Quantitative Games with Interval Objectives*

Paul Hunter and Jean-François Raskin

Département d'Informatique, Université Libre de Bruxelles (U.L.B.)
{paul.hunter,jraskin}@ulb.ac.be

Abstract

Traditionally quantitative games such as mean-payoff games and discount sum games have two players – one trying to maximize the payoff, the other trying to minimize it. The associated decision problem, "Can Eve (the maximizer) achieve, for example, a positive payoff?" can be thought of as one player trying to attain a payoff in the interval $(0, \infty)$. In this paper we consider the more general problem of determining if a player can attain a payoff in a finite union of arbitrary intervals for various payoff functions (liminf/limsup, mean-payoff, discount sum, total sum). In particular this includes the interesting exact-value problem, "Can Eve achieve a payoff of exactly (e. g.) 0?"

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Quantitative games, Mean-payoff games, Discount sum games

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2014.365

1 Introduction

Quantitative two-player games on graphs have been extensively studied in the verification community [8, 6, 14, 10, 18]. Those models target applications in reactive system synthesis with resource constraints. In these games two players, Eve and Adam, interact by moving a token around a weighted, directed graph, for a possibly infinite number of moves. This interaction results in a play which is an infinite path in the graph. The value of the play is computed by applying a payoff function to the sequence of weights of the edges traversed along the path. Typical payoff functions are (lim)sup, (lim)inf, mean-payoff, (total) sum, and discounted sum.

In the literature is usual to assume that Eve is attempting to maximize the payoff and Adam is attempting to minimize it. In this context all these games are determined, that is the maximum that Eve can ensure is equal to the minimum that Adam can ensure, and this value can be computed in polynomial time for (lim)inf and (lim)sup [5], and in pseudo-polynomial time for mean-payoff, discounted sum, and total sum [18, 10]. The associated decision problem is the *threshold problem*: Given a game graph, a payoff function and a threshold ν does Eve have a strategy to ensure all consistent plays have payoff at least ν? The threshold problems for the aforementioned payoff functions are all closely related, and it is known that Eve and Adam can play optimally in those games with *memoryless strategies* [11]. Consequently the decision problem for all those games is in $\mathsf{NP} \cap \mathsf{coNP}$. In fact, it can be shown in $\mathsf{UP} \cap \mathsf{coUP}$ for mean-payoff, discounted sum, and total sum, and in PTIME for (lim)inf and (lim)sup.

The threshold problem can be seen as game in which Eve is trying to force the payoff to belong to the interval of values $[\nu, \infty)$. In this paper we consider the more general problem of determining if a player can attain a payoff in a finite union of arbitrary intervals for

* This work was supported by the ERC inVEST (279499) project.

Table 1 Complexity of deciding the winner in interval games.

Payoff type	Single interval	Multiple intervals	
		Binary	Unary
Liminf/limsup	PTIME	NP ∩ coNP	Parity game-c
Mean-payoff	NP ∩ coNP	PSPACE	Parity game-hard
Discounted sum (non-singleton)	PSPACE-c		PTIME
Discounted sum (exact value)	PSPACE-hard		?
Total sum	EXPSPACE-c		PSPACE-c

Table 2 Memory requirements for interval games.

Payoff type	Single interval (Eve/Adam)	Multiple intervals
Liminf/limsup	Positional	
Mean-payoff	Finite/Positional	Infinite
Discounted sum (non-singleton)	Finite	
Discounted sum (exact value)	Infinite	
Total sum	Finite/Infinite	Infinite

the classical payoff functions mentioned above. That is, we are interested in the following question: Given a weighted arena G and a finite union of real intervals, what is the complexity of determining if Eve has a winning strategy to ensure the payoff of any consistent play lies within the interval union? In particular this includes the interesting exact-value problem: Can Eve achieve a payoff of exactly ν? Such objectives arise when considering efficiency constraints, for example can a system achieve a certain payoff without exceeding a certain target? We consider two versions of our problem depending on whether the numeric inputs (weights, interval bounds and discount factor) are given in binary or unary. We also consider the memory requirements for a winning strategy both for Eve and Adam. Our games are a natural subclass of multi-dimensional quantitative games (see e. g. [6]), however our results are largely incomparable with that paper as we consider a wider array of payoff functions and our objective corresponds to *disjunctions* of multi-dimensional objectives which were not considered.

Tables 1 and 2 summarize the results of this paper: the first table highlights the complexity results and the second table highlights the memory requirements for playing optimally. While the classical threshold problems for weighted games can be solved in PTIME for (lim)inf and (lim)sup and in NP ∩ coNP for mean-payoff, discounted sum and total sum, and memoryless strategies always suffice, the situation for our interval objectives is far richer:

- For liminf and limsup, we provide a polynomial time algorithm in the case of a single interval. For a union of intervals, we show that these games are polynomially equivalent to parity games: so we can solve them in NP ∩ coNP, and a polynomial time algorithm for interval liminf games would provide a polynomial time algorithm for parity games (a long-standing open question in the area). Optimal strategies are memoryless for both players.
- For interval mean-payoff games, we provide a recursive algorithm that executes in polynomial space. This algorithm leads to a NP ∩ coNP algorithm in the case of single interval objectives. While mean-payoff games can be solved in polynomial time when weights are given in unary, we show here that interval mean-payoff games are at least as

hard as parity games even when weights are given in unary. So, a pseudo-polynomial time algorithm for interval mean-payoff games would lead to a polynomial algorithm for parity games. For a union of intervals, infinite memory may be necessary for both players, and for single interval exponential memory may be necessary for Eve while Adam can always play a memoryless strategy.

- Interval discounted sum games are complete for polynomial space when singleton intervals (and singleton gaps between intervals) are forbidden. The decidability for the case when singletons are allowed is left open and it generalizes known open problems in single player discounted sum graphs [7, 1]. Finite memory suffices for both players in the non-singleton case and infinite memory is needed for both players when singletons are allowed.
- For the total sum payoff, we establish a strong link with one counter parity games that leads to a PSPACE-complete result for unary encoding and an EXPSPACE-complete result for binary encoding. For single interval games Eve need only play finite memory strategies, while she may need infinite memory in the general case. In both cases, Adam may require infinite memory.

Structure of the paper. Section 2 introduces the necessary preliminaries. In Sections 3, 4, 5, and 6 we consider the decision problems and memory requirements for the liminf/limsup, mean-payoff, discounted sum, and total sum payoff functions, respectively. Due to space restrictions the full details of several proofs are left for the full version.

2 Preliminaries

A game graph is a tuple $G = (V, V_\exists, E, w, q_0)$ where (V, E, w) is an edge-weighted graph, $V_\exists \subseteq V$, and $q_0 \in V$ is the initial state. Without loss of generality we will assume all weights are integers. In the sequel we will depict vertices in $V_\exists$ with squares and vertices in $V \setminus V_\exists$ with circles. In complexity analyses we will denote the maximum absolute value of a weight in a game graph by W. If $V' \subseteq V$, we denote by $G \setminus V'$ the game graph induced by $V \setminus V'$.

A play in a game graph is an infinite sequence of states $\pi = v_0 v_1 \cdots$ where $v_0 = q_0$ and $(v_i, v_{i+1}) \in E$ for all i. Given a play $\pi = v_0 v_1 \cdots$ and integers k, l we define $\pi[k..l] = v_k \cdots v_l$, $\pi[..k] = \pi[0..k]$, and $\pi[l..] = v_l v_{l+1} \cdots$. We extend the weight function to partial plays by setting $w(\pi[k..l]) = \sum_{i=k}^{l-1} w((v_i, v_{i+1}))$. A strategy for Eve (Adam) is a function σ that maps partial plays ending with a vertex v in $V_\exists$ ($V \setminus V_\exists$) to a successor of v. A strategy has memory M if it can be realized as the output of a finite state machine with M states. A memoryless (or positional) strategy is a strategy with memory 1, that is, a function that only depends on the last element of the given partial play. A play $\pi = v_0 v_1 \cdots$ is consistent with a strategy σ for Eve (Adam) if whenever $v_i \in V_\exists$ ($v_i \in V \setminus V_\exists$), $\sigma(\pi[..i]) = v_{i+1}$.

2.1 Payoff functions

A play in a game graph defines an infinite sequence of weights. We define below several common functions that map such sequences to real numbers.

Liminf/limsup. The liminf (limsup) payoff is determined by the minimum (maximum) weight seen infinitely often. Given a play $\pi = v_0 v_1 \cdots$ we define:

$$\liminf(\pi) = \liminf_{i \to \infty} w(v_i, v_{i+1}) \qquad \limsup(\pi) = \limsup_{i \to \infty} w(v_i, v_{i+1}).$$

Note that by negating all weights and the endpoints of the intervals we transform a limsup game to a liminf game and vice-versa.

Mean-payoff. The *mean-payoff* value of a play is the limiting average weight, however there are several suitable definitions because the running averages might not converge. The mean-payoff values of a play π we are interested in are defined as:

$$\underline{MP}(\pi) = \liminf_{k\to\infty} \frac{1}{k} w(\pi[..k]) \qquad \overline{MP}(\pi) = \limsup_{k\to\infty} \frac{1}{k} w(\pi[..k]).$$

As with liminf/limsup games we can switch between definitions by negating weights and interval endpoints, so we will only consider the $\underline{MP}$ function.

Discounted sum. The *discounted sum* is defined by a discount factor $\lambda \in (0, 1)$. Given a play $\pi = v_0 v_1 \cdots$, we define:

$$DS_\lambda(\pi) = \sum_{i=0}^{\infty} \lambda^i \cdot w(v_i, v_{i+1}).$$

Total sum. The *total sum* condition can be thought of as a refinement of the mean-payoff condition, enabling discrimination between plays that have a mean-payoff of 0. Given a play π we define:

$$\underline{Total}(\pi) = \liminf_{k\to\infty} w(\pi[..k]) \qquad \overline{Total}(\pi) = \limsup_{k\to\infty} w(\pi[..k]).$$

As with liminf/limsup games we can switch between definitions by negating weights and interval endpoints, so we will only consider the $\underline{Total}$ function.

2.2 Interval games

For a fixed payoff function F, an *interval F game* consists of a finite game graph and a finite union of real intervals $I = I_1 \cup \cdots \cup I_r$ (given as a list of finitely presentable end-points). Given an interval F game (G, I), a play π in G is winning for Eve if $F(\pi) \in I$ and winning for Adam if $F(\pi) \notin I$. We say a player wins the interval game if he or she has a strategy σ such that all plays consistent with σ are winning for that player. For convenience we will assume the intervals are non-overlapping and ordered such that $\sup I_i \leq \inf I_{i+1}$ for all i.

2.3 Parity games

A parity game is a pair (G, Ω) where G is a game graph (with no weight function) and $\Omega : V \to \mathbb{N}$ is a function that assigns a priority to each vertex. Plays and strategies are defined as with interval games. A play defines an infinite sequence of priorities, and we say it is winning for Eve if and only if the minimal priority seen infinitely often is even.

3 Liminf games

The first payoff function we consider is the lim inf function. Note that as this always takes integer values, we can assume all intervals are closed or open as necessary. We show below that deciding interval liminf games is polynomially equivalent to deciding parity games. In particular the number of intervals is equal to the number of even priorities required, so single interval liminf games are equivalent to parity games with at most three priorities and can therefore be solved in polynomial time [15]. Further, the range of the priorities is determined by range of the weight function and vice versa, so this equivalence also holds for unary encoded interval liminf games.

▶ **Theorem 1.** *The following problems are polynomially equivalent:*
(i) *Deciding if Eve wins a unary encoded interval liminf game;*
(ii) *Deciding if Eve wins a binary encoded interval liminf game; and*
(iii) *Deciding if Eve wins a parity game.*

Proof. (i)⇒(ii): Trivial.

(ii)⇒(iii): For this reduction, we use the following function which will also be used in Section 6. Let $I = I_1 \cup I_2 \cup \cdots \cup I_r$ be a finite union of closed integer intervals such that $\sup I_i < \inf I_{i+1}$ for all i. Define $\Omega_I : \mathbb{Z} \to [1, 2r+1]$ as follows:

$$\Omega_I(n) = \begin{cases} 2i & \text{if } n \in I_i, \\ 1 & \text{if } n < \inf I_1, \text{ and} \\ \max\{1 + 2i : \sup I_i < n\} & \text{otherwise.} \end{cases}$$

Now suppose (G, I) is an interval liminf game. We transform the game graph G to G' as follows. Every edge e is sub-divided and the subdividing vertex is given priority $\Omega_I(w(e))$. The original vertices of G are all given priority $2r + 1$.

It is not difficult to see that there is a 1-1 correspondence between plays in G and plays in G', and that for any play in G, $\liminf w(e) \in I_i$ for some i if and only if the minimum priority in the corresponding play in G' seen infinitely often is even.

(iii)⇒(i): To go the other direction, given a parity game played on G we transform it to an interval liminf game played on G' as follows. G' is the weighted graph obtained by setting the weight of an edge to be the priority at the vertex at the tail of the edge (that is, the vertex for which the edge is outgoing). The intervals are singleton intervals containing each of the even priorities that occur in G. Clearly any play in G is a play in G' and it is not difficult to see that for a play in G the minimum priority seen infinitely often is even if and only if the $\liminf$ of the weights of all edges in a play of G' lie in a given interval. ◀

It follows from our reduction and the positional determinacy of parity games [17], that:

▶ **Corollary 2.** *Positional strategies suffice for interval liminf games.*

4 Mean-payoff games

In this section we investigate interval mean-payoff games. We give a recursive algorithm that repeatedly asks for a solution for the *mean-payoff threshold problem*. As mentioned earlier this problem is known to be in $\mathsf{NP} \cap \mathsf{coNP}$, and solvable in time $O(|V| \cdot |E| \cdot W)$ and space $O(|V| \cdot \log(|E| \cdot W))$ [4]. We denote this problem by $\mathbf{MP}_{\sim\nu}(G)$ where $\sim \in \{\geq, >, \leq, <\}$ depending on whether Eve is maximizing or minimizing[1] the payoff and whether or not a payoff of ν is winning for Eve. It is well known [8] that the strict threshold problem can be reduced to a non-strict threshold problem – this follows from the fact that mean-payoff values are restricted to a finite set of rationals.

Our algorithm implies that for a fixed number of intervals the problem reduces to the classic threshold problem (under polynomial-time Turing reductions). In the sequel we consider the memory requirements of single interval mean-payoff games in more detail. In particular we show that in this case finite memory strategies (indeed, positional strategies

[1] Note that by positional determinacy of mean-payoff games [8], $\underline{MP}$ and $\overline{MP}$ compute the same values when one player is maximizing the payoff. Hence our decision to use $\underline{MP}$ is not affected by whether Eve is maximizing or minimizing.

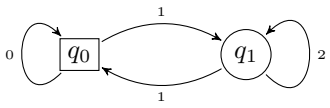

Figure 1 Interval mean-payoff game ($I = \{1\} \cup \{2\}$) which requires infinite memory.

for Adam) suffice for winning strategies. However, our first observation of this section is that in general, interval mean-payoff games may require infinite memory.

▶ **Lemma 3.** *Finite memory winning strategies are insufficient in interval mean-payoff games.*

Proof. Consider the game in Figure 1 where $I = \{1\} \cup \{2\}$. Eve has an infinite memory winning strategy in this game as follows. First she plays to q_1. Then she counts how many times Adam takes the loop (q_1, q_1). If Adam returns to q_0 then Eve takes the loop (q_0, q_0) the same number of times before returning to q_1. Clearly any play consistent with this strategy will either have $\underline{MP} = 2$ (if it eventually remains in q_1), or $\underline{MP} = 1$ (otherwise). Therefore the strategy is winning for Eve. Now suppose Eve plays a finite memory strategy σ with memory M. We observe that under this strategy Eve cannot stay at q_0 for more than M turns without staying there indefinitely. Adam's strategy is thus to stay at q_1 for $2M$ turns before returning to q_0. It is not difficult to see that any play consistent with this strategy and σ will yield a mean-payoff of either 0 or in the range $(1, 2)$, and so the strategy is winning for Adam. ◀

Upper bounds

We present an algorithm for computing the winning regions in an interval mean-payoff game in Algorithm 1. The correctness of the algorithm is given by Lemma 4.

Algorithm 1 $\mathbf{MP}_I(G)$

Input: A game graph $G = (V, V_\exists, E, w, q_0)$ and a finite union of real intervals I.
Output: $(X^\exists, X^\forall)$ where $X^\exists$ ($X^\forall$) are the vertices from which Eve (Adam) has a winning strategy.

if $I = \emptyset$ **then**
  return $(\emptyset, V)$
end if
$a \leftarrow \inf I$
if $a = -\infty$ **then**
  $(X, X') \leftarrow \mathbf{MP}_{\mathbb{R}\setminus I}(\overline{G})$ {$\overline{G}$ is G with $V_\exists$ and $V \setminus V_\exists$ swapped}
else
  $X \leftarrow \emptyset$
  repeat
    $(A, A') \leftarrow \mathbf{MP}_{\succ a}(G)$ {If $a \in I$ then $\succ = \geq$, otherwise $\succ = >$}
    $(B, B') \leftarrow \mathbf{MP}_{(-\infty, a] \cup I}(G)$
    $X \leftarrow X \cup A' \cup B'$
    $G \leftarrow G \setminus (A' \cup B')$
  until $A' \cup B' = \emptyset$
end if
return $(V \setminus X, X)$

▶ **Lemma 4.** *Let (G, I) be an interval mean-payoff game. $\boldsymbol{MP}_I(G)$ correctly computes the winning regions for Adam and Eve.*

Proof sketch. We observe that at each iteration of the loop we remove vertices from which Adam can either ensure the mean-payoff lies below I or from which he can (recursively) win on the larger set (with fewer interval end-points) $I' = (-\infty, \inf I] \cup I$. Thus it is clear that Adam has a winning strategy from any vertex removed. We now argue that Eve has a winning strategy on any vertex remaining. Observe that from these vertices she has two strategies: a positional strategy $\sigma_>$ which ensures $\underline{MP} \succ \inf I$ and $\sigma_<$ which wins with the larger objective I'. We combine these into a winning strategy for I as follows. Eve chooses any t in the first interval and keeps track of the current average weight. Whenever the average weight lies below t she plays $\sigma_>$, and whenever it lies above t she plays $\sigma_<$. As the mean-payoff objective is a tail objective and therefore independent of an initial play prefix, it is not difficult to see that if she eventually plays only one strategy then the mean-payoff will lie in I. If she changes strategy infinitely often, then the fact that $\sigma_>$ is positional means she will never deviate too far below $\inf I$ so the $\liminf$ average will lie between $\inf I$ and t, and thus in I. ◀

The running time for Algorithm 1 is $|V|^{2r-1} \cdot \mathbf{MP}$, where $\mathbf{MP}$ is the running time for an algorithm to solve the mean-payoff threshold problem. It is straightforward to see that the algorithm can be implemented in polynomial space.

▶ **Theorem 5.** *Let G be a game graph and I a finite union of r real intervals. Whether Eve wins the interval mean-payoff game (G, I) can be decided in time $O(|V|^{2r} \cdot |E| \cdot W)$ and space $O(r \cdot |V| \cdot \log(|E| \cdot W))$.*

▶ **Corollary 6.** *Determining if Eve wins an interval mean-payoff game for a fixed number of intervals is in* NP ∩ coNP.

We observe that although the players may require infinite memory for a winning strategy, Algorithm 1 shows that a winning strategy can be succinctly represented by $2r$ positional sub-strategies. It is not clear that given such a certificate whether there exists an efficient algorithm for computing the winning region, however we believe that this is the case. By the symmetry of the roles of the players, such an algorithm would show that the interval mean-payoff game is both in NP and coNP.

▶ **Conjecture 1.** *Determining if Eve wins an interval mean-payoff game is in* NP ∩ coNP.

Lower bound

The above conjecture would hold if we could solve interval mean-payoff games with only polynomially many calls to the mean-payoff threshold problem. We now give a lower bound for the complexity of deciding interval mean-payoff games which suggests any such algorithm would yield quite remarkable results: we reduce parity games to interval mean-payoff games with small weights and interval bounds. In particular this implies that any pseudo-polynomial time algorithm (including polynomially many calls to the threshold problem) would yield a polynomial time algorithm for parity games.

▶ **Theorem 7.** *There is a polynomial time reduction from parity games to unary encoded interval mean-payoff games.*

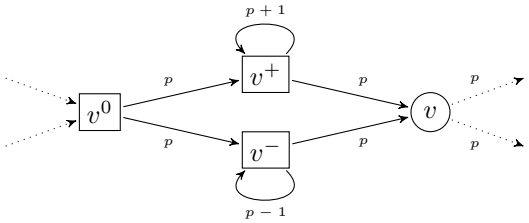

Figure 2 Vertex gadget for vertex $v \in V \setminus V_\exists$ with even priority p.

Proof sketch. Intuitively, we replace each vertex in the original game with the gadget shown in Figure 2. If the priority of the vertex is even then the gadget is controlled by Eve, and if it is odd then it is controlled by Adam. The last vertex in the gadget is controlled by the player that controlled the original vertex. Given a winning strategy in the parity game, Eve's corresponding strategy is to play the same on the original vertices, and whenever the play reaches a gadget corresponding to an even priority p she remains in the gadget until the current average lies in the interval $[p, p + \frac{1}{2}]$. This ensures that the minimum average seen infinitely often corresponds to the minimum priority seen infinitely often and so, with the interval set $I = \bigcup_{p \text{ even}} [p, p + 1)$, the strategy will be winning for Eve. Conversely, Adam can translate winning strategies from the parity game in the same manner. ◀

Memory requirements for single interval mean-payoff games

The strategies for Adam and Eve described in the proof of Lemma 4 require infinite memory. We now show, with a careful analysis, that in the case of a single interval this can be improved. Note that as we can replace any strict threshold call with a non-strict threshold we can assume without loss of generality that I is closed.

▶ **Theorem 8.** *Let (G, I) be a single interval mean-payoff game. If Adam has a winning strategy then he has a positional winning strategy. If Eve has a winning strategy then she has a strategy that requires finite memory.*

Proof sketch. The fact that Adam only requires positional strategies follows from [13] and the observation that in the single interval case the winning condition is convex.

The idea behind Eve's finite memory strategy on $X^\exists$ is to keep track of the total weight seen so far (rather than the average as in Lemma 4) *modulo cycles with average weight in I*. The intuition is that such cycles do not affect whether the overall mean-payoff will lie in the interval, so we can safely ignore them. As $\sigma_<$ and $\sigma_>$ are both positional in this case, by ignoring these cycles the total weight will never deviate too far from a central value, so finite memory suffices. ◀

5 Discount sum games

In this section we consider interval discount sum games. Here we make a distinction between whether or not singleton intervals (and singleton gaps between intervals) are permitted, because unlike other payoff functions considered in this paper there is a marked difference between the corresponding games. We show that for non-singleton intervals the problem of determining the winner is PSPACE-complete and as a consequence of our algorithm we show that finite memory strategies suffice. For singleton intervals (including the exact value problem) our PSPACE-hardness result holds, but is not even known if determining the winner is decidable.

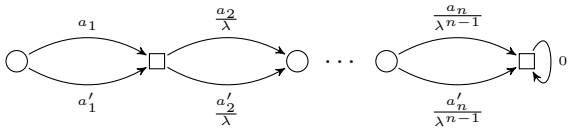

Figure 3 Reduction from subset sum games to interval discount sum games.

5.1 Single, non-singleton intervals

Lower bound

To show PSPACE-hardness we reduce from the subset sum game defined in [9]. A subset sum game is specified by a target $t \in \mathbb{N}$ and a list of pairs of natural numbers $(a_1, a'_1), \ldots, (a_n, a'_n)$. The game takes n rounds, in round i, one player (Adam if i is odd, Eve if i is even) chooses a_i or a'_i. After n rounds Eve wins if and only if the sum of the selected numbers is t. Given an instance of the subset sum game we construct the interval discount sum game shown in Figure 3, with interval $[t, t+1)$.

It is clear that a play in this game corresponds to a selection of elements from the pairs, and the discounted sum of the play is equal to the sum of the corresponding elements. As this sum is always an integer, the discounted sum lies in the interval $[t, t+1)$ if and only if the selected sum is equal to t.

Upper bound

Given $v \in V$ and strategies σ and τ for Eve and Adam respectively, we define $\mathrm{v}^v_{\sigma\tau}$ to be the payoff of the unique play from v consistent with σ and τ. Two important (memoryless) strategies for Eve are $\sigma_{\max}$ and $\sigma_{\min}$, the strategies which, for all states v, maximize $\min_\tau \mathrm{v}^v_{\sigma\tau}$ and minimize $\max_\tau \mathrm{v}^v_{\sigma\tau}$ respectively.

The idea behind the upper bound centres around the observation that after sufficiently many steps the remainder of any play does not contribute much to the overall discounted sum. If the target interval is non-singleton then the problem reduces to the classical threshold problem. Thus we can stop the game after finitely many steps when it becomes a trivial matter to determine if the overall discounted sum will lie in the interval or not. The key lemma for the result, the proof of which we defer to the full paper, is the following:

▶ **Lemma 9.** *Let (G, I, λ) be an interval discount sum game that Eve wins, and let*

$$N = \left\lfloor \frac{\log(\sup I - \inf I) + \log(1-\lambda) - \log(2W)}{\log \lambda} \right\rfloor.$$

Then Eve has a winning strategy that agrees with either $\sigma_{\max}$ or $\sigma_{\min}$ after N steps.

Note that whether the strategy agrees with $\sigma_{\max}$ or $\sigma_{\min}$ depends on the play up to the N-th step. It is feasible that against one strategy of Adam this strategy will agree with $\sigma_{\max}$ but against another strategy it will agree with $\sigma_{\min}$.

▶ **Corollary 10.** *Finite memory strategies suffice in non-singleton interval discount sum games.*

The algorithm for determining the winner of a non-singleton interval discount sum game is straightforward. We run an alternating Turing Machine for N steps to guess an initial play.

Note that N is polynomial in the size of the input, so this can be done in PSPACE. Suppose the play ends in state v with the current discounted sum S. We compute the four values:

$$\overline{M} = \max_{\tau} \mathrm{v}^v_{\sigma_{\max}\tau} \quad \underline{M} = \min_{\tau} \mathrm{v}^v_{\sigma_{\max}\tau} \quad \overline{m} = \max_{\tau} \mathrm{v}^v_{\sigma_{\min}\tau} \quad \underline{m} = \min_{\tau} \mathrm{v}^v_{\sigma_{\min}\tau}.$$

These are computable in NP ∩ coNP using an algorithm that computes values in classic discount sum games [18]. Finally we check if either:

$$S + \lambda^{N+1} \cdot \underline{M} \in I \quad \text{and} \quad S + \lambda^{N+1} \cdot \overline{M} \in I, \text{ or}$$
$$S + \lambda^{N+1} \cdot \underline{m} \in I \quad \text{and} \quad S + \lambda^{N+1} \cdot \overline{m} \in I.$$

It is clear that one of the above conditions holds if and only if $\sigma_{\max}$ or $\sigma_{\min}$ is winning from the current position. Therefore, from Lemma 9, one of the above conditions holds if and only if Eve has a winning strategy.

▶ **Theorem 11.** *Let G be a game graph, $I \subseteq \mathbb{R}$ a non-singleton interval, and $\lambda \in (0,1)$. Deciding if Eve wins the interval discount sum game (G, I, λ) is* PSPACE*-complete.*

We observe that if the weights, interval bounds and discount factor are all encoded in unary then N is logarithmic in the size of the input and $\overline{M}$, $\underline{M}$, $\overline{m}$, and $\underline{m}$ can all be computed in polynomial time using a pseudo-polynomial time algorithm for discount sum games [18]. Thus the above algorithm runs in polynomial time.

▶ **Theorem 12.** *Let G be a game graph, $I \subseteq \mathbb{R}$ a non-singleton interval and $\lambda \in (0,1)$. Deciding if Eve wins the unary encoded interval discount sum game (G, I, λ) is in* P*.*

5.2 Multiple intervals

The algorithm of the previous section also applies to multiple intervals *as long as the gaps between the intervals are also non-singleton.* This follows from the observation that after sufficiently many steps the overall discount payoff will not deviate too far from the current value, so at that point the game reduces to the single interval case.

▶ **Theorem 13.** *Let G be a game graph, I a finite union of real intervals such that neither I nor $\mathbb{R} \setminus I$ contains singleton elements, and $\lambda \in (0,1)$. Deciding if Eve wins the interval discount sum game (G, I, λ) is* PSPACE*-complete.*

5.3 Singleton intervals

When the set of intervals (or their complement) include singleton intervals, the situation is more complicated. Following the same argument as the previous section, after sufficiently many steps the problem reduces to the exact value problem: Given a discount sum game (G, λ) and a target $t \in \mathbb{Q}$, does Eve have a strategy to ensure the discounted sum is exactly t?

It is currently open whether this problem is even decidable, however the PSPACE-hardness result from the previous section (using the interval $\{t\}$ rather than $[t, t+1)$) gives a lower-bound. The problem is related to the universality problem for discount sum automata [2], a well-known problem for which decidability remains open [1]. The problem was also studied for Markov Decision Processes and graphs (i.e. one-player games) in [7] where it was shown to be decidable for discount factors of the form $\lambda = \frac{1}{n}$, and that in general infinite memory is required for winning strategies.

▶ **Lemma 14** ([7])**.** *There exist exact value discount sum games for which an infinite memory is required for a winning strategy.*

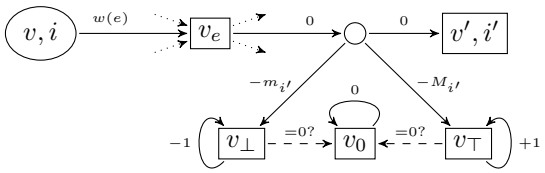

Figure 4 Edge gadget for edge $e = (v, v')$, $v \notin V_\exists$, $v' \in V_\exists$.

6 Total sum games

Total sum games refine mean-payoff games and can be seen as a special case of discount sum games where the discount factor is 1. Assuming the graph has integer weights, $\underline{Total}$ will always be an integer (or $\pm\infty$), thus we can assume all intervals are closed or open as necessary.

In this section we show that determining the winner of such games is PSPACE-complete for unary encoded games and EXPSPACE-complete for binary encoded games. Our bounds are obtained by relating interval total sum games with various one-counter games, that is, games played on the transition graph of a one-counter machine (equivalently, one-dimensional vector addition systems with states). Intuitively a one-counter game graph is a game graph augmented with a counter which is incremented or decremented by weights on traversed edges. A special set of edges are activated only if the counter has value 0. We give a reduction from one-counter reachability games, studied in [3, 14, 12] to establish lower bounds, and a reduction to one-counter parity games, studied in [16], for the upper bounds.

Lower bounds

We give a reduction from counter reachability games to (singleton) interval total sum games. This establishes the necessary lower bounds because determining the winner in such games was shown to be PSPACE-complete for unary encoded games in [14] and EXPSPACE-complete for binary encoded games in [12], even for the restricted one-counter game graphs considered here.

Given a one-counter reachability game with no zero-activated edges and target set $F \subseteq V_\exists$, we construct a game graph as follows. We double the weights of all edges; add a new initial vertex with an edge of weight $+1$ to the original initial vertex; and add a new global sink with a self loop of weight 0 and edges of weight -1 from all vertices in F. By parity arguments the new vertices are the only vertices where the total sum can be 0, and Eve has a strategy to reach the global sink if and only if she can reach F in the original game with counter value 0. Thus Eve wins the interval total sum game with interval $\{0\}$ if and only if she wins the original one-counter reachability game.

Upper bounds

We now show that interval total sum games can be solved in EXPSPACE by reducing them to parity games on infinite graphs defined by the transition graphs of one-counter machines. Such games were considered in [16], where a PSPACE algorithm was given for unary weighted one-counter games (equivalently, single alphabet pushdown processes). As a binary one-counter graph can be described by an exponentially larger unary one-counter graph, our reduction yields an EXPSPACE algorithm.

The key observation for the reduction is that interval total sum games can be viewed as parity games on $V \times \mathbb{Z}$, where the second component indicates the total sum so far. The

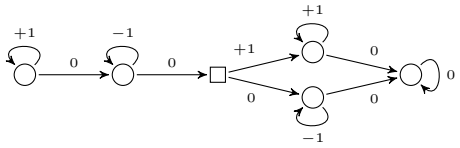

Figure 5 Interval total sum game ($I = \mathbb{R} \setminus \{0\}$) which requires infinite memory.

priority of a vertex (v, c) is determined by which interval (or gap between intervals) contains c, in the same manner used in the equivalence between liminf games and parity games in Section 3. However, we cannot use the result of [16] directly because for those games the priorities are defined by the states of the counter-machine and not the values of the counter. Instead, we have Eve assert which interval (or gap between intervals) the counter is in, and give Adam the ability to punish her if she claims falsely.

Let (G, I) be an interval total sum game. Recall from Section 3 the definition of Ω_I. Let us define $m_i := \min \Omega_I^{-1}(i)$ and $M_i := \max \Omega_I^{-1}(i)$. Intuitively, the reduction creates $2r + 1$ copies of the G (one for each interval and one for each gap), but we replace edges with the edge gadget shown in Figure 4. The priority of a vertex (v, i) is i, and for each gadget vertex it is $2r + 1$ except for v_0 which has priority $2r$.

We now show that Eve has a winning strategy in this parity game if and only if she has a winning strategy in the interval total sum game. We first observe that if the play reaches the predecessor of (v, i) and the counter value is outside $[m_i, M_i]$ then Adam can win by playing to $v_\perp$ if the counter is $< m_i$ or to $v_\top$ if the counter is $> M_i$. On the other hand, if the counter is in the range $[m_i, M_i]$ then Eve wins if Adam plays to either of these vertices. Thus the gadget defined by the vertices $\{v_\perp, v_\top, v_0\}$ allows Adam to punish Eve if and only if the counter is not in the asserted interval. Now, assuming Eve plays correctly, it is easy to see that the minimal priority seen infinitely often corresponds to the lowest interval or interval gap visited infinitely often by the counter. Thus Eve has a winning strategy in the parity game if and only if she has a winning strategy in the interval game.

▶ **Theorem 15.** *Deciding if Eve wins a binary (unary) encoded interval total sum game is* EXPSPACE*-complete (*PSPACE*-complete).*

Memory requirements

We now consider memory requirements for winning strategies in interval total sum games. Figure 5 shows that, in general, infinite memory is required for winning strategies.

▶ **Lemma 16.** *Finite memory winning strategies are insufficient in interval total sum games.*

By swapping the roles of the players and complementing the interval, we see that even for single interval games Adam may require infinite memory. We now show this is not the case for Eve. Indeed, using König's lemma, we can show a more general result.

▶ **Lemma 17.** *Let (G, I) be an interval total sum game where $I \cap \mathbb{Z}$ is finite. If Eve has a winning strategy then she has a finite memory winning strategy.*

To complete the argument for single interval total sum games, we observe that if the interval is infinite then we are considering the classical threshold problem for total sum games. Positional strategies for these games were shown to be sufficient in [11].

▶ **Theorem 18.** *Let (G, I) be a single interval total sum game. If Eve has a winning strategy then she has a finite memory winning strategy.*

References

1 U. Boker and T. A. Henzinger. Determinizing discounted-sum automata. In *CSL*, pages 82–96, 2011.
2 U. Boker and J. Otop. Personal communcation, 2014.
3 T. Brázdil, P. Jancar, and A. Kucera. Reachability games on extended vector addition systems with states. In *ICALP (2)*, pages 478–489, 2010.
4 L. Brim, J. Chaloupka, L. Doyen, R. Gentilini, and J.-F. Raskin. Faster algorithms for mean-payoff games. *Formal Methods in System Design*, 38(2):97–118, 2011.
5 K. Chatterjee, L. Doyen, and T. A. Henzinger. A survey of partial-observation stochastic parity games. *Formal Methods in System Design*, 43(2):268–284, 2013.
6 K. Chatterjee, L. Doyen, T. A. Henzinger, and J.-F. Raskin. Generalized mean-payoff and energy games. In *Proc. of FSTTCS*, pages 505–516, 2010.
7 K. Chatterjee, V. Forejt, and D. Wojtczak. Multi-objective discounted reward verification in graphs and mdps. In *LPAR*, pages 228–242, 2013.
8 A. Ehrenfeucht and J. Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8:109–113, 1979.
9 J. Fearnley and M. Jurdzinski. Reachability in two-clock timed automata is pspace-complete. In *ICALP*, volume 2, pages 212–223, 2013.
10 T. Gawlitza and H. Seidl. Games through nested fixpoints. In *CAV*, pages 291–305, 2009.
11 H. Gimbert and W. Zielonka. When can you play positionally? In *MFCS*, pages 686–697, 2004.
12 P. Hunter. Reachability in succinct one-counter games. Available at `http://arxiv.org/abs/1407.1996`, 2014.
13 E. Kopczynski. Omega-regular half-positional winning conditions. In *CSL*, pages 41–53, 2007.
14 J. Reichert. On the complexity of counter reachability games. In *RP*, pages 196–208, 2013.
15 S. Schewe. Solving parity games in big steps. In *FSTTCS*, pages 449–460, 2007.
16 O. Serre. Parity games played on transition graphs of one-counter processes. In *FoSSaCS*, pages 337–351, 2006.
17 W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200:135–183, 1998.
18 U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1):343–359, 1996.