Quantitative games with interval objectives

Paul Hunter

Jean-François Raskin

April 2014

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, 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?"

1 Introduction

Quantitative two-player games on graphs have been extensively studied in the verification community [6, 8, 10, 15, 19]. 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 [10, 19]. 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 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.

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	EXP-hard, EXPSPACE		PSPACE-c

Table 1: Complexity of deciding the winner in interval games

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

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

Table 2: Memory requirements for interval games

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 [1, 7]. 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 solution for the binary encoding together with an EXP-hardness result. 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 an infinite memory strategy.

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.

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 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 $\liminf$ 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 [16]. Further, the range of the priorities are 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. □

We observe that the above reductions between parity and liminf games do not significantly alter the topology of the game graph (if at all). In particular, positional strategies in one game readily translate to positional strategies in the other. It follows from the positional determinacy of parity games [18], that:

Corollary 1. *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*: Given a game graph G and a threshold $\nu \in \mathbb{Q}$ does Eve have a strategy to ensure the (liminf) mean-payoff of all consistent plays is at least[1] ν? 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 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 Section 4.3 we consider single interval mean-payoff games in more detail. In particular we show that in this case finite memory strategies (indeed, positional strategies 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 1. *Finite memory winning strategies are not sufficient in interval mean-payoff games.*

Proof. Consider the game in Figure 1 where $I = (0, 1] \cup [2, \infty)$. 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 that only visits q_0 finitely often will satisfy $\underline{MP} = 2$, and any play that visits q_0 infinitely often will satisfy $\underline{MP} = 1$. Therefore the strategy is winning for Eve. Now suppose Eve plays a finite memory strategy σ with memory M. We observe that any play consistent with σ that visits q_0 either remains in q_0 or exits q_0 in at most M steps – if a play stays in q_0 for more than M steps then a memory state must have been revisited, thus the strategy will keep the play in q_0 indefinitely. Consider the following (finite memory) strategy of Adam: whenever the play reaches q_1, take the loop (q_1, q_1) $M+1$ times then move to q_0. We claim this strategy is winning for Adam. If at some point the play consistent with σ and this strategy remains in q_0 indefinitely then it has $\underline{MP} = 0$, so it is winning for Adam. Otherwise the play exits q_0 infinitely often, that is the edge (q_0, q_1) is taken infinitely often. Let us break up the play into the segments defined by successive occurrences of this edge. Following the above argument the length of each of these segments is between $M + 3$ and $2M + 3$, and the weight of each of these segments is exactly $2M + 4$. Thus the average weight for each segment lies between $1 + \frac{1}{2M+3}$ and $2 - \frac{2}{M+3}$ inclusive. As M is fixed, it follows that $\underline{MP} \in (1, 2)$ and thus the play is winning for Adam. □

[1] or at most if she is minimizing the payoff

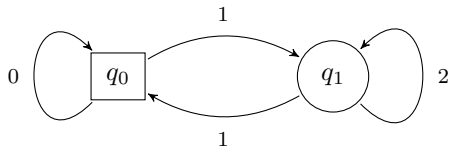

Figure 1: Interval mean-payoff game ($I = (0,1] \cup [2,\infty)$) which requires infinite memory

4.1 Upper bounds

We now present an algorithm, Algorithm 1, for computing the winning regions in an interval mean-payoff game.

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: $(W^\exists, W^\forall)$ where $W^\exists$ ($W^\forall$) are the vertices from which Eve (Adam) has a winning strategy.

```
if I = ∅ then
    return (∅, V)
end if
a ← inf I
if a = −∞ then
    (W, W′) ← MP_{ℝ\I}(Ḡ)                {Ḡ is G with V_∃ and V \ V_∃ swapped}
else
    W ← ∅
    repeat
        (A, A′) ← MP_{≻a}(G)              {If a ∈ I then ≻=≥ otherwise ≻=>}
        (B, B′) ← MP_{(−∞,a]∪I}(G)
        W ← W ∪ A′ ∪ B′
        G ← G \ (A′ ∪ B′)
    until A′ ∪ B′ = ∅
end if
return (V \ W, W)
```

The correctness of the algorithm is given by the following lemma.

Lemma 2. *Let (G, I) be an interval mean-payoff game. $\mathbf{MP}_I(G)$ correctly computes the winning regions for Adam and Eve.*

Proof. We observe that by symmetry the winning regions of $\mathbf{MP}_I(G)$ are precisely the complements of the winning regions of $\mathbf{MP}_{\mathbb{R}\setminus I}(G)$. Thus the algorithm correctly computes the winning regions for I if and only if correctly computes the winning regions for $\mathbb{R} \setminus I$. In particular we can assume that either $I = \emptyset$ or $\inf I > -\infty$.

The proof is by induction on the number of interval boundaries in I. If there are no boundaries then $I = \emptyset$ and so $\mathbf{MP}_I(G)$ returns the correct value: $(\emptyset, V)$. Now suppose $a = \inf I > -\infty$. Note that $I' = (-\infty, a] \cup I$ has one interval boundary fewer

than I, so by the induction hypothesis the recursive call in line 11 correctly computes the winning regions of G for the interval I'. Let W_i ($i = 0, 1, \ldots$) denote the set of vertices in W after i iterations. Note that the algorithm runs until $W_n = W_{n+1}$, and the subgraph of G used in the i-th iteration is $G \setminus W_{i-1}$. We prove by induction on i that Adam has a winning strategy from every vertex in W_i. For $i = 0$, $W_0 = \emptyset$ so the result holds trivially. Now suppose Adam has a winning strategy from every vertex in W_i, and let $v \in W_{i+1} \setminus W_i$. Either v is in the winning region of Adam for $\mathbf{MP}_{\succ a}(G \setminus W_i)$ or v is in the winning region of Adam for $\mathbf{MP}_{I'}(G \setminus W_i)$. In both cases the corresponding winning strategy will ensure a payoff outside I and will therefore be winning for plays restricted to $G \setminus W_i$. Thus his strategy from v is to play this strategy until a vertex in W_i is reached, whereupon he switches to the winning strategy from that vertex.

We now show that Eve has a winning strategy on the vertices in $V \setminus W$. Note that on these vertices Eve has two strategies: a memoryless strategy $\sigma_>$ which ensures $\underline{MP} \succ a$; and, by the inductive hypothesis, a strategy $\sigma_<$ which ensures a payoff in the interval I'. Also note that plays consistent with these strategies remain in $V \setminus W$. We now show how to combine these two strategies to obtain a winning strategy for the interval I. For simplicity we will assume $a \in I$, if it is not the case, then the same arguments apply by replacing a with the smallest payoff Adam can attain against $\sigma_>$. Let I_1 be the interval of I with $a = \inf I_1$, and let t be any element of I_1. The strategy for Eve is to track the current average weight of the play so far. If it is less than t then she plays $\sigma_>$ and if it is greater than or equal to t then she plays $\sigma_<$. Clearly if she changes strategy only finitely often then her strategy is winning: if she eventually only plays $\sigma_>$ then the payoff will be in $[a, t) \subseteq I_1 \subseteq I$; and if she eventually only plays $\sigma_<$ then the payoff will be in $[t, \infty) \cap I' \subseteq I$. Now suppose the play causes Eve to switch strategy infinitely often. The problem here is that when switching to $\sigma_>$ the average weight may go below a, and if this happens infinitely often the $\liminf$ average may be below a. However, as $\sigma_>$ is memoryless, the average after n steps will never be more than $\frac{(|V|+1)W}{n}$ below a: this is seen easiest by taking $a = 0$ and considering the total, rather than the average, weight. This tends to 0 as n tends to ∞ hence $\underline{MP}$ is at least a. As the average goes below t infinitely often, $\underline{MP} \leq t$. Therefore the payoff of the play is in $[a, t] \subseteq I_1 \subseteq I$, and hence the combined strategy is winning for Eve. □

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 2. *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 $O(r \cdot |V| \cdot \log(|E| \cdot W))$.*

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 whether Eve wins an interval mean-payoff game is in* NP∩ coNP.

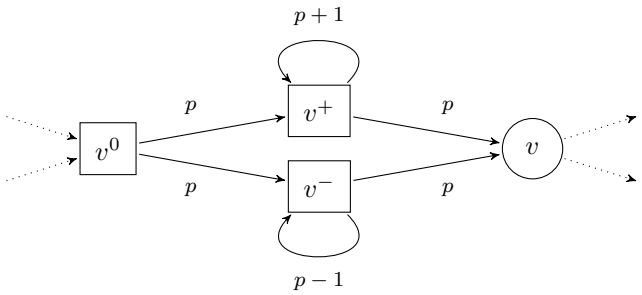

Figure 2: Vertex gadget for vertex $v \in V \setminus V_\exists$ with even priority p

4.2 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 small 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 3. *There is a polynomial time reduction from parity games to unary-encoded interval mean-payoff games.*

Proof. Let $(V, V_\exists, E, q_0, \Omega)$ be a (min-)parity game. Without loss of generality we can assume that the set of priorites is contained in $[0, |V|]$. We construct an interval mean-payoff game $(V', V'_\exists, E', w, q'_0, I)$ as follows.

- $I = [0, 1) \cup [2, 3) \cup \cdots \cup [n, n + 1)$ where n is the smallest even integer greater than or equal to $|V|$;
- $V' = V \cup V \times \{0, +, -\}$. For simplicity we write $(v, *)$ as v^*;
- $q'_0 = q_0$;
- $V'_\exists = V_\exists \cup \{v^0, v^+, v^- : \Omega(v) \text{ is even}\}$;
- E' and w are constructed as follows:
  - For each $(v, w) \in E$, $(v, w^0) \in E'$ and the weight of this edge is $\Omega(v)$,
  - For each $v \in V$: $(v^0, v^+), (v^0, v^-), (v^+, v), (v^-, v) \in E'$ all with weight $\Omega(v)$, $(v^+, v^+) \in E'$ with weight $\Omega(v)+1$, and $(v^-, v^-) \in E'$ with weight $\Omega(v) - 1$.

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.

As the weights and interval boundaries are integers in $[0, |V| + 1]$ this is clearly a polynomial time translation to a unary-encoded interval mean-payoff game. We claim that Eve wins the parity game if and only if she wins the interval mean-payoff game. Suppose she has a positional winning strategy σ in the parity game. We define her strategy σ' as follows. For any vertex $v \in V_\exists$ she moves to the vertex gadget corresponding to the vertex she would have moved to under σ. That is, $\sigma'(v) = (\sigma(v), 0)$. Whenever the play reaches a vertex gadget that she controls (i.e. a vertex v^0 where v has even priority p in the parity game), her strategy is to remain in the gadget until the average weight of the current play lies in the interval $[p, p + \frac{1}{2}]$. She does this by moving to v^+ if the current average is below the interval, and to v^- if the average is above, and then staying at that vertex until the average weight reaches the interval. Note that after sufficiently many steps this will always be possible. When the average weight lies in $[p, p + \frac{1}{2}]$ she moves to v and the game continues. There is a clear 1-1 correspondence between plays consistent with σ and plays consistent with σ', and if a play in the parity game visits a vertex with even priority p infinitely often, then the running average of the corresponding play will lie in the interval $[p, p + \frac{1}{2}] \subseteq I$ infinitely often. By construction, Adam can never reduce the mean-payoff below the interval $[p, p + \frac{1}{2}]$ unless the play reaches a gadget corresponding to a vertex of lower priority. This is important because we use the $\liminf$ definition of mean-payoff. Further, if he chooses to remain in a gadget indefinitely he will lose. As all plays consistent with σ have the property that the minimal priority visited infinitely often is even, it follows that for all plays π consistent with σ' there is some even priority p such that $\underline{MP}(\pi) \in [p, p + \frac{1}{2}] \subseteq I$. Thus the σ' is winning for Eve. For the converse we see that Adam can translate a winning strategy from the parity game in the same manner.

□

4.3 Single interval

We now examine in more detail the case when I is a single interval. As we can replace any strict threshold call with a non-strict threshold we can assume without loss of generality that I is closed. The simplification of Algorithm 1 to a single closed interval is given in Algorithm 2.

Algorithm 2 $\mathbf{MP}_{[a,b]}(G)$

Input: A game graph G and a bounded closed real interval $[a, b]$.
Output: $(W^\exists, W^\forall)$ where $W^\exists$ ($W^\forall$) are the vertices from which Eve (Adam) has a winning strategy.

```
W ← ∅
repeat
  (A, A') ← MP_{≥a}(G)
  (B, B') ← MP_{≤b}(G \ A')
  W ← W ∪ A' ∪ B'
  G ← G \ (A' ∪ B')
until A' ∪ B' = ∅
return (V \ W, W)
```

We observe that Algorithm 2 makes at most a linear number of calls to the mean-payoff threshold problem, so lies in the intersection of NP and coNP.

Theorem 4. *Deciding if Eve wins a single interval mean-payoff game is in* NP∩coNP.

4.3.1 Memory considerations

The strategies for Adam and Eve described in the proof of Lemma 2 require infinite memory. We now show, with a careful analysis, that in the case of a single interval this can be improved.

Theorem 5. *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. Algorithm 2 consists of repeatedly removing vertices from which Adam can either ensure the mean-payoff lies above or below I. Clearly Adam has a winning strategy from any vertex removed: he plays his (positional) winning strategy corresponding to the level at which the vertex was removed, until the play reaches a vertex removed at an earlier stage. We observe that any consistent play will never return to a vertex removed at a later stage (as such vertices are in the winning set for Eve at the same point of the iteration), so this strategy is in fact positional. Any play consistent with this strategy will eventually stabilize at some stage of the iteration, whereupon Adam's strategy for that stage will ensure the mean-payoff lies outside I. We also observe that this result follows from the fact that the objective is prefix-independent and convex, so from [14] Adam has a positional winning strategy.

The idea behind Eve's finite memory strategy on $W^{\exists}$ is to keep track of the total weight seen so far (rather than the average as in the proof of Lemma 2) *modulo cycles with average weight in I*. This ensures, with the strategy outlined below, that the total weight will remain within some bounded range, and hence the strategy will only require finite memory.

By subtracting a constant from the weights of all edges and the interval bounds, we can assume that $0 \in I$. We observe on the vertices in $W^{\exists}$ Eve has two (positional) strategies: $\sigma_<$ which ensures $\underline{MP} \leq \sup I$ and $\sigma_>$ which ensures $\underline{MP} \geq \inf I$. Eve's strategy is to alternate between these two strategies, as in the proof of Lemma 2, however now she changes when the following condition is met. We keep a stack-based history of the current play and when a cycle χ is completed we remove it from the history of the current play, keeping the first vertex of the cycle on the top of the stack. If $w(\chi)/|\chi| \in I$ we say χ is *good* and she continues to play her current strategy. If $w(\chi)/|\chi| \notin I$, she adds $w(\chi)$ to a counter. Note that if she was playing $\sigma_<$ she would only subtract from the counter and if she was playing $\sigma_>$ then she would only add to the counter because $\sigma_<$ and $\sigma_>$ are winning positional strategies. She switches strategies if the counter changes sign. That is, if she was playing $\sigma_<$ and the counter value falls below 0 she switches to $\sigma_>$, and she switches to $\sigma_<$ if she was playing $\sigma_>$ and the counter value goes above 0. Clearly this strategy requires only exponential memory: Eve needs only to store at most $|V|$ vertices in the history and because $\sigma_>$ and $\sigma_<$ are positional the counter values are bounded by $\pm|V| \cdot W$. We claim that any play π consistent with this strategy has $\underline{MP}(\pi) \in I$.

Let π be a play consistent with the strategy. Let us consider the state of the strategy after k steps of the play. Let w_k be the total weight of all good cycles popped, and $l_k \leq k$ their total length. Let c_k denote the counter value. We observe that the stack contents being stored are always a finite prefix of π (when read from bottom to top), so we can define s_k, the weight of the stack, as the weight of the corresponding prefix. It is clear from the definition of the strategy that:

$$w(\pi[..k]) = w_k + c_k + s_k.$$

Also, $-|V| \cdot W \leq c_k, s_k \leq |V| \cdot W$, and $\frac{w_k}{l_k} \in I$. As $0 \in I$ we have $\inf I \leq 0 \leq \sup I$, so

$$\inf I \leq \frac{l_k(\inf I)}{k} \leq \frac{w_k}{k} \leq \frac{w_k}{l_k} \leq \sup I.$$

Therefore,

$$\begin{array}{rcll} \frac{w(\pi[..k])}{k} & \geq & \frac{-2|V| \cdot W}{k} + \inf I & \rightarrow \quad \inf I \text{ as } k \rightarrow \infty, \text{ and} \\ \frac{w(\pi[..k])}{k} & \leq & \frac{2|V| \cdot W}{k} + \sup I & \rightarrow \quad \sup I \text{ as } k \rightarrow \infty. \end{array}$$

Hence, as I is closed, $\underline{MP}(\pi) \in I$ as required. □

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 stategies 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. We give a simple example that shows that infinite memory is required for winning strategies in this case.

5.1 Single, non-singleton intervals

We show that the problem for discount sum games in this case is PSPACE-complete for any discount factor λ.

Lower bound. To show PSPACE-hardness we reduce from the subset sum game defined in [9]. The subset sum game is specified by a target $t \in \mathbb{N}$ and a list of pairs of natural numbers $(a_1, a'_1), (a_2, a'_2), \ldots, (a_n, a'_n)$. 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 following interval discount sum game (for discount factor λ):

- $V = \{v_1, v_2, \ldots, v_{n+1}\}$,
- $V_\exists = \{v_i : i \text{ is even}\}$,
- $q_0 = v_1$,
- E and w defined as follows:
  - For $1 \leq i \leq n$ there are two edges from v_i to v_{i+1}, one with weight $\frac{a_i}{\lambda^{i-1}}$ and one with weight $\frac{a'_i}{\lambda^{i-1}}$,
  - There is a loop with weight 0 on v_{n+1}.
- $I = (t-1, t+1)$

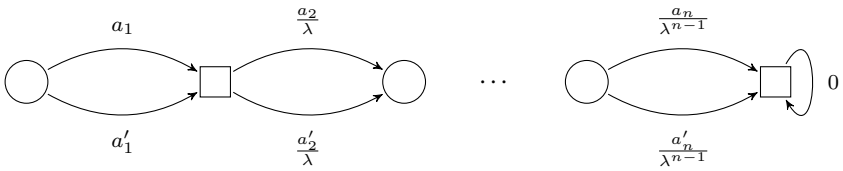

Figure 3: Reduction from subset sum games to interval discount sum games

The reduction is illustrated in Figure 3.

Note that as $\log\left(\frac{a}{\lambda^n}\right) = \log(a) - n \cdot \log(\lambda)$ the binary representations of the weights on this graph are still polynomial in the size of the input, so this is a polynomial time translation. 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 $(t-1, t+1)$ if and only if the sum is equal to t. Thus this is a polynomial time reduction from subset sum games to interval discounted sum games.

A corollary of this construction is that positional strategies are not sufficient for interval discount sum games.

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 many steps the remainder of any play does not contribute much to the overall discounted sum. If the target interval is non-singleton then after sufficiently many steps 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 is the following:

Lemma 3. *Suppose Eve has a winning strategy to ensure the discounted sum lies in an interval I, and let*

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

where W is the maximum absolute value of any weight occurring in G. 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 $\sigma_{\min}$.

Proof. We first observe that N is chosen such that for all $n > N$ we have

$$|I| > \lambda^n \cdot \left(\frac{2W}{1-\lambda}\right). \tag{1}$$

That is, after the N-th step of any play, the overall contribution of the remainder of the play is restricted to an interval smaller than I.

Let σ be a winning strategy for Eve. The desired winning strategy will follow σ for N steps and then one of $\sigma_{\max}$ or $\sigma_{\min}$ depending on the value of the play in a manner described presently. Suppose after N steps the current play has value $\mathtt{x}$ and is in state v. As σ is a winning strategy, we have for any strategy τ for Adam:

$$\mathtt{x} + \lambda^{N+1} \cdot \mathtt{v}^v_{\sigma\tau} \in I. \tag{2}$$

Now, as $|\mathtt{v}^v_{\sigma\tau}| \leq \frac{W}{1-\lambda}$, it follows from (1) and (2) that at least one of the following is true:

$$\mathtt{x} + \lambda^{N+1} \cdot \frac{W}{1-\lambda} \in I, \text{ or} \tag{3a}$$

$$\mathtt{x} - \lambda^{N+1} \cdot \frac{W}{1-\lambda} \in I. \tag{3b}$$

If (3a) holds then we follow $\sigma_{\max}$, otherwise we follow $\sigma_{\min}$. To show that the resulting strategy is winning, let us suppose (3a) holds, the case for (3b) being similar. From the definition of $\sigma_{\max}$ we have, for any state w and any strategy τ of Adam:

$$\mathtt{v}^w_{\sigma\tau} \leq \mathtt{v}^w_{\sigma_{\max}\tau} \leq \frac{W}{1-\lambda}.$$

Hence it follows from (2) that for any strategy τ of Adam:

$$\mathtt{x} + \lambda^{N+1} \cdot \mathtt{v}^v_{\sigma_{\max}\tau} \geq \mathtt{x} + \lambda^{N+1} \cdot \mathtt{v}^v_{\sigma\tau} \in I,$$

and from (3a):

$$\mathtt{x} + \lambda^{N+1} \cdot \mathtt{v}^v_{\sigma_{\max}\tau} \leq \mathtt{x} + \lambda^{N+1} \cdot \frac{W}{1-\lambda} \in I.$$

Thus the payoff of any play consistent with this strategy lies in I and is therefore winning for Eve. □

Corollary 2. *Finite memory strategies are sufficient 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 $\mathtt{x}$. We compute the four values:

$$\begin{array}{ll} \mathrm{maxmax} = \max_\tau \mathtt{v}^v_{\sigma_{\max}\tau} & \mathrm{minmax} = \min_\tau \mathtt{v}^v_{\sigma_{\max}\tau} \\ \mathrm{maxmin} = \max_\tau \mathtt{v}^v_{\sigma_{\min}\tau} & \mathrm{minmin} = \min_\tau \mathtt{v}^v_{\sigma_{\min}\tau}. \end{array}$$

These are computable in $\mathsf{NP} \cap \mathsf{coNP}$: minmax and maxmin using the standard algorithm for discount sum games, and maxmax (minmin) by fixing $\sigma_{\max}$ ($\sigma_{\min}$ respectively), computed in the previous step, and treating the resulting game as a solitaire discount sum game with Adam trying to maximize (minimize) the payoff. Finally we check if either:

$$\begin{array}{lll} \mathtt{x} + \lambda^{N+1} \cdot \mathrm{minmax} \in I & \text{and} & \mathtt{x} + \lambda^{N+1} \cdot \mathrm{maxmax} \in I, \text{ or} \\ \mathtt{x} + \lambda^{N+1} \cdot \mathrm{minmin} \in I & \text{and} & \mathtt{x} + \lambda^{N+1} \cdot \mathrm{maxmin} \in I. \end{array}$$

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 3, one of the above conditions holds if and only if Eve has a winning strategy.

Theorem 6. *Let G be a game graph, $I \subseteq \mathbb{R}$ a non-singleton real 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 maxmax, minmax, maxmin, and minmin can all be computed in polynomial time using a pseudo-polynomial time algorithm for the threshold problem for discount sum games (see e.g. [19]). Thus the above algorithm runs in polynomial time.

Theorem 7. *Let G be a game graph, $I \subseteq \mathbb{R}$ a non-singleton real interval and $\lambda \in (0,1)$ all encoded in unary. Deciding if Eve wins the interval discount sum game (G, I, λ) is in* PTIME*.*

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 8. *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 game graph G, a discount factor $\lambda \in \mathbb{Q}$ 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 $(t-1, t+1)$) 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.

Lemma 4 ([7])**.** *There exist exact value discount sum games for which an infinite memory is required for a winning strategy.*

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.

The objective of total sum games is similar to reachability in one-dimensional vector addition systems with states [3] and counter reachability games [15], however we

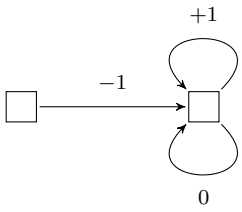

Figure 4: Exact value discount sum game ($\lambda = \frac{2}{3}$, $t = 0$) that requires infinite memory (modified from [7])

are interested in values seen infinitely often rather than reaching a particular state and counter value. The complexity bounds we obtain are similar to these problems, indeed we use the same problems for establishing the bounds. However it is not clear if there is a more direct reduction between these problems.

6.1 Lower bounds

In this section we establish the following result:

Theorem 9.
- *The problem of deciding if Eve wins an interval total sum game is* EXP*-hard.*
- *The problem of deciding if Eve wins a unary-encoded interval total sum game is* PSPACE*-hard.*

6.1.1 Binary encoding

We first show that deciding the winner of interval total sum games is EXP-hard by reducing from *countdown games*. A countdown game is played on a weighted graph, where all weights are negative. The play starts by setting a counter to a given initial value. Whenever an edge is taken the counter is decremented by the weight. Eve wins if and only if she reaches a vertex with the counter exactly 0. Deciding the winner of countdown games is known to be EXP-complete [13]. Note that by subdividing edges if necessary we can assume that the players play alternately, that is the graph is bipartite. The reduction is straightforward, given a countdown game G with initial credit c we construct the following total sum game. We add two new vertices (of Eve) v_I and $v_\perp$. There is an edge from v_I to the initial vertex of G with weight c, and an edge of weight 0 from every vertex of Eve in G to $v_\perp$. Also, for every edge $e = (v, v')$ where v is a vertex of Eve we add another edge $(v, v_\perp)$ of weight $w(e)$. Finally we have an edge $(v_\perp, v_\perp)$ of weight 0. Clearly Eve can ensure $\underline{Total} = 0$ if and only if she can reach $v_\perp$ with a total sum of 0. Thus she can win the interval total sum game, with interval $\{0\}$, if and only if she can win the countdown game.

6.1.2 Unary encoding

For unary-encoded interval total sum games, we reduce from the non-emptiness problem for one letter alphabet alternating automata, shown to be PSPACE-complete in [12]. Again, the reduction is simple as this problem can be viewed as a countdown game

where all edges have weight -1 and Eve has to guess the initial credit. The guessing stage can be implemented by having a loop on v_I with weight $+1$. The remainder of the reduction is as in the reduction from countdown games.

6.2 Upper bound

We now show that interval total sum games can be solved in EXPSPACE by reducing them to parity games on infinite graphs described by one-counter machines. Such games were studied in [17] where determining the winner was shown to be decidable in PSPACE, but the graphs were described by a *unary* counter machine, or equivalently, pushdown graphs with a single-letter alphabet. Here we use a definition corresponding to the use of a binary-valued counter (also called long-range in [15]). More formally, a one-counter game graph is described by a tuple $(V, V_\exists, E, E_0, w, q_0)$ where (V, E, w) is a finite weighted graph, $V_\exists \subseteq V$, $E_0 \subseteq V \times V$ and $q_0 \in V$. The (infinite) unweighted game graph corresponding to such a tuple is $(V \times \mathbb{Z}, V_\exists \times \mathbb{Z}, E', (q_0, 0))$ where E' is defined as follows:

- If $e = (v, v') \in E$ then for all $c \in \mathbb{Z}$, $\big((v, c), (v', c + w(e))\big) \in E'$, and
- If $(v, v') \in E_0$ then $\big((v, 0), (v', 0)\big) \in E'$.

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, E_0, are activated only if the counter has value 0. It is clear a binary one-counter graph can be described by an exponentially larger unary one-counter graph[2], hence 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 keeps track of the total sum seen so far. The 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 on parity games on one-counter graphs directly for this observation 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 $(V, V_\exists, E, w, q_0, I)$ 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 $M_i := \max \Omega_I^{-1}(i)$. We construct a parity game on a one-counter graph $(V', V'_\exists, E', E'_0, w', q'_0, \Omega)$ as follows.

- $V' = (V \times \{0, 1\}) \times [1, 2r + 1] \cup \{v_e : e \in E\} \cup \{v_0, v_\bot, v_\top\}$;
- $V'_\exists = E \cup \{v_0, v_\bot, v_\top\} \cup \{(v, 1, i) : v \in V_\exists \text{ and } i \in [1, 2r + 1]\}$;
- $q'_0 = (q_0, 1, \Omega_I(0))$;
- $E'_0 = \{(v_\bot, v_0), (v_\top, v_0)\}$;
- E' and w' given as follows, for all $i \in [1, 2r + 1]$:
  - For every $e = (v, v') \in E$, an edge from $(v, 1, i)$ to v_e with weight $w(e)$ and an edge from v_e to $(v', 0, i)$ with weight 0,

[2]We allow negative counter values, but this can be handled with non-negative counter values by doubling the state space

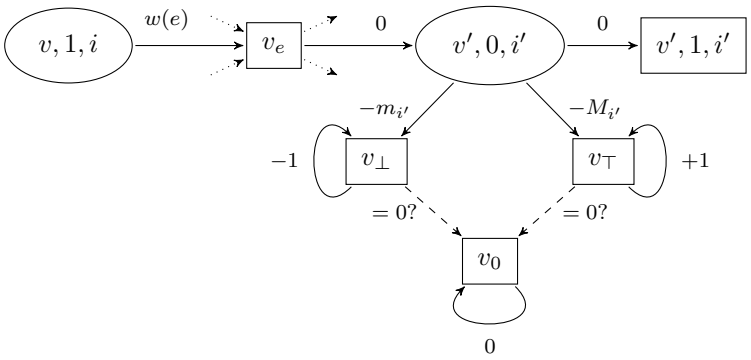

Figure 5: Edge gadget for edge $e = (v, v')$, $v \notin V_\exists$, $v' \in V_\exists$

  - An edge from $(v, 0, i)$ to $v_\perp$ with weight $-m_i$ if $m_i > -\infty$,
  - An edge from $(v, 0, i)$ to $v_\top$ with weight $-M_i$ if $M_i < \infty$,
  - An edge from $(v, 0, i)$ to $(v, 1, i)$ with weight 0, and
  - Loops on $v_\perp$, $v_\top$ and v_0 with weights -1, $+1$ and 0 respectively.
- $\Omega((v, 0, i)) = \Omega((v, 1, i)) = \Omega_I(i)$, $\Omega(v_e) = \Omega(v_\perp) = \Omega(v_\top) = 2r + 1$, and $\Omega(v_0) = 2r$.

Intuitively, we create $2r + 1$ copies of the game graph (one for each interval and one for each gap), but replace edges with the edge gadget shown in Figure 5.

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 $v_\perp$ ($v_\top$) is reached with a negative (positive) counter value then the edge to v_0 is never activated so the vertex acts as a sink which is winning for Adam. Conversely, if $v_\perp$ ($v_\top$) is reached with a non-negative (non-positive) counter value then the loop decrements (increments) the counter until the edge to v_0 is activated, whereupon Eve can win by moving to this sink which is winning for her. It follows that if the play reaches a vertex $(v, 0, i)$ 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 the counter is not in the asserted interval and lets Eve win if Adam attempts to falsely punish her. 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 10. *Deciding if Eve wins an interval total sum game is in* EXPSPACE.

We conclude by observing that if the interval game is encoded in unary, then the above reduction is a polynomial time reduction to the parity games on one-counter graphs considered in [17], giving an upper bound to match our lower bound.

Theorem 11. *Deciding if Eve wins a unary encoded interval total sum game is* PSPACE*-complete.*

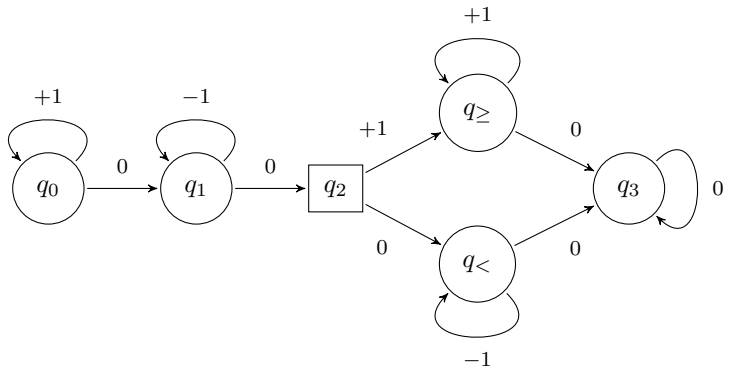

Figure 6: Interval total sum game ($I = \mathbb{R} \setminus \{0\}$) which requires infinite memory

6.3 Memory requirements

We now consider memory requirements for winning strategies in interval total sum games. We show that, in general, infinite memory is required for winning strategies, but for single interval games, winning strategies for Eve need only finite memory.

Lemma 5. *Finite memory winning strategies are not sufficient in interval total sum games.*

Proof. Consider the game in Figure 6 with the intervals $(-\infty, 0) \cup (0, \infty)$. Eve has a winning strategy in this game: if the play ever reaches q_2 then she moves to $q_{\geq}$ if the total sum is non-negative, and moves to $q_{<}$ otherwise. Clearly any play consistent with this strategy will have $\underline{Total} \neq 0$ so it is winning for Eve. Now suppose Eve plays a finite memory strategy. It follows there exists a memory state which cannot distinguish between two distinct sums at q_0. Therefore, after taking sufficiently many loops at q_1, it follows that there exists a memory state which cannot distinguish between two sums of different signs at q_2. As Eve's play depends only on her location and memory state, there is a total value for which Eve makes the "wrong choice", i.e. she either moves to $q_{\geq}$ with a negative sum or to $q_{<}$ with a non-negative sum. Adam's winning strategy is then to play to this move of Eve and then to increase or decrease the total sum to 0 before moving to q_3. □

By exchanging 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. In fact, we show that having unbounded intervals is necessary for Eve to not have a finite memory winning strategy.

Lemma 6. *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.*

Proof. As observed in the previous section, we can regard an interval total sum game as a parity game on $V \times \mathbb{Z}$. It is well known [18] that positional strategies suffice in parity games, even on infinite graphs. However, in our case such a strategy would depend on the current state *and on the counter value*, so it would not immediately be realizable with finite memory. We now show that if $I \cap \mathbb{Z}$ is finite and Eve has a winning strategy

then we only need to consider a bounded set of counter values so we can realize the strategy with finite memory. Let σ be a positional winning strategy for Eve on $V \times \mathbb{Z}$, and let $\overline{I} = [\inf I, \sup I]$. If σ is winning from (v, c) where $c \notin \overline{I}$ we claim she only requires finite memory to reach a state (v', c') from which σ is winning and where $c' \in \overline{I}$. Consider the finitely-branching, infinite tree of plays consistent with σ from (v, c)[3]. Let us cut a branch when it first reaches a vertex (v', c') with $c' \in \overline{I}$. Note that as we are following plays consistent with σ, such a state is in the winning set of σ. We claim the resulting tree is finite. If it were not, then by König's lemma there exists an infinite branch, that is, an infinite play consistent with σ that does not reach a vertex (v', c') with $c' \in \overline{I}$. As all even priority states are only of the form (v', c') where $c \in I$, such a play is winning for Adam, contradicting the fact that σ is a winning strategy for Eve. This finite tree then serves as the memory states for the strategy to reach $\overline{I}$ from (v, c). The finite memory strategy is now clear: if the current state is (v, c) with $c \in \overline{I}$ she moves to $\sigma(v, c)$. If the play ever reaches a state (v', c') with $c' \notin \overline{I}$ she plays her finite memory strategy until the play returns to (v'', c'') with $c'' \in \overline{I}$. As σ is positional, there are at most $|V| \times |\overline{I}|$ of these "out-of-bounds" states reachable (and possibly the initial state $(q_0, 0)$) so overall we only require finite memory. □

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 12. *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] Udi Boker and Thomas A. Henzinger. Determinizing discounted-sum automata. In *CSL*, pages 82–96, 2011.

[2] Udi Boker and Jan Otop. Personal communcation, 2014.

[3] Tomás Brázdil, Petr Jancar, and Antonín Kucera. Reachability games on extended vector addition systems with states. In *ICALP (2)*, pages 478–489, 2010.

[4] Lubos Brim, Jakub Chaloupka, Laurent Doyen, Raffaella Gentilini, and Jean-François Raskin. Faster algorithms for mean-payoff games. *Formal methods in system design*, 38(2):97–118, 2011.

[5] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. A survey of partial-observation stochastic parity games. *Formal Methods in System Design*, 43(2):268–284, 2013.

[6] Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. Generalized mean-payoff and energy games. In *Proc. of FSTTCS*, pages 505–516, 2010.

[7] Krishnendu Chatterjee, Vojtech Forejt, and Dominik Wojtczak. Multi-objective discounted reward verification in graphs and mdps. In *LPAR*, pages 228–242, 2013.

[3] That is, the tree rooted at (v, c) where the branches are all the plays consistent with σ and a branching occurs when Adam has a choice of moves

[8] A. Ehrenfeucht and J. Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8:109–113, 1979.

[9] John Fearnley and Marcin Jurdzinski. Reachability in two-clock timed automata is pspace-complete. In *ICALP*, volume 2, pages 212–223, 2013.

[10] Thomas Gawlitza and Helmut Seidl. Games through nested fixpoints. In *CAV*, pages 291–305, 2009.

[11] Hugo Gimbert and Wieslaw Zielonka. When can you play positionally? In *MFCS*, pages 686–697, 2004.

[12] Markus Holzer. On emptiness and counting for alternating finite automata. In *Developments in Language Theory*, pages 88–97, 1995.

[13] Marcin Jurdzinski, Jeremy Sproston, and François Laroussinie. Model checking probabilistic timed automata with one or two clocks. *Logical Methods in Computer Science*, 4(3), 2008.

[14] Eryk Kopczynski. Omega-regular half-positional winning conditions. In *CSL*, pages 41–53, 2007.

[15] Julien Reichert. On the complexity of counter reachability games. In *RP*, pages 196–208, 2013.

[16] Sven Schewe. Solving parity games in big steps. In *FSTTCS*, pages 449–460, 2007.

[17] Olivier Serre. Parity games played on transition graphs of one-counter processes. In *FoSSaCS*, pages 337–351, 2006.

[18] Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200:135–183, 1998.

[19] Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1):343–359, 1996.