G}(n, M)$ can be counted as (where we use Newton's Binomial Theorem)

$$\begin{aligned}
|\mathscr{G}(n,M)| &= \sum_{k=0}^{n^2} \sum_{\substack{E \subset [n]\times[n]: \\ |E|=k}} |\{\mu : E \to \{-M,\dots,M\} : \|\mu\|_\infty = M\}| \\
&= \sum_{k=0}^{n^2} \binom{n^2}{k} \cdot |\{f : [k] \to \{-M,\dots,M\} : \|f\|_\infty = M\}| \\
&= \sum_{k=0}^{n^2} \binom{n^2}{k} \cdot \left(\left|\{-M,\dots,M\}^{[k]}\right| - |\{f : [k] \to \{-M,\dots,M\} : \|f\|_\infty < M\}| \right) \\
&= \sum_{k=0}^{n^2} \binom{n^2}{k} \cdot \left((2M+1)^k - (2M-1)^k \right) \\
&= \sum_{k=0}^{n^2} \binom{n^2}{k} \cdot (2M+1)^k \cdot 1^{n^2-k} - \sum_{k=0}^{n^2} \binom{n^2}{k} \cdot (2M-1)^k \cdot 1^{n^2-k} \\
&= (2M+2)^{n^2} - (2M)^{n^2} = 2^{n^2} \cdot \left((M+1)^{n^2} - M^{n^2} \right).
\end{aligned}$$

Given a *property* P on $\mathscr{G}(n, M)$, we can define the probability that a weighted graph (G, μ) from $\mathscr{G}(n, M)$ when uniformly randomly selected satisfies P as

$$\mathbb{P}_{n,M}((G,\mu) \text{ satisfies } P) := \frac{|(G,\mu) \in \mathscr{G}(n,M) : (G,\mu) \text{ satisfies } P|}{|\mathscr{G}(n,M)|}.$$

More generally, for any finite class $\mathscr{G}$ of weighted digraphs (G, μ) and property P on $\mathscr{G}$ one can write the probability that a weighted digraph drawn uniformly at random satisfies P as

$$\mathbb{P}_{\mathscr{G}}((G,\mu) \text{ satisfies } P) := \frac{|(G,\mu) \in \mathscr{G} : (G,\mu) \text{ satisfies } P|}{|\mathscr{G}|}$$

[10] We do not include n as n is fixed beforehand

and one notices that we have $\mathbb{P}_{n,M} = \mathbb{P}_{\mathscr{G}(n,M)}$ for short. Given two properties P, Q on a finite class $\mathscr{G}$ of weighted graphs, we also have the conditional probability

$$\begin{aligned}\mathbb{P}_{\mathscr{G}}((G,\mu) \text{ satisfies } P \mid (G,\mu) \text{ satisfies } Q) &:= \frac{\mathbb{P}_{\mathscr{G}}((G,\mu) \in \mathscr{G} : (G,\mu) \text{ satisfies } P \text{ and } Q)}{P_{\mathscr{G}}((G,\mu) \in \mathscr{G} : (G,\mu) \text{ satisfies } Q)} \\ &= \frac{|(G,\mu) \in \mathscr{G} : (G,\mu) \text{ satisfies } P \text{ and } Q|}{|(G,\mu) \in \mathscr{G} : (G,\mu) \text{ satisfies } Q|}.\end{aligned}$$

Remark 5.1. *An alternative method to count $\mathscr{G}(n,M)$ is by viewing a weighted graph $(G,\mu) \in \mathscr{G}(n,M)$ as a matrix $A \in (([-M,M] \cap \mathbf{Z}) \cup \{\infty\})^{n \times n}$, where $A_{vw} = \mu(v,w)$ if $vw \in E(G)$ and $A_{wv} = \infty$ otherwise. We see that $\mathscr{G}(n,M)$ is characterized as*

$$([-M,M] \cap \mathbf{Z}) \cup \{\infty\})^{n \times n} \backslash ([-(M-1),M-1] \cap \mathbf{Z}) \cup \{\infty\})^{n \times n} \tag{5.1}$$

and the above collection contains precisely $(2M+2)^{n^2} - (2M)^{n^2}$ elements.

5.2 The Probabilistic Speed-Up of Algorithm 4

For a weighted graph (G,μ), let $R(G,\mu)$ be the quantity $\min\{|m(C)| : m(C) \neq 0\}$, where C runs over all directed cycles in (G,μ) with nonzero mean weight, as defined in Definition 3.3.

Theorem 5.2. *Let $n \in \mathbf{N}$, and let $h : \mathbf{N} \to \mathbf{R}$ be a function of $M \in \mathbf{N}$ satisfying $h(M) = \omega(1)$, i.e. $h(M) \to \infty$ as $M \to \infty$. Then we have*

$$\lim_{M \to \infty} \mathbb{P}_{n,M}\left(R(G,\mu) \geqslant \frac{M}{h(M)}\right) = 1. \tag{5.2}$$

The proof of Theorem 5.2 is surprisingly elementary, and we utilize the counting method as presented in Remark 5.1 to deduce (5.2). We first prove an auxiliary result, that in turn relies on a technical lemma — Lemma 5.7 — that we prove independently at the very end of this section.

Definition 5.3. *Construct, for all $k \in [n]$, the property P_k on $\mathscr{G}(n,M)$ as follows: we say that $(G,\mu) \in \mathscr{G}(n,M)$ satisfies P_k precisely when for each selection of k distinct edges $e_1, \ldots, e_k \in E(G)$ we have $\frac{M}{h(M)} \leqslant \left|\frac{1}{k}\sum_{i=1}^{k} \mu(e_i)\right|$, and we denote this by $P_k(G,\mu)$.*

Lemma 5.4. *Let n and h be as in Theorem 5.2, and let $k \in [n]$. Then we have*

$$\lim_{M \to \infty} \mathbb{P}_{n,M}(P_k(G,\mu)) = 1. \tag{5.3}$$

Proof. In the proof, we drop the subscripts indicating n, M whence we write $\mathbb{P}(\cdot)$ instead of $\mathbb{P}_{n,M}(\cdot)$. We modify P_k to the property P_k' where we say that (G,μ) satisfies P_k' precisely when for each k-tuple of edges $e_1, \ldots, e_k \in E(G)$ we have $\frac{M}{h(M)} \leqslant \left|\frac{1}{k}\sum_{i=1}^{k} \mu(e_i)\right|$. Then we have $\mathbb{P}(P_k(G,\mu)) \geqslant \mathbb{P}(P_k'(G,\mu))$ as $P_k'(G,\mu)$ implies $P_k(G,\mu)$. Now, we have

$$\begin{aligned}\mathbb{P}(P_k'(G,\mu)) &= \sum_{k=0}^{n^2} \mathbb{P}(P_k'(G,\mu) || E(G)| = k) \cdot \mathbb{P}(|E(G)| = k) \\ &\geqslant \min_{0 \leqslant k \leqslant n^2} \mathbb{P}(P_k'(G,\mu) || E(G)| = k)\end{aligned} \tag{5.4}$$

as the middle term is a convex combination in the quantities $\mathbb{P}(P'_k(G,\mu)||E(G)| = k)$ (where $k = 0,\ldots,n^2$), with coefficients $\mathbb{P}(|E(G)| = k)$ for $k = 0,\ldots,n^2$. Clearly, the minimum in (5.4) is attained precisely for $k = n^2$, and we have

$$\mathbb{P}(P'_k(G,\mu)) \geqslant \mathbb{P}(P'_k(G,\mu)||E(G)| = n^2).$$

We invoke Lemma 5.7 (given at the end of this subsection) by identifying the collection of weighted, complete digraphs with maximum absolute weight M with $[n^2, M]'$ (where $[\cdot,\cdot]'$ is defined in Lemma 5.7) to deduce that

$$\lim_{M\to\infty} \mathbb{P}(P'_k(G,\mu)||E(G)| = n^2) = 1,$$

and from this we obtain (5.3). □

Proof of Theorem 5.2. In this proof, we again drop the subscripts indicating n, M whence we write $\mathbb{P}(\cdot)$ instead of $\mathbb{P}_{n,M}(\cdot)$. We observe that if (G,μ) satisfies P_k for all $k \in [n]$, then certainly $R(G,\mu) \geqslant \frac{M}{h(M)}$. It follows that, given M, we have

$$\begin{aligned}
\mathbb{P}\left(R(G,\mu) \geqslant \frac{M}{h(M)}\right) &\geqslant \mathbb{P}\left(P_k(G,\mu) \text{ for all } k \in [n]\right) \\
&= 1 - \mathbb{P}(\exists k \in [n] : \neg P_k(G,\mu)) \\
&\geqslant 1 - \sum_{k=1}^{n} \mathbb{P}(\neg P_k(G,\mu)) \\
&= 1 - n + \sum_{k=1}^{n} \mathbb{P}(P_k(G,\mu))
\end{aligned}$$

and we obtain (5.2) from Lemma 5.4. □

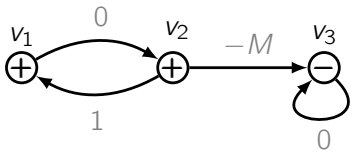
An immediate consequence of Theorem 5.2 becomes apparent once we combine it with Theorem 2.16 in which it was stated that Algorithm 4 correctly terminates in $4nM/R+1$ steps. Here, R again indicates the minimum among the absolute values of the mean weights of the directed cycles of the weighted graphs that have nonzero mean weight. Upon combining these results we derive the central result of this thesis.

Theorem 5.5 (Probabilistic Speed-Up of Algorithm 4)**.** *Let $n \in \mathbf{N}$, and let $h : \mathbf{N} \to \mathbf{R}$ be a function of $M \in \mathbf{N}$ satisfying $h(M) = \omega(1)$, i.e. $h(M) \to \infty$ as $M \to \infty$. Consider mean payoff games on n vertices, with weight functions with maximum absolute value equal to M, selected uniformly at random. Then the probability that Algorithm 4 correctly terminates upon running the loop starting at line 7 from $k = 1$ to $k = \lfloor 4n \cdot h(M) \rfloor + 1$, instead of from $k = 1$ to $k = 4n^2M + 1$, converges to 1 when $M \to \infty$.*

Proof. All mean payoff games on n vertices with weight size M satisfying $R \geqslant M/h(M)$ ensure Algorithm 4 to terminate with success probability 1 if we let k run to at least $4nM/R + 1 \leqslant 4nM/(M/h(M)) + 1 = 4n \cdot h(M) + 1$, where the minimum of $4nM/R+1$ steps was derived in Theorem 2.16. Since the proportion of such mean payoff games relative to the collection of mean payoff games on n vertices with weight size M goes to unity as M goes to infinity (this follows by combining Theorem 5.2 with the observation that the weights are independent of the partition of the vertex set), it must be that upon selecting mean payoff games on n vertices with weight size M uniformly at random, the probability that Algorithm 4 terminates correctly in this way converges to 1 as M goes to infinity. □

Despite the fact that for fixed n the proportion of mean payoff games to which are assigned the correct signs of the characteristic values goes to 1 as $M \to \infty$, this proportion never coincides with 1 for sufficiently large M in the case $n \geqslant 3$. This is demonstrated by the following example, for $n = 3$, and thus extends to every finite graph on at least three vertices as we can add isolated vertices indefinitely to this graph to obtain families of examples in terms of increasing M and increasing $n \geqslant 3$.

Example 5.6. *Consider the following mean payoff game on* 3 *vertices:*



The only choice a player has is the choice Max has at v_2, and when playing optimally, Max will play from v_2 to v_1 and hence $\chi(v_1) = \chi(v_2) = 1/2$ and $\chi(v_3) = 0$ in this case. However, upon considering Algorithm 4, the first value of k for which $\eta_k(v_2) > 2nM$ occurs at $k = 4nM + 1$, which for fixed n is in $\Omega(M)$. Considering that the occurrence of a k satisfying $\eta_k(v_2) > 2nM$ is needed to establish that $\gamma(v_2) = 1$, this example proves that the probability in Theorem 5.5 never attains 1*, for whatever choice of $n \geqslant 3$ and $h(M) = \omega(1)$ and $h(M) = o(M)$.*

Finally, we prove the technical result needed to establish Lemma 5.4. We do not give a counting argument, but we give a probabilistic reasoning instead.

Lemma 5.7. *Let h be as in Theorem 5.2, let $N \in \mathbf{N}$, and let $k \in [N]$. For $M \in \mathbf{N}$, write $[N, M] := ([-M, M] \cap \mathbf{Z})^N$ and $[N, M]' := [N, M] \backslash [N, M-1]$. Then we have*

$$\lim_{M\to\infty} \frac{\left|\left\{v \in [N, M]' : \frac{M}{h(M)} \leqslant \left|\frac{1}{k}\sum_{i=1}^{k} v_{x_i}\right| \ \forall\, x_1, \ldots, x_k \in [N]\right\}\right|}{|[N, M]'|} = 1. \tag{5.5}$$

Proof. We prove that one minus the quantity in (5.5) the limit is taken of goes to zero as M goes to infinity. To this end, we first derive

$$\begin{aligned}
&1 - \frac{\left|\left\{v \in [N, M]' : \frac{M}{h(M)} \leqslant \left|\frac{1}{k}\sum_{i=1}^{k} v_{x_i}\right| \ \forall\, x_1, \ldots, x_k \in [N]\right\}\right|}{|[N, M]'|} \\
&= \frac{\left|\left\{v \in [N, M]' : \exists x_1, \ldots, x_k \in [N] : \left|\frac{1}{k}\sum_{i=1}^{k} v_{x_i}\right| < \frac{M}{h(M)}\right\}\right|}{|[N, M]'|} \\
&= \frac{1}{|[N, M]'|}\left|\bigcup_{(x_1,\ldots,x_k)\in[N]^k} \left\{v \in [N, M]' : \left|\frac{1}{k}\sum_{i=1}^{k} v_{x_i}\right| < \frac{M}{h(M)}\right\}\right| \\
&\leqslant \sum_{(x_1,\ldots,x_k)\in[N]^k} \frac{\left|\left\{v \in [N, M]' : \left|\frac{1}{k}\sum_{i=1}^{k} v_{x_i}\right| < \frac{M}{h(M)}\right\}\right|}{|[N, M]'|}
\end{aligned} \tag{5.6}$$

so we obtain (5.5) if we can show that all N^k summands in the final expression in the above go to zero as M goes to infinity. To this end, take $x_1, \ldots, x_k \in [N]$. We select v uniformly at random from $[N, M]'$. For $x, y \in [N]$, we set $\delta_{xy} := 1$ if $x = y$ and $\delta_{xy} := 0$ otherwise. Due to the additive symmetry of $[N, M]$

and hence $[N, M]'$, we have

$$\mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} v_{x_i}\right] = 0. \tag{5.7}$$

Next, we compute a lower bound of the expectation value of the square of $\frac{1}{k}\sum_{i=1}^{k} v_{x_i}$ and show that it is in $\Omega(M^2)$:

$$\begin{aligned}
\mathbb{E}\left[\left(\frac{1}{k}\sum_{i=1}^{k} v_{x_i}\right)^2\right] &= \frac{1}{k^2}\sum_{i=1}^{k}\mathbb{E}[v_{x_i}^2] + \frac{1}{k^2}\sum_{i\neq j}\delta_{x_i x_j}\mathbb{E}[v_{x_i} v_{x_j}] \\
&\geqslant \frac{1}{k^2}\sum_{i=1}^{k}\mathbb{E}[v_{x_i}^2] \\
&\geqslant \frac{1}{k^2}\cdot k\cdot\mathbb{E}[v_{x_1}^2] \\
&= \frac{1}{k}\cdot\frac{1}{2M+1}\cdot\sum_{j=-M}^{M} j^2 \\
&= \frac{1}{k}\cdot\frac{1}{2M+1}\cdot 2\cdot\sum_{j=1}^{M} j^2 \\
&= \frac{1}{k}\cdot\frac{1}{2M+1}\cdot 2\cdot\frac{M(M+1)(2M+1)}{6} \\
&= \frac{1}{3k}M(M+1) > \left(\frac{M}{\sqrt{3k}}\right)^2.
\end{aligned} \tag{5.8}$$

The first equality in (5.8) follows as for distinct $x, y \in [N]$ we have $\mathbb{E}[v_x v_y] = 0$, due to the additive symmetry of $[N, M]'$. The first inequality in (5.8) follows as whenever $x_i = x_j$, we have $\mathbb{E}[v_{x_i} v_{x_j}] = \mathbb{E}[v_{x_j}^2] \geqslant 0$. The second inequality in (5.8) follows because before the inequality, the values v_{x_i} were still coupled to v, and $[N, M]'$ has on average wider spacing than $[N, M]$, whereas the value v_{x_1} was understood to be taken independently from $[-M, M] \cap \mathbf{Z}$ from that point on. Let σ_M be the standard deviation of $\frac{1}{k}\sum_{i=1}^{k} v_{x_i}$ upon selecting v uniformly at random from $[N, M]'$. Combining (5.7) and (5.8) gives $\sigma_M > \frac{M}{\sqrt{3k}}$ hence $\sigma_M = \Omega(M)$. Then it is immediate that $(M/h(M))/\sigma_M \xrightarrow{M\to\infty} 0$. Spelled out,

$$\frac{M/h(M)}{\sigma_M} < \frac{M/h(M)}{M/\sqrt{3k}} = \frac{\sqrt{3k}}{h(M)} \xrightarrow{M\to\infty} 0$$

as $h(M) \xrightarrow{M\to\infty} \infty$. Since we have been selecting v uniformly at random from $[N, M]'$ and the standard deviation of $\frac{1}{k}\sum_{i=1}^{k} v_{x_i}$ compared to $M/h(M)$ approaches infinity as $M \to \infty$, it must be that the proportion of $v \in [N, M]'$ satisfying $\left|\frac{1}{k}\sum_{i=1}^{k} v_{x_i}\right| < M/h(M)$ approaches zero as $M \to \infty$. As $x_1, \ldots, x_k \in [N]$ were selected arbitrarily, each summand in the final expression in (5.6) goes to zero as $M \to \infty$, and we obtain (5.5). □

6 Discussion and Conclusion

Before we state our final conclusions, we discuss our results and suggest how to improve on those, as well as mention a few other ideas for further research.

6.1 Discussion of Results

In this thesis we have kept a global picture in sight instead of purely focusing on subclasses of mean payoff games. The central result in this thesis (Theorem 5.5) was found by keeping this view, whilst exploiting a very small subtlety in the proof of correctness of Algorithm 4. This subtlety came down to observing that the minimum R of the absolute values of the cycle means of cycles with nonzero mean is always at least $1/n$ where n is the number of vertices, and subsequently noting that R is statistically sufficiently larger to derive the main result.

We indicate however that the convergence rate of the probability in Theorem 5.5 deserves its own analysis. Specifically, we consider it very likely that the convergence rate is heavily negatively affected if we let n increase. The result of Theorem 5.5 still stands of course, yet at a questionable rate of convergence.

6.2 Suggestions for Further Research

Here, we discuss suggestions for further research and potential further improvements.

6.2.1 The Potential of Problem (1) Lying in P

Given that Zwick and Paterson conjecture in [12, Section 8] that the decision problem of mean payoff games lies in P, and given Theorem 5.5, we would like to emphasize that we now have even more reason to believe that the decision problem lies in P. A potential solution could be as follows. The key idea is to come up with a function $h(M) = \omega(1)$ simultaneously residing in $O((\log M)^d)$ for some $d \in \mathbf{N}$ that separates mean payoff games into two subclasses $\mathrm{MPG}_{R \geqslant M/h(M)}$ and $\mathrm{MPG}_{R<M/h(M)}$ upon comparing $M/h(M)$ to the minimum absolute nonzero cycle mean R, and then to hybridize Algorithm 4 with a new algorithm with pre-condition $R < M/h(M)$.

Since in a hybride algorithm we cannot avoid the running time of a potential check on a pre-condition, we must come up with an efficient method to distinguish between the classes $\mathrm{MPG}_{R \geqslant M/h(M)}$ and $\mathrm{MPG}_{R<M/h(M)}$. The most natural check would be an algorithm that solves problem (C2), i.e. the problem of finding the minimum among the absolute values of the means of the cycles that have nonzero cycle mean. Despite the apparent similarity of this problem to the minimum (or maximum) cycle mean problem for which Karp gave an efficient algorithm in [10], we have not found an efficient solution to problem (C2) yet. However, an efficient method to determine whether the minimum absolute nonzero cycle mean R lies in an interval of the form $[-\alpha, \alpha]$ with $\alpha \in \mathbf{R}$ (or in $\mathbf{Q}$ or even in $\mathbf{Z}$) ought to be sufficient for the pre-condition to separate games into the classes $\mathrm{MPG}_{R \geqslant M/h(M)}$ and $\mathrm{MPG}_{R<M/h(M)}$. The reason that we can relax α to be an integer lies in the fact that, as M goes to infinity, so will $M/h(M)$ if we choose $h(M) = \omega(1)$. A very tangible approach to keep the running time polynomial in $\log M$ would be to select $h(M) = (\log M)^d$ for

some $d \in \mathbf{N}$.

Once we have the efficient pre-condition check, a second ingredient will be the most challenging as it will require an efficient algorithm to determine the winning positions. However, it may assume that $R < M/(\log M)^d$. From that point onward, one might want to look for methods to efficiently solve the decision problem that assumes this pre-condition. On the other hand, in case problem (1) does not admit a polynomial-time solution, then one has to come up with a grid of mean payoff games $(\mathcal{G}_{n,M})_{(n,M)\in\mathrm{N}\times\mathrm{N}}$ for which $R(\mathcal{G}_{n,M}) < M/h(M)$ for large enough M for any $h(M) = \omega(1)$, and such grids lie sparse in the collection of all mean payoff games, as Theorem 5.2 suggests but does not guarantee.

Again, such hybdridization is only possible if we have an efficient check to distinguish between weighted digraphs with $R < M/h(M)$ and those with $R \geqslant M/h(M)$. This calls for further research on finding an efficient solution to the minimum absolute nonzero cycle mean problem on weighted digraphs.[11]

6.2.2 The Potential of Problem (2) Lying in P

Since the probabilistic speed-up of an algorithm solving the decision problem was based on mapping how far away from zero the mean weights of the cycles with nonzero mean weight are situated, Zwick's and Paterson's value finding algorithm might admit a similar probabilistic speed-up. Instead of looking at the quantity R for weighted digraphs, we look at the quantity Δm, defined by

$$\Delta m := \min_{(C,C'):m(C)\neq m(C')} |m(C) - m(C')|$$

where (C, C') runs over all pairs of directed cycles of the weighted digraph in question such that the mean weights of the two cycles in the pair are not the same. We believe that, for fixed $n \in \mathbf{N}$ and $h(M) = \omega(1)$, the proportion of weighted digraphs with $\Delta m \geqslant M/h(M)$ approaches 1 if M goes to infinity. This would be similar to the proportion of weighted digraphs with $R \geqslant M/h(M)$ approaching 1 if M goes to infinity, as proven in Theorem 5.2. Then if we can show by re-analyzing the proof of the best-known method to the value finding problem [12, Thm. 2.3] that it is sufficient to compute the first $O(h(M))$ collections of finite values $(\eta_k(v)/k)_{v\in V}$ for $k \leqslant O(h(M))$ to correctly give all exact values $\chi(v)$, we have extended the result in Section 5 to problem (2).

Should this be possible, then there is a higher likelihood that problem (2) as well resides in P. Again, in this case one could come up with a hybrid algorithm of which the second half considers mean payoff games with the precondition that Δm is small.

6.2.3 Improving Bounds

The initial running times $O(n^4 \cdot M)$ and $O(n^3 \cdot M)$ of the (ideas of the) algorithms Zwick and Paterson proposed in [12] followed from two estimations. The first is [12, Thm. 2.2] (or, equivalently, Theorem 2.14, the second is the fact that $R \geqslant 1/n$ for every weighted digraph on n vertices with at least one

[11] Due to the nature of mean payoff games considered here, one can restrict to integer-weighted digraphs, as opposed to Karp who considered general real-weighted digraphs

nonzero weight. We believe that both inequalities can be improved. However, the second was improved probabilistically and does not allow for a deterministic improvement, as [2, Figure 3] illustrates, where we have $R = O(\frac{1}{n} \cdot M^0)$ ([12]). The first inequality assumes the worst-case scenario, but we also believe this inequality cannot be deterministically improved as well, as there are plenty of weighted digraphs for which the absolute weight of each edge is in $\Omega(M)$ yet there are cycles with mean weight very close to $1/n$ in absolute value.

A similar suggestion for further research is to investigate the convergence rate of (5.2) as a function of n. We have reason to believe that the rate of convergence is highly dependend on n, but not in what particular fashion.

6.3 More on Probabilistic Complexity Classification

The probabilistic speed-up of Zwick and Paterson's algorithm to determine the winning positions might allow for the following classification. Given $\epsilon > 0$, there might exist $K, C, d > 0$ (depending on ϵ) only such that whenever a mean payoff game on n vertices with weight size M satisfies $\max(n, M) > K$, Zwick and Paterson's algorithm runs in time at most $C \cdot n^3 \cdot (\log M)^d$ with error probability at most ϵ. Should this be the case, then the class of mean payoff games might be part of one of the randomized complexity classes as described in [3].

6.4 Conclusion

In this thesis, we have formulated the three main problems on mean payoff games on integer-weighted digraphs. We first gave an overview of existing methods in Section 2. It proved hard to find families of mean payoff games on which there are algorithms solving the decision problem that terminate in purely polynomial time. On the other hand, a small subtlety (Theorem 2.14 and Theorem 2.16) allowed us to see that for large weights, the proportion of mean payoff games that allow for a decision algorithm terminating in purely polynomial time goes to one. Since there are always very particular classes of mean payoff games to find that admit polynomial time solutions, a more global result might have been refreshing. Particularly, Theorem 5.5 might be seen as support of Zwick and Paterson's conjecture that the decision problem of mean payoff games lies in P. We have seen that trying to look at computational problems in both a local and a global way is benificial to gain new insights in problems of this nature □

[12] We include M in this notation as to indicate that we do not pre-fix M

Appendices

As appendices, we include implementations in PARI/GP [11] of the main algorithms in this thesis, along with auxiliary functions. A great deal of these scripts have been used to come up with concrete examples throughout this thesis. Before displaying the actual scripts, we give an overview of the most important algorithms and functions.

Name	*Implementing*	*Description*
`addInf`	—	addition on $\mathbf{R} \cup \{\pm\infty\}$
`decide`	Algorithm 4	deciding the winning positions
`derMPG`	Algorithm 6	computing the derived MPG
`finVals`	Definition 2.13	computing finite values
`isSubset`	—	check if $A \subseteq B$
`multInf`	—	multiplication on $\mathbf{R} \cup \{\pm\infty\}$
`posStr2Vals`	—	compute values from strategy
`Reach`	Algorithm 2	reachability on (G, ε)
`ReachOne`	Algorithm 2	one-step reachability on (G, ε)
`solveVertex`	Algorithm 3	compute characteristic value
`strFromVals`	—	compute strategy from values
`underlyingGraph`	—	MPG to underlying graph

A Implementations of Main Algorithms in PARI/GP

Below, we give implementations in PARI/GP of the main algorithms in this thesis. Quite some of them use auxiliary functions, and those are given in Appendix B. Mean Payoff Games (G, ε, μ) on $n \in \mathbf{N}$ vertices are inputted as a pair `[A, eps]`, in such a way that (G, μ) is encoded in `A`. Then we can view $[n] = \{1, \ldots, n\}$ as the vertex set. Here, `A` is an $n \times n$ matrix containing values from $\mathbf{Z} \cup \{\pm\infty\}$, and `eps` is a vector of length n with entries in $\{\pm 1\}$, indicating which player owns which vertex. Then the element `A[i, j]` equals the weight of ij if ij is an edge, and if ij is not an edge then `A`$[i, j]$ equals $+\infty$ if vertex i belongs to Min and $-\infty$ if vertex i belongs to Max. Note that in this fashion ε is reflected in `A` if the graph is not complete but only deducible from `A` if none of the n sets of out-neighbours coincides with the entire vertex set. Positional strategies are given as vectors of length n with entries in $\{0\} \cup [n]$, where all values of such vector equal to zero are indexed precisely by all integers in $[n]$ representing sinks. Such matrix `A` can be translated to its underlying (unweighted) graph `G`. Unweighted graphs `G` are represented as $n \times n$ matrices, where `G[i, j]` equals 1 if there is an edge from `i` to `j` and zero otherwise.

Listing 1. Below we implement the two-player reachability algorithm (Algorithm 2) in PARI/GP, but not for p ranging over $\{0, 1\}$ but for p ranging over $\{\pm 1\}$.

```
Reach = ((G, eps, S, p) -> \
  R = S; R2 = S; \
  R = ReachOne(G, eps, R2, p); \
  while (R2 != R, \
    R2 = R; \
    R = ReachOne(G, eps, R2, p); \
  ); \
  R; \
);

ReachOne = ((G, eps, S, p) -> \
  n = matsize(G)[1]; \
  T = S; \
  for (v = 1, n, \
  if (S[v], \
    for (u = 1, n, \
    if (G[u, v] && !S[u], \
      if(eps[u] == p || isSubset(G[u,], S), T[u] = 1); \
    ); \
    ); \
  ); \
  ); \
  T; \
);
```

Listing 1: Reachability for two players $\{-, +\}$ in PARI/GP

Listing 2. Here, we implement an algorithm that finds a strategy given the outcomes of the characteristic values of the mean payoff game. The approach is brute-force, and we note that the method in [12, Thm. 3.1] is much faster.

```
strFromVals = ((MPG, vals) -> \
  [A, eps] = MPG; \
  n = matsize(A)[1]; \
  G = underlyingGraph(A); \
  deg = vector(n, v, vecsum(G[v,])); \
  f = vector(n); \
  ind = vector(n); \
  finalInd = vector(n); \
  for (v = 1, n, \
  if (deg[v] > 0, \
    for (w = 1, n, \
      ind[v] = if(ind[v] == 0 && G[v, w], w, ind[v]); \
      finalInd[v] = if(G[v, w], w, finalInd[v]); \
    ); \
  ); \
  ); \
  terminate = 0; \
  while (!terminate, \
    f = vector(n, v, if(deg[v] > 0, ind[v], 0)); \
    valsf = posStr2Vals(MPG, f); \
    if (vals == valsf, break); \
    terminate = 1; \
    success = 0; \
    for (v = 1, n, \
    if (deg[v] > 0, \
      if(ind[v] < finalInd[v], \
        for(w = ind[v] + 1, n, \
          if(G[v, w], \
            ind[v] = w; success = 1; \
            terminate = 0; break; \
          ); \
        ); \
      ); \
    ); \
    if (success, break); \
    ); \
  ); \
  f; \
```

```
);
```

Listing 2: Computing a strategy from values in PARI/GP

Listing 3. Here, we implement (2.1) to find the value of a given vertex v.

```
/*  Pre-condition: MPG[2] is of the form
  [p, ..., p, -p, ..., -p] where p is in {-1, 1}
*/
solveVertex = ((MPG, v) -> \
  [A, eps] = MPG; \
  n = matsize(A)[1]; \
  G = underlyingGraph(A); \
  V1 = vector(n, w, vecsum(G[w,]) >= 1); \
  V2 = vector(n, w, vecsum(G[w,]) >= 2); \
  f = vector(n); \
  if (vecsum(V2) == 0, \
    for (w = 1, n, \
    if (V1[w], \
      for (x = 1, n, \
        if (G[w, x], f[w] = x); \
      ); \
    ); \
    ); \
    chi = posStr2Val(MPG, v, f); \
  ); \
  if (vecsum(V2) > 0, \
    for (v1 = 1, n, \
    if (V2[v1], \
      v2 = v1; break; \
    ); \
    ); \
    E = vector(n); \
    for (w = 1, n, \
    if (G[v2, w], \
      E0 = A; \
      for (x = 1, n, \
      if (x != w && G[v2, x], \
        E0[v2, x] = if(eps[v2] == 1, -oo, +oo); \
      ); \
      ); \
      E[w] = E0; \
    ); \
```

```
    ); \
    if (eps[v2] == 1, \
      chiVec = vector(n, w, if(G[v2, w], \
        solveVertex([E[w], eps], v)[1], -oo)); \
      chi = vecmax(chiVec); \
    ); \
    if (eps[v2] == -1, \
      chiVec = vector(n, w, if(G[v2, w], \
        solveVertex([E[w], eps], v)[1], +oo)); \
      chi = vecmin(chiVec); \
    ); \
  ); \
  [chi, G]; \
);
```

Listing 3: Finding the value of a vertex using (2.1), in PARI/GP

Listing 4. Next, we implement a function that computes the first derived mean payoff game, according to Definition 4.13.

```
derMPG = ((MPG, reduce) -> \
  [A, eps] = MPG; \
  n = matsize(A)[1]; \
  G = underlyingGraph(A); \
  deg = vector(n, v, vecsum(G[v,])); \
  f = finVals(MPG, n)[1]; \
  Ef = matrix(n, n, i, j, multInf(-eps[i], +oo)); \
  for (i = 1, n, \
    if (f[i] > 0, Ef[i, f[i]] = A[i, f[i]]); \
  ); \
  l = vector(n); \
  c = vector(n); \
  for (v = 1, n, \
    w = v; \
    while (deg[w] > 0 && l[v] < n, \
      w = f[w]; l[v]++; \
      if (v == w, c[v] = 1); \
    ); \
  ); \
  V1 = vector(n, v, c[v]); \
  EC1 = Ef; \
  for (v = 1, n, \
    for (w = 1, n, \
```

```
      if (!V1[v] || !V1[w], \
        EC1[v, w] = if(eps[v] == 1, -oo, +oo); \
      ); \
    ); \
  ); \
  L = vector(n); \
  m = vector(n); \
  for (v = 1, n, \
  if (V1[v], \
    w = f[v]; \
    L[v]++; \
    while (w != v, w = f[w]; L[v]++); \
    w = v; \
    for (i = 1, L[v], \
      w0 = w; w = f[w]; \
      m[v] += A[w0, w]; \
    ); \
    m[v] /= L[v]; \
  ); \
  ); \
  ER1 = matrix(n, n, v, w, if(eps[v] == 1, -oo, +oo)); \
  for (v = 1, n, for(v2 = 1, n, \
    goOn = 1; \
    w = v; \
    for (i = 0, L[v], \
      if (v2 == w, goOn = 0; break); \
      if (i < L[v], w = f[w]); \
    ); \
    if (goOn, \
      if (multInf(eps[v], m[v2]) >= multInf(eps[v], m[v]), \
        if (Reach(G, eps, vector(n, w, w == v2), eps[v])[v], \
          ER1[v, v2] = 0; \
        ); \
      ); \
    ); \
  ); \
  ); \
  E1 = matrix(n, n, v, w, \
    if(EC1[v, w] > -oo && EC1[v, w] < +oo,  EC1[v,w], \
    if(ER1[v, w] > -oo && ER1[v, w] < +oo, ER1[v,w], \
    if(eps[v] == 1, -oo, +oo)))); \
  eps1 = vector(n, v, if(V1[v], eps[v], 0)); \
```

```
66    if(reduce, reduceMPG([E1, eps1]), [E1, eps1]); \
67 )
```

Listing 4: Deriving mean payoff games in PARI/GP

B Auxiliary Functions in PARI/GP

Below, we give all auxiliary functions in PARI/GP that algorithms in Appendix A make use of.

Listing 5. Below, we implement the most basic auxiliary functions in PARI/GP.

```
addInf  = ((a, b) ->  if(a < +oo && b < +oo && a > -oo && b > -oo, \
            a + b, if (a < +oo && b < +oo, -oo, +oo)));
multInf = ((a, b) ->  if(a < +oo && b < +oo && a > -oo && b > -oo, \
            a * b,  if(a < 0, if(b < 0, +oo, -oo), \
            if(b < 0, -oo, +oo))));

isSubset = ((A, B) -> vector(#A, i, A[i] || B[i]) == B);

underlyingGraph = (A -> \
  n = matsize(A)[1]; \
  matrix(n, n, i, j, A[i, j] > -oo && A[i, j] < +oo); \
);

permuteMPG = ((MPG, perm) -> \
  [A, eps] = MPG; \
  n = matsize(A)[1]; \
  A2 = matrix(n, n, v, w, A[perm[v], perm[w]]); \
  eps2 = vector(n, v, eps[perm[v]]); \
  [A2, eps2]; \
); \

reduceMPG = (MPG -> \
  [A, eps] = MPG; \
  n = matsize(A)[1]; \
  V = []; \
  for (v = 1, n, if(eps[v] != 0, V = concat(V, v))); \
  A2 = vecextract(vecextract(A, V)~, V~)~; \
  eps2 = vecextract(eps, V); \
  [[A2, eps2], V]; \
);
```

Listing 5: The most basic auxiliary functions in PARI/GP

Listing 6. Below, we implement in PARI/GP an auxiliary algorithm that computes the values of a mean payoff game induced by a given positional strategy. On n vertex, the object `f` is a vector of length n such that `f[v] = w` if a vertex `v` as an integer in $[n]$ is mapped to a vertex `w`, and `f[v]` equals zero if and only if `v` represents a sink. We also include an algorithm that computes the value of just a single vertex `v`.

```
posStr2Vals = ((MPG, f) -> \
  [A, eps] = MPG; \
  n = matsize(A)[1]; \
  G = underlyingGraph(A); \
  V1 = vector(n, v, vecsum(G[v,]) >= 1); \
  l = vector(n); \
  chi = vector(n); \
  for (v = 1, n, \
    w = v; \
    while (V1[w] && l[v] < n, \
      w = f[w]; l[v]++; \
    ); \
  ); \
  for (v = 1, n, \
    if (l[v] < n, \
      w = v; \
      for (i = 1, l[v], \
        w = f[w]; \
      ); \
      chi[v] = if(eps[w] == 1, -oo, +oo); \
    ); \
    if (l[v] == n, \
      path = vector(n + 1); \
      w = v; \
      for (i = 1, n + 1, \
        path[i] = w; \
        w = f[w]; \
      ); \
      cycleSum = 0; \
      i = n + 1; \
      cycleSum += A[path[i-1], path[i]]; \
      i--; \
      while(path[i] != path[n + 1], \
        cycleSum += A[path[i-1], path[i]]; \
        i--; \
      ); \
      chi[v] = cycleSum / (n + 1 - i); \
    ); \
  ); \
  chi; \
);

```

```
posStr2Val = ((MPG, v, f) -> \
  [A, eps] = MPG; \
  n = matsize(A)[1]; \
  G = underlyingGraph(A); \
  V1 = vector(n, v, vecsum(G[v,]) >= 1); \
  l = 0; \
  w = v; \
  while (V1[w] && l < n, \
    w = f[w]; l++; \
  ); \
  if (l < n, \
    w = v; \
    for (i = 1, l, \
      w = f[w]; \
    ); \
    chi = if(eps[w] == 1, -oo, +oo); \
  ); \
  if (l == n, \
    path = vector(n + 1); \
    w = v; \
    for (i = 1, n + 1, \
      path[i] = w; \
      w = f[w]; \
    ); \
    cycleSum = 0; \
    i = n + 1; \
    cycleSum += A[path[i-1], path[i]]; \
    i--; \
    while(path[i] != path[n + 1], \
      cycleSum += A[path[i-1], path[i]]; \
      i--; \
    ); \
    chi = cycleSum / (n + 1 - i); \
  ); \
  chi; \
);
```

Listing 6: Deriving characteristic values from a positional strategy in PARI/GP

Listing 7. Here, we implement the finite values of mean payoff games as defined in Definition 2.13. The input is a mean payoff game and a positive integer `k`, and the algorithm outputs a pair `[f, et]`. Here, `et` is the vector of `k`-th finite values, and `f` is the unique vector of length equal to the number of vertices

such that `f[i]` is zero whenever `i` represents a sink, and `f[i]` is the smallest positive integer representing a vertex for which the optimization in computing the `k`-th finite value of `i` was attained. Along with it, we also implement a function `decide` that determines the signs of the characteristic values.

```
decide = (MPG -> \
  [A, eps] = MPG; \
  n = matsize(A)[1]; \
  G = underlyingGraph(A); \
  M = 0; \
  for (v = 1, n, \
    for (w = 1, n, \
    if (G[v, w], \
      M = max(M, abs(A[v, w])); \
    ); \
    ); \
  ); \
  k = 4*n^2*M+1; \
  vals = finVals(MPG, 4*n^2*M+1)[2]; \
  k = 4*n^2*M+1; \
  gam = vector(n); \
  for (v = 1, n, \
    if (vals[v] > 2 * n * M, gam[v] = 1); \
    if (vals[v] < -2 * n * M, gam[v] = -1); \
  ); \
  gam; \
);

finVals = ((MPG, k) -> \
  [A, eps] = MPG; \
  n = matsize(A)[1]; \
  G = underlyingGraph(A); \
  f = vector(n); \
  et = vector(n); \
  newet = vector(n); \
  while (k > 0, \
    for (v = 1, n, \
      locVals = []; \
      for (w = 1, n, \
        if(G[v, w], locVals = concat(locVals, addInf(A[v, w], et[w]))) ; \
      ); \
      if (locVals != [], \
```

```
38          newet[v] = if(eps[v] == 1, vecmax(locVals), vecmin(locVals));
    \
39        ); \
40        if (locVals == [], \
41          newet[v] = if(eps[v] == 1, -oo, +oo); \
42        ); \
43      ); \
44      if (k == 1, \
45        for (v = 1, n, \
46          for (w = 1, n, \
47          if (A[v, w] > -oo && A[v, w] < +oo, \
48            if (addInf(A[v, w], et[w]) == newet[v], \
49              f[v] = w; \
50              break; \
51            ); \
52          ); \
53          ); \
54        ); \
55      ); \
56      et = newet; \
57      k--; \
58    ); \
59    [f, et]; \
60  );
```

Listing 7: Deciding algorithm and finite values algorithm in PARI/GP

C Various mean payoff games in PARI/GP

Listing 8. Below, we entered various mean payoff games into PARI/GP, all in one file.

```
ZPmod = ((N, M) -> \
  n = 2*N + 1; \
  eps = vector(n, v, 1); \
  eps[1] = -1; \
  A = matrix(n, n, v, w, if(eps[v] == 1, -oo, +oo)); \
  for (v = 1, N, A[v, v+1] = M); \
  A[N+1, N+1] = 0; \
  A[1, N+2] = 0; \
  A[N+2, N+3] = 1; \
  for (v = N+3, 2*N, A[v, v+1] = 0); \
  A[2*N+1, N+2] = 0; \
  [A, eps]; \
);

fun = (M -> \
  [[-oo, 0, -oo; 1, -oo, -M; +oo, +oo, 0], [1,+1,-1]]; \
);

allValsAre0 = \
  [[-oo, -1, -oo, -oo, 0, -oo, -oo, -oo; \
  +oo, +oo, -1, +oo, +oo, +oo, +oo, +oo; \
  -oo, -oo, -oo, -1, -oo, -oo, 0, -oo; \
  -1, +oo, +oo, +oo, +oo, +oo, +oo, +oo; \
  0, +oo, +oo, +oo, +oo, 1, +oo, +oo; \
  -oo, -oo, -oo, -oo, -oo, -oo, 1, -oo; \
  +oo, +oo, 0, +oo, +oo, +oo, +oo, 1; \
  -oo, -oo, -oo, -oo, 1, -oo, -oo, -oo], \
  [1, -1, 1, -1, -1, 1, -1, 1]]; \

fourCycles = (M -> \
  [[-oo, 3, -oo, -oo, -oo, -oo; \
  0, +oo, +oo, M, +oo, +oo; \
  +oo, +oo, +oo, 1, +oo, +oo; \
  -oo, -oo, 0, -oo, -M, -oo; \
  +oo, +oo, +oo, +oo, +oo, 2; \
  -oo, -M, -oo, -oo, 0, -oo], \
  [1, -1, -1, 1, -1, 1]] \
); \
```

Listing 8: MPGs— Implementing various mean payoff games in PARI/GP

References

[1] S. Abbott. *Understanding Analysis*. Springer, second edition, 2015.

[2] M. Akian, S. Gaubert, O. Lorscheid, and M. Mnich. Mean payoff games with the signature of a potential. 2022. `https://oliver.impa.br/paper/meanpayoff.pdf`.

[3] S. Arora and B. Barak. *Computational Complexity*. Cambridge University Press, 2009.

[4] H. Björklund and S. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. *Discrete Applied Mathematics*, 155:210–229, 2007.

[5] K. Chatterjee, M. Henzinger, and A. Svozil. Faster algorithms for mean-payoff parity games. *arXiv:1706.06139v1*, 2017.

[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, third edition, 2009.

[7] L. Daviaud, M. Jurdziński, and R. Lazić. A pseudo-quasi-polynomial algorithm for mean-payoff parity games. *arXiv:1803.04756v3*, 2018.

[8] A. Ehrenfeucht and J. Mycielski. Positional strategies for mean payoff games. 8:109–113, 1979.

[9] V.A. Gurvich, A.V. Karzanov, and L.G. Khachivan. Cyclic games and an algorithm to find minimax cycle means in directed graphs. 28:85–91, 1988.

[10] R.M. Karp. A characterization of the minimum cycle mean in a digraph. 23:309–311, 1978.

[11] The PARI Group, Univ. Bordeaux. *PARI/GP version* `2.11.0`, 2018. Available from `http://pari.math.u-bordeaux.fr/`.

[12] U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. 158:343–359, 1996.