Optimal Reachability in Divergent Weighted Timed Games

Damien Busatto-Gaston, Benjamin Monmege, Pierre-Alain Reynier

Optimal Reachability in Divergent Weighted Timed Games*

Damien Busatto-Gaston, Benjamin Monmege and Pierre-Alain Reynier

Aix Marseille Univ, LIF, CNRS, France
{damien.busatto,benjamin.monmege,pierre-alain.reynier}@lif.univ-mrs.fr

Abstract. Weighted timed games are played by two players on a timed automaton equipped with weights: one player wants to minimise the accumulated weight while reaching a target, while the other has an opposite objective. Used in a reactive synthesis perspective, this quantitative extension of timed games allows one to measure the quality of controllers. Weighted timed games are notoriously difficult and quickly undecidable, even when restricted to non-negative weights. Decidability results exist for subclasses of one-clock games, and for a subclass with non-negative weights defined by a semantical restriction on the weights of cycles. In this work, we introduce the class of *divergent weighted timed games* as a generalisation of this semantical restriction to arbitrary weights. We show how to compute their optimal value, yielding the first decidable class of weighted timed games with negative weights and an arbitrary number of clocks. In addition, we prove that divergence can be decided in polynomial space. Last, we prove that for untimed games, this restriction yields a class of games for which the value can be computed in polynomial time.

1 Introduction

Developing programs that verify real-time specifications is notoriously difficult, because such programs must take care of delicate timing issues, and are difficult to debug a posteriori. One research direction to ease the design of real-time software is to automatise the process. We model the situation into a timed game, played by a *controller* and an antagonistic *environment*: they act, in a turn-based fashion, over a *timed automaton* [2], namely a finite automaton equipped with real-valued variables, called clocks, evolving with a uniform rate. A usual objective for the controller is to reach a target. We are thus looking for a *strategy* of the controller, that is a recipe dictating how to play (timing delays and transitions to follow), so that the target is reached no matter how the environment plays. Reachability timed games are decidable [4], and EXPTIME-complete [21].

If the controller has a winning strategy in a given reachability timed game, several such winning strategies could exist. Weighted extensions of these games

* The first author has been supported by ENS Cachan, Université Paris-Saclay. This work has been funded by the DeLTA project (ANR-16-CE40-0007), and by the SoSI project (PEPS SISC CNRS).

have been considered in order to measure the quality of the winning strategy for the controller [9,1]. This means that the game now takes place over a *weighted (or priced) timed automaton* [5,3], where transitions are equipped with weights, and states with rates of weights (the cost is then proportional to the time spent in this state, with the rate as proportional coefficient). While solving weighted timed automata has been shown to be PSPACE-complete [6] (i.e. the same complexity as the non-weighted version), weighted timed games are known to be undecidable [12]. This has led to many restrictions in order to regain decidability, the first and most interesting one being the class of strictly non-Zeno cost with only non-negative weights (in transitions and states) [9,1]: this hypothesis states that every execution of the timed automaton that follows a cycle of the region automaton has a weight far from 0 (in interval $[1, +\infty)$, for instance).

Less is known for weighted timed games in the presence of negative weights in transitions and/or states. In particular, no results exist so far for a class that does not restrict the number of clocks of the timed automaton to 1. However, negative weights are particularly interesting from a modelling perspective, for instance in case weights represent the consumption level of a resource (money, energy...) with the possibility to spend and gain some resource. In this work, we introduce a generalisation of the strictly non-Zeno cost hypothesis in the presence of negative weights, that we call *divergence*. We show the decidability of the class of divergent weighted timed games, with a 2-EXPTIME complexity (and an EXPTIME-hardness lower bound). These complexity results match the ones that could be obtained in the non-negative case from the study of [9,1].

Other types of payoffs than the accumulated weight we study (i.e. total payoff) have been considered for weighted timed games. For instance, energy and mean-payoff timed games have been introduced in [11]. They are also undecidable in general. Interestingly, a subclass called *robust timed games*, not far from our divergence hypothesis, admits decidability results. A weighted timed game is robust if, to say short, every simple cycle (cycle without repetition of a state) has weight non-negative or less than a constant $-\varepsilon$. Solving robust timed game can be done in EXPSPACE, and is EXPTIME-hard. Moreover, deciding if a weighted timed game is robust has complexity 2-EXPSPACE (and coNEXPTIME-hard). In contrast, we show that deciding the divergence of a weighted timed game is a PSPACE-complete problem.[1] In terms of modeling power, we do believe that divergence is sufficient for most cases. It has to be noted that extending our techniques and results in the case of robust timed games is intrinsically not possible: indeed, the value problem for this class is undecidable [10].

The property of divergence is also interesting in the absence of time. Indeed, weighted games with reachability objectives have been recently explored as a refinement of mean-payoff games [14,15]. A pseudo-polynomial time (i.e. polynomial if weights are encoded in unary) procedure has been proposed to solve them, and they are at least as hard as mean-payoff games. In this article, we also study divergent weighted games, and show that they are the first

[1] Whereas all divergent weighted game are robust, the converse may not be true, since it is possible to mix positive and negative simple cycles in an SCC.

Table 1. Deciding weighted (timed) games with arbitrary weights

	Value of a game	Value of a divergent game	Deciding the divergence
Untimed	pseudo-poly. [15]	PTIME-complete	NL (unary), PTIME (binary)
Timed	Undecidable [12]	2-EXPTIME, EXPTIME-hard	PSPACE-complete

non-trivial class of weighted games with negative weights solvable in polynomial time. Table 1 summarises our results. We start in Sections 2 and 3 by studying weighted (untimed) games, before considering the timed setting in Sections 4 and 5. Complete proofs can be found in [17].

2 Weighted games

We start our study with untimed games. We consider two-player turn-based games played on weighted graphs and denote the two *players* by Max and Min. A *weighted game*[2] is a tuple $\mathcal{G} = \langle V = V_{\mathsf{Min}} \uplus V_{\mathsf{Max}}, V_t, A, E, \mathsf{Weight}\rangle$ where V are vertices, partitioned into vertices belonging to Min (V_{Min}) and Max (V_{Max}), $V_t \subseteq V_{\mathsf{Min}}$ is a subset of target vertices for player Min, A is an alphabet, $E \subseteq V \times A \times V$ is a set of directed edges, and $\mathsf{Weight}\colon E \to \mathbb{Z}$ is the weight function, associating an integer weight with each edge. These games need not be finite in general, but in Sections 2 and 3, we limit our study to the resolution of finite weighted games (where all previous sets are finite). We suppose that: (i) the game is deadlock-free, i.e. for each vertex $v \in V$, there is a letter $a \in A$ and a vertex $v' \in V$, such that $(v, a, v') \in E$; (ii) the game is deterministic, i.e. for each pair $(v, a) \in V \times A$, there is at most one vertex $v' \in V$ such that $(v, a, v') \in E$.[3]

A *finite play* is a finite sequence of edges $\rho = v_0 \xrightarrow{a_0} v_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{k-1}} v_k$, i.e. for all $0 \leqslant i < k$, $(v_i, a_i, v_{i+1}) \in E$. We denote by $|\rho|$ the length k of ρ. We often write $v_0 \xrightarrow{\rho} v_k$ to denote that ρ is a finite play from v_0 to v_k. The play ρ is said to be a *cycle* if $v_k = v_0$. We let $\mathsf{Plays}_{\mathcal{G}}$ be the set of all finite plays in $\mathcal{G}$, whereas $\mathsf{Plays}_{\mathcal{G}}^{\mathsf{Min}}$ and $\mathsf{Plays}_{\mathcal{G}}^{\mathsf{Max}}$ denote the finite plays that end in a vertex of Min and Max, respectively. A *play* is then an infinite sequence of consecutive edges.

A *strategy* for Min (respectively, Max) is a mapping $\sigma\colon \mathsf{Plays}_{\mathcal{G}}^{\mathsf{Min}} \to A$ (respectively, $\sigma\colon \mathsf{Plays}_{\mathcal{G}}^{\mathsf{Max}} \to A$) such that for all finite plays $\rho \in \mathsf{Plays}_{\mathcal{G}}^{\mathsf{Min}}$ (respectively, $\rho \in \mathsf{Plays}_{\mathcal{G}}^{\mathsf{Max}}$) ending in vertex v_k, there exists a vertex $v' \in V$ such that $(v_k, \sigma(\rho), v') \in E$. A play or finite play $\rho = v_0 \xrightarrow{a_0} v_1 \xrightarrow{a_1} \cdots$ conforms to a strategy σ of Min (respectively, Max) if for all k such that $v_k \in V_{\mathsf{Min}}$ (respectively, $v_k \in V_{\mathsf{Max}}$), we have that $a_k = \sigma(v_0 \xrightarrow{a_0} v_1 \cdots v_k)$. A strategy σ is *memoryless* if for all finite plays ρ, ρ' ending in the same vertex, we have that $\sigma(\rho) = \sigma(\rho')$. For all strategies σ_{Min} and σ_{Max} of players Min and Max, respectively, and for all vertices v, we let $\mathsf{Play}_{\mathcal{G}}(v, \sigma_{\mathsf{Max}}, \sigma_{\mathsf{Min}})$ be the outcome of σ_{Max} and σ_{Min}, defined as the unique play conforming to σ_{Max} and σ_{Min} and starting in v.

[2] Weighted games are called *min-cost reachability games* in [15].

[3] Actions are not standardly considered, but they become useful in the timed setting.

The objective of Min is to reach a target vertex, while minimising the accumulated weight up to the target. Hence, we associate to every finite play $\rho = v_0 \xrightarrow{a_0} v_1 \dots \xrightarrow{a_{k-1}} v_k$ its accumulated weight $\mathsf{Weight}_{\mathcal{G}}(\rho) = \sum_{i=0}^{k-1} \mathsf{Weight}(v_i, a_i, v_{i+1})$. Then, the weight of an infinite play $\rho = v_0 \xrightarrow{a_0} v_1 \xrightarrow{a_1} \cdots$, also denoted by $\mathsf{Weight}_{\mathcal{G}}(\rho)$, is defined by $+\infty$ if $v_k \notin V_t$ for all $k \geqslant 0$, or the weight of $v_0 \xrightarrow{a_0} v_1 \dots \xrightarrow{a_{k-1}} v_k$ if k is the first index such that $v_k \in V_t$. Then, we let $\mathsf{Val}_{\mathcal{G}}(v, \sigma_{\mathsf{Min}})$ and $\mathsf{Val}_{\mathcal{G}}(v, \sigma_{\mathsf{Max}})$ be the respective values of the strategies:

$$\mathsf{Val}_{\mathcal{G}}(v, \sigma_{\mathsf{Min}}) = \sup_{\sigma_{\mathsf{Max}}} \mathsf{Weight}_{\mathcal{G}}(\mathsf{Play}(v, \sigma_{\mathsf{Max}}, \sigma_{\mathsf{Min}}))$$
$$\mathsf{Val}_{\mathcal{G}}(v, \sigma_{\mathsf{Max}}) = \inf_{\sigma_{\mathsf{Min}}} \mathsf{Weight}_{\mathcal{G}}(\mathsf{Play}(v, \sigma_{\mathsf{Max}}, \sigma_{\mathsf{Min}})) \,.$$

Finally, for all vertices v, we let $\underline{\mathsf{Val}}_{\mathcal{G}}(v) = \sup_{\sigma_{\mathsf{Max}}} \mathsf{Val}_{\mathcal{G}}(v, \sigma_{\mathsf{Max}})$ and $\overline{\mathsf{Val}}_{\mathcal{G}}(v) = \inf_{\sigma_{\mathsf{Min}}} \mathsf{Val}_{\mathcal{G}}(v, \sigma_{\mathsf{Min}})$ be the *lower* and *upper values* of v, respectively. We may easily show that $\underline{\mathsf{Val}}_{\mathcal{G}}(v) \leqslant \overline{\mathsf{Val}}_{\mathcal{G}}(v)$ for all v. We say that strategies $\sigma^{\star}_{\mathsf{Min}}$ of Min and $\sigma^{\star}_{\mathsf{Max}}$ of Max are optimal if, for all vertices v, $\mathsf{Val}_{\mathcal{G}}(v, \sigma^{\star}_{\mathsf{Max}}) = \underline{\mathsf{Val}}_{\mathcal{G}}(v)$ and $\mathsf{Val}_{\mathcal{G}}(v, \sigma^{\star}_{\mathsf{Min}}) = \overline{\mathsf{Val}}_{\mathcal{G}}(v)$, respectively. We say that a game $\mathcal{G}$ is *determined* if for all vertices v, its lower and upper values are equal. In that case, we write $\mathsf{Val}_{\mathcal{G}}(v) = \underline{\mathsf{Val}}_{\mathcal{G}}(v) = \overline{\mathsf{Val}}_{\mathcal{G}}(v)$, and refer to it as the *value* of v in $\mathcal{G}$. Finite weighted games are known to be determined [15]. If the game is clear from the context, we may drop the index $\mathcal{G}$ from all previous notations.

Problems. We want to compute the value of a *finite* weighted game, as well as optimal strategies for both players, if they exist. The corresponding decision problem, called the *value problem*, asks whether $\mathsf{Val}_{\mathcal{G}}(v) \leqslant \alpha$, given a finite weighted game $\mathcal{G}$, one of its vertices v, and a threshold $\alpha \in \mathbb{Z} \cup \{-\infty, +\infty\}$.

Related work. The value problem is a generalisation of the classical shortest path problem in a weighted graph to the case of two-player games. If weights of edges are all non-negative, a generalised Dijkstra algorithm enables to solve it in polynomial time [22]. In the presence of negative weights, a pseudo-polynomial-time (i.e. polynomial with respect to the game where weights are stored in unary) solution has been given in [15], based on a fixed point computation with value iteration techniques. Moreover, the value problem with threshold $-\infty$ is shown to be in $\mathsf{NP} \cap \mathsf{coNP}$, and as hard as solving mean-payoff games.

3 Solving divergent weighted games

Our first contribution is to solve in polynomial time the value problem, for a subclass of finite weighted games that we call *divergent*. To the best of our knowledge, this is the first attempt to solve a non-trivial class of weighted games with arbitrary weights in polynomial time. Moreover, the same core technique is used for the decidability result in the timed setting that we will present in the next sections. Let us first define the class of divergent weighted games:

Definition 1. *A weighted game $\mathcal{G}$ is divergent when every cycle ρ of $\mathcal{G}$ satisfies* $\mathsf{Weight}(\rho) \neq 0$.

Divergence is a property of the underlying weighted graph, independent from the repartition of vertices between players. The term *divergent* reflects that cycling in the game ultimately makes the accumulated weight grow in absolute value. We will first formalise this intuition by analysing the strongly connected components (SCC) of the graph structure of a divergent game (the repartition of vertices into players does not matter for the SCC decomposition). Based on this analysis, we will obtain the following results:

Theorem 1. *The value problem over finite divergent weighted games is* PTIME*-complete. Moreover, deciding if a given finite weighted game is divergent is an* NL*-complete problem when weights are encoded in unary, and* PTIME *when they are encoded in binary.*

SCC analysis. A play ρ in $\mathcal{G}$ is said to be positive (respectively, negative) if $\mathsf{Weight}(\rho) > 0$ (respectively, $\mathsf{Weight}(\rho) < 0$). It follows that a cycle in a divergent weighted game is either positive or negative. A cycle is said to be simple if no vertices are visited twice (except for the common vertex at the beginning and the end of the cycle). We will rely on the following characterisation of divergent games in terms of SCCs.

Proposition 1. *A weighted game $\mathcal{G}$ is divergent if and only if, in each SCC of $\mathcal{G}$, all simple cycles are either all positive, or all negative.*

Proof. Let us first suppose that $\mathcal{G}$ is divergent. By contradiction, consider a negative simple cycle ρ (of weight $-p < 0$) and a positive simple cycle ρ' (of weight $p' > 0$) in the same SCC. Let v and v' be respectively the first vertices of ρ and ρ'. By strong connectivity, there exists a finite play η from v to v' and a finite play η' from v' to v. Let us consider the cycle ρ'' obtained as the concatenation of η and η'. If ρ'' has weight $q > 0$, the cycle obtained by concatenating q times ρ and p times ρ'' has weight 0, which contradicts the divergence of $\mathcal{G}$. The same reasoning on ρ'' and ρ' proves that ρ'' can not be negative. Thus, ρ'' is a cycle of weight 0, which again contradicts the hypothesis.

Reciprocally, consider a cycle of $\mathcal{G}$. It can be decomposed into simple cycles, all belonging to the same SCC. Therefore they are all positive or all negative. As the accumulated weight of the cycle is the sum of the weights of these simple cycles, $\mathcal{G}$ is divergent. □

Computing the values. Consider a divergent weighted game $\mathcal{G}$. Let us start by observing that vertices with value $+\infty$ are those from which Min can not reach the target vertices: thus, they can be computed with the classical attractor algorithm, and we can safely remove them, without changing other values or optimal strategies. In the rest, we therefore assume all values to be in $\mathbb{Z} \cup \{-\infty\}$.

Our computation of the values relies on a value iteration algorithm to find the greatest fixed point of operator $\mathcal{F} \colon (\mathbb{Z} \cup \{-\infty, +\infty\})^V \to (\mathbb{Z} \cup \{-\infty, +\infty\})^V$, defined for every vector $\boldsymbol{x}$ by $\mathcal{F}(\boldsymbol{x})_v = 0$ if $v \in V_t$, and otherwise

$$\mathcal{F}(\boldsymbol{x})_v = \begin{cases} \min\limits_{e=(v,a,v') \in E} \mathsf{Weight}(e) + \boldsymbol{x}_{v'} & \text{if } v \in V_{\mathsf{Min}} \\ \max\limits_{e=(v,a,v') \in E} \mathsf{Weight}(e) + \boldsymbol{x}_{v'} & \text{if } v \in V_{\mathsf{Max}}\,. \end{cases}$$

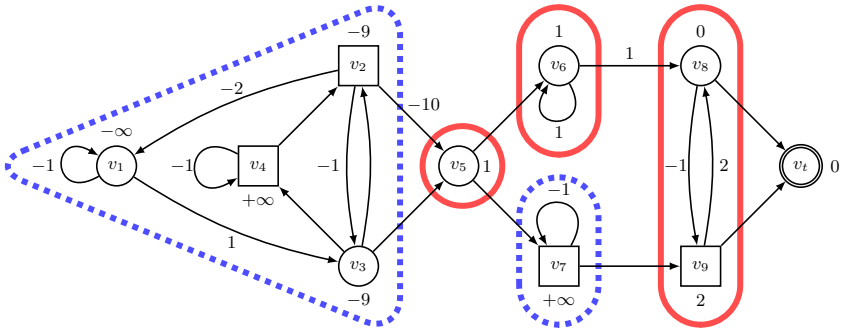

Fig. 1. SCC decomposition of a divergent weighted game: $\{v_1, v_2, v_3, v_4\}$ and $\{v_7\}$ are negative SCCs, $\{v_6\}$ and $\{v_8, v_9\}$ are positive SCCs, and $\{v_5\}$ is a trivial positive SCC.

Indeed, this greatest fixed point is known to be the vector of values of the game (see, e.g., [15, Corollary 11]). In [15], it is shown that, by initialising the iterative evaluation of $\mathcal{F}$ with the vector $\boldsymbol{x}^0$ mapping all vertices to $+\infty$, the computation terminates after a number of iterations pseudo-polynomial in $\mathcal{G}$ (i.e. polynomial in the number of vertices and the greatest weight in $\mathcal{G}$). For $i > 0$, we let $\boldsymbol{x}^i = \mathcal{F}(\boldsymbol{x}^{i-1})$. Notice that the sequence $(\boldsymbol{x}^i)_{i\in\mathbb{N}}$ is non-increasing, since $\mathcal{F}$ is a monotonic operator. Value iteration algorithms usually benefit from decomposing a game into SCCs (in polynomial time), considering them in a bottom-up fashion: starting with target vertices that have value 0, SCCs are then considered in inverse topological order since the values of vertices in an SCC only depend on values of vertices of greater SCCs (in topological order), that have been previously computed.

Example 1. Consider the weighted game of Fig. 1, where Min vertices are drawn with circles, and Max vertices with squares. Vertex v_t is the only target. Near each vertex is placed its value. For a given vector $\boldsymbol{x}$, we have $\mathcal{F}(\boldsymbol{x})_{v_8} = \min(0 + \boldsymbol{x}_{v_t}, -1 + \boldsymbol{x}_{v_9})$ and $\mathcal{F}(\boldsymbol{x})_{v_2} = \max(-2 + \boldsymbol{x}_{v_1}, -1 + \boldsymbol{x}_{v_3}, -10 + \boldsymbol{x}_{v_5})$. By a computation of the attractor of $\{v_t\}$ for Min, we obtain directly that v_4 and v_7 have value $+\infty$. The inverse topological order on SCCs prescribes then to compute first the values for the SCC $\{v_8, v_9\}$, with target vertex v_t associated with value 0. Then, we continue with SCC $\{v_6\}$, also keeping a new target vertex v_8 with (already computed) value 0. For the trivial SCC $\{v_5\}$, a single application of $\mathcal{F}$ suffices to compute the value. Finally, for the SCC $\{v_1, v_2, v_3, v_4\}$, we keep a new target vertex v_5 with value 1.[4] Notice that this game is divergent, since, in each SCC, all simple cycles have the same sign.

For a divergent game $\mathcal{G}$, Proposition 1 allows us to know in polynomial time if a given SCC is positive or negative, i.e. if all cycles it contains are positive

[4] This means that, in the definition of $\mathcal{F}$, a vertex v of V_t is indeed mapped to its previously computed value, not necessarily 0.

or negative, respectively: it suffices to consider an arbitrary cycle of it, and compute its weight. A trivial SCC (i.e. with a single vertex and no edges) will be arbitrarily considered positive. We now explain how to compute in polynomial time the value of all vertices in a positive or negative SCC.

First, in case of a positive SCC, we show that:

Proposition 2. *The value iteration algorithm applied on a positive SCC with n vertices stabilises after at most n steps.*

Proof (inspired by techniques used in [9]). Let $W = \max_{e \in E} |\mathsf{Weight}(e)|$ be the greatest weight in the game. There are no negative cycles in the SCC, thus there are no vertices with value $-\infty$ in the SCC, and all values are finite. Let K be an upper bound on the values $|\boldsymbol{x}_v^n|$ obtained after n steps of the algorithm.[5] Fix an integer $p > (2K + W(n-1))n$. We will show that the values obtained after $n+p$ steps are identical to those obtained after n steps only. Therefore, since the algorithm computes non-increasing sequences of values, we have indeed stabilised after n steps only. Assume the existence of a vertex v such that $\boldsymbol{x}_v^{n+p} < \boldsymbol{x}_v^n$. By induction on p, we can show the existence of a vertex v' and a finite play ρ from v to v' with length p and weight $\boldsymbol{x}_v^{n+p} - \boldsymbol{x}_{v'}^n$: the play is composed of the edges that optimise successively the min/max operator in $\mathcal{F}$. This finite play being of length greater than $(2K+W(n-1))n$, there is at least one vertex appearing more than $2K + W(n-1)$ times. Thus, it can be decomposed into at least $2K + W(n-1)$ cycles and a finite play ρ' visiting each vertex at most once. All cycles of the SCC being positive, the weight of ρ is at least $2K + W(n-1) - (n-1)W = 2K$, bounding from below the weight of ρ' by $-(n-1)W$. Then, $\boldsymbol{x}_v^{n+p} - \boldsymbol{x}_{v'}^n \geqslant 2K$, so $\boldsymbol{x}_v^{n+p} \geqslant 2K + \boldsymbol{x}_{v'}^n \geqslant K$. But $K \geqslant \boldsymbol{x}_v^n$, so $\boldsymbol{x}_v^{n+p} \geqslant \boldsymbol{x}_v^n$, and that is a contradiction. □

Example 2. For the SCC $\{v_8, v_9\}$ of the game in Fig. 1, starting from $\boldsymbol{x}$ mapping v_8 and v_9 to $+\infty$, and v_t to 0, after one iteration, $\boldsymbol{x}_{v_8}$ changes for value 0, and after the second iteration, $\boldsymbol{x}_{v_9}$ stabilises to value 2.

Consider then the case of a negative SCC. Contrary to the previous case, we must deal with vertices of value $-\infty$. However, in a negative SCC, those vertices are easy to find[6]. These are all vertices where Max can not unilaterally guarantee to reach a target vertex:

Proposition 3. *In a negative SCC with no vertices of value $+\infty$, vertices of value $-\infty$ are all the ones not in the attractor for* Max *to the targets.*

Proof. Consider a vertex v in the attractor for Max to the targets. Then, if Max applies a winning memoryless strategy for the reachability objective to the target

[5] After n steps, the value iteration algorithm has set to a finite value all vertices, since it extends the attractor computation.

[6] This is in contrast with the general case of (non divergent) finite weighted games where the problem of deciding if a vertex has value $-\infty$ is as hard as solving mean-payoff games [15].

vertices, all strategies of Min will generate a play from v reaching a target after at most $|V|$ steps. This implies that v has a finite (lower) value in the game.

Reciprocally, if v is not in the attractor, by determinacy of games with reachability objectives, Min has a (memoryless) strategy σ_{Min} to ensure that no strategy of Max permits to reach a target vertex from v. Applying σ_{Min} long enough to generate many negative cycles, before switching to a strategy allowing Min to reach the target (such a strategy exists since no vertex has value $+\infty$ in the game), allows Min to obtain from v a negative weight as small as possible. Thus, v has value $-\infty$. □

Thus, we can compute vertices of value $-\infty$ in polynomial time for a negative SCC. Then, finite values of other vertices can be computed in polynomial time with the following procedure. From a negative SCC $\mathcal{G}$ that has no more vertices of value $+\infty$ or $-\infty$, consider the dual (positive) SCC $\widetilde{\mathcal{G}}$ obtained by: (i) switching vertices of Min and Max; (ii) taking the opposite of every weight in edges. Sets of strategies of both players are exchanged in those two games, so that the upper value in $\mathcal{G}$ is equal to the opposite of the lower value in $\widetilde{\mathcal{G}}$, and vice versa. Since weighted games are determined, the value of $\mathcal{G}$ is the opposite of the value of $\widetilde{\mathcal{G}}$. Then, the value of $\mathcal{G}$ can be deduced from the value of $\widetilde{\mathcal{G}}$, for which Proposition 2 applies. We may also interpret this result as follows:

Proposition 4. *The value iteration algorithm, initialised with $\boldsymbol{x}_v^0 = -\infty$ (for all v), applied on a negative SCC with n vertices, and no vertices of value $+\infty$ or $-\infty$, stabilises after at most n steps.*

Proof. It is immediate that the vectors computed with this modified value iteration (that computes the smallest fixed point of $\mathcal{F}$) are exactly the opposite vectors of the ones computed in the dual positive SCC. The previous explanation is then a justification of the result. □

Example 3. Consider the SCC $\{v_1, v_2, v_3, v_4\}$ of the game in Fig. 1, where the value of vertex v_5 has been previously computed. We already know that v_4 has value $+\infty$ so we do not consider it further. The attractor of $\{v_5\}$ for Max is $\{v_2, v_3\}$, so that the value of v_1 is $-\infty$. Then, starting from $\boldsymbol{x}_0$ mapping v_2 and v_3 to $-\infty$, the value iteration algorithm computes this sequence of vectors: $\boldsymbol{x}_1 = (-9, -\infty)$ (Max tries to maximise the payoff, so he prefers to jump to the target to obtain $-10 + 1$ than going to v_3 where he gets $-1 - \infty$, while Min chooses v_2 to still guarantee $0 - \infty$), $\boldsymbol{x}_2 = (-9, -9)$ (now, Min has a choice between the target giving $0 + 1$ or v_3 giving $0 - 9$).

The proof for PTIME-hardness comes from a reduction (in logarithmic space) of the problem of solving finite games with reachability objectives [19]. To a reachability game, we simply add weights 1 on every transition, making it a divergent weighted game. Then, Min wins the reachability game if and only if the value in the weighted game is lower than $|V|$.

In a divergent weighted game where all values are finite, optimal strategies exist. As observed in [15], Max always has a memoryless optimal strategy,

whereas Min may require (finite) memory. Optimal strategies for both players can be obtained by combining optimal strategies in each SCC, the latter being obtained as explained in [15].

Class decision when weights are encoded in unary. We explain why deciding the divergence of a weighted game is an NL-complete problem. First, to prove the membership in NL, notice that a weighted game is *not divergent* if and only if there is a positive cycle and a negative cycle, both of length at most $|V|$, and belonging to the same SCC. To test this property in NL, we first guess a starting vertex for both cycles. Verifying that those are in the same SCC can be done in NL. Then, we guess the two cycles on-the-fly, keeping in memory their accumulated weights (smaller than $W \times |V|$, with W the biggest weight in the game, and thus of size at most logarithmic in the size of $\mathcal{G}$), and stop the on-the-fly exploration when the length of the cycles exceeds $|V|$. Therefore testing divergence is in $\mathsf{coNL} = \mathsf{NL}$ [20,25].

The NL-hardness (indeed coNL-hardness, which is equivalent [20,25]) is shown by a reduction of the reachability problem in a finite automaton. More precisely, we consider a finite automaton with a starting state and a different target state without outgoing transitions. We construct from it a weighted game by distributing all states to Min, and equipping all transitions with weight 1. We also add a loop with weight -1 on the target state and a transition from the target state to the initial state with weight 0. Then, the game is not divergent if and only if the target can be reached from the initial state in the automaton.

4 Weighted timed games

We now turn our attention to a timed extension of the weighted games. We will first define weighted timed games, giving their semantics in terms of *infinite* weighted games. We let X be a finite set of variables called clocks. A valuation of clocks is a mapping $\nu\colon X \to \mathbb{R}_{\geqslant 0}$. For a valuation ν, $d \in \mathbb{R}_{\geqslant 0}$ and $Y \subseteq X$, we define the valuation $\nu + d$ as $(\nu + d)(x) = \nu(x) + d$, for all $x \in X$, and the valuation $\nu[Y \leftarrow 0]$ as $(\nu[Y \leftarrow 0])(x) = 0$ if $x \in Y$, and $(\nu[Y \leftarrow 0])(x) = \nu(x)$ otherwise. The valuation $\mathbf{0}$ assigns 0 to every clock. A guard on clocks of X is a conjunction of atomic constraints of the form $x \bowtie c$, where $\bowtie \in \{\leqslant, <, =, >, \geqslant\}$ and $c \in \mathbb{N}$. A valuation $\nu\colon X \to \mathbb{R}_{\geqslant 0}$ satisfies an atomic constraint $x \bowtie c$ if $\nu(x) \bowtie c$. The satisfaction relation is extended to all guards g naturally, and denoted by $\nu \models g$. We let $G(X)$ the set of guards over X.

A weighted timed game is then a tuple $\mathcal{G} = \langle S = S_{\mathsf{Min}} \uplus S_{\mathsf{Max}}, S_t, \Delta, \mathsf{Weight}\rangle$ where S_{Min} and S_{Max} are *finite* disjoint subsets of states belonging to Min and Max, respectively, $S_t \subseteq S_{\mathsf{Min}}$ is a subset of target states for player Min, $\Delta \subseteq S \times G(X) \times 2^X \times S$ is a *finite* set of transitions, and $\mathsf{Weight}\colon \Delta \uplus S \to \mathbb{Z}$ is the weight function, associating an integer weight with each transition and state. Without loss of generality, we may suppose that for each state $s \in S$ and valuation ν, there exists a transition $(s, g, Y, s') \in \Delta$ such that $\nu \models g$.

The semantics of a weighted timed game $\mathcal{G}$ is defined in terms of the infinite weighted game $\mathcal{H}$ whose vertices are configurations of the weighted timed game.

A configuration is a pair (s, ν) with a state and a valuation of the clocks. Configurations are split into players according to the state. A configuration is final if its state is final. The alphabet of $\mathcal{H}$ is given by $\mathbb{R}_{\geqslant 0} \times \Delta$ and will encode the delay that a player wants to spend in the current state, before firing a certain transition. For every delay $d \in \mathbb{R}_{\geqslant 0}$, transition $\delta = (s, g, Y, s') \in \Delta$ and valuation ν, there is an edge $(s, \nu) \xrightarrow{d,\delta} (s', \nu')$ if $\nu + d \models g$ and $\nu' = (\nu + d)[Y \leftarrow 0]$. The weight of such an edge e is given by $d \times \mathsf{Weight}(s) + \mathsf{Weight}(\delta)$.

Plays, strategies, and values in the weighted timed game $\mathcal{G}$ are then defined as the ones in $\mathcal{H}$. It is known that weighted timed games are determined ($\underline{\mathsf{Val}}_{\mathcal{G}}(s, \nu) = \overline{\mathsf{Val}}_{\mathcal{G}}(s, \nu)$ for all state s and valuation ν).[7]

As usual in related work [1,9,10], we assume that all clocks are *bounded*, i.e. there is a constant $M \in \mathbb{N}$ such that every transition of the weighted timed games is equipped with a guard g such that $\nu \models g$ implies $\nu(x) \leqslant M$ for all clocks $x \in X$. We will rely on the crucial notion of regions, as introduced in the seminal work on timed automata [2]: a region is a set of valuations, that are all time-abstract bisimilar. There is only a finite number of regions and we denote by $\mathsf{Reg}(X, M)$ the set of regions associated with set of clocks X and maximal constant M in guards. For a valuation ν, we denote by $[\nu]$ the region that contains it. A region r' is said to be a time successor of region r if there exist $\nu \in r$, $\nu' \in r'$, and $d > 0$ such that $\nu' = \nu + d$. Moreover, for $Y \subseteq X$, we let $r[Y \leftarrow 0]$ be the region where clocks of Y are reset.

The region automaton $\mathcal{R}(\mathcal{G})$ of a game $\mathcal{G} = \langle S = S_{\mathsf{Min}} \uplus S_{\mathsf{Max}}, S_t, \Delta, \mathsf{Weight} \rangle$ is the finite automaton with states $S \times \mathsf{Reg}(X, M)$, alphabet Δ, and a transition $(s, r) \xrightarrow{\delta} (s', r')$ labelled by $\delta = (s, g, Y, s')$ if there exists a region r'' time successor of r such that r'' satisfies the guard g, and $r' = r''[Y \leftarrow 0]$. We call *path* an execution (not necessarily accepting) of this finite automaton, and we denote by π the paths. A play ρ in $\mathcal{G}$ is projected on a execution π in $\mathcal{R}(\mathcal{G})$, by replacing actual valuations by the regions containing them: we say that ρ *follows* path π. It is important to notice that, even if π is a cycle (i.e. starts and ends in the same state of the region automaton), there may exist plays following it in $\mathcal{G}$ that are not cycles, due to the fact that regions are sets of valuations.

Problems. As in weighted (untimed) games, we consider the *value problem*, mimicked from the one in $\mathcal{H}$. Precisely, given a weighted timed game $\mathcal{G}$, a configuration (s, ν) and a threshold $\alpha \in \mathbb{Z} \cup \{-\infty, +\infty\}$, we want to know whether $\mathsf{Val}_{\mathcal{G}}(s, \nu) \leqslant \alpha$. In the context of timed games, optimal strategies may not exist. We generally focus on ε-optimal strategies, that guarantee the optimal value, up to a small error ε.

Related work. In the one-player case, computing the optimal value and an ε-optimal strategy for weighted timed automata is known to be PSPACE-complete [6]. In the two-player case, much work for weighted timed games (also called priced timed games in the literature) has been achieved in the case of non-negative weights. In this setting, the value problem is undecidable [12,10]. To

[7] The result is stated in [13] for weighted timed games (called priced timed games) with one clock, but the proof does not use the assumption on the number of clocks.

obtain decidability, one possibility is to limit the number of clocks to 1: then, there is an exponential-time algorithm to compute the value as well as ε-optimal strategies [7,24,18], whereas the problem is only known to be PTIME-hard. The other possibility to obtain a decidability result [1,9] is to enforce a semantical property of divergence, originally called strictly non-Zeno cost: it asks that every play following a cycle in the region automaton has weight at least 1.

In the presence of negative weights, undecidability even holds for weighted timed games with only 2 clocks [16] (for the existence problem asking if a strategy of player Min can guarantee a given threshold). Only the 1-clock restriction has been studied so far allowing one to obtain an exponential-time algorithm, under restrictions on the resets of the clock in cycles [13]. For weighted timed games, the strictly non-Zeno cost property has only been defined and studied in the absence of negative weights [9]. As already mentioned in the introduction, the notion is close, but not equivalent, to the one of robust weighted timed games, studied for mean-payoff and energy objectives [11]. In the next section, we extend the strictly non-Zeno cost property to negative weights calling it the divergence property, in order to obtain decidability of a large class of multi-clocks weighted timed games in the presence of arbitrary weights.

5 Solving divergent weighted timed games

We introduce divergent weighted timed games, as an extension of divergent weighted games to the timed setting.

Definition 2. *A weighted timed game $\mathcal{G}$ is divergent when every finite play ρ in $\mathcal{G}$ following a cycle in the region automaton $\mathcal{R}(\mathcal{G})$ satisfies* $\mathsf{Weight}(\rho) \notin (-1, 1)$.[8]

The weight is not only supposed to be different from 0, but also far from 0: otherwise, the original intuition on the ultimate growing of the values of plays would not be fulfilled. If $\mathcal{G}$ has only non-negative weights on states and transitions, this definition matches with the *strictly non-Zeno cost* property of [9, Thm. 6]. Our contributions summarise as follows:

Theorem 2. *The value problem over divergent weighted timed games is decidable in* 2-EXPTIME*, and is* EXPTIME*-hard. Moreover, deciding if a given weighted timed game is divergent is a* PSPACE*-complete problem.*

Remember that these complexity results match the ones that can be obtained from the study of [9] for non-negative weights.

SCC analysis. Keeping the terminology of the untimed setting, a cycle π of $\mathcal{R}(\mathcal{G})$ is said to be positive (respectively, negative) if every play ρ following π satisfies $\mathsf{Weight}(\rho) \geqslant 1$ (respectively, $\mathsf{Weight}(\rho) \leqslant -1$). By definition, every cycle of the region automaton of a divergent weighted timed game is positive or

[8] As in [9], we could replace $(-1, 1)$ by $(-\kappa, \kappa)$ to define a notion of κ-divergence. However, since weights and guard constraints in weighted timed games are integers, for $\kappa \in (0, 1)$, a weighted timed game $\mathcal{G}$ is κ-divergent if and only if it is divergent.

negative. Moreover, notice that checking if a cycle π is positive or negative can be done in polynomial time with respect to the length of π. Indeed, the set $\{\mathsf{Weight}(\rho) \mid \rho \text{ is a play following } \pi\}$ is an interval, as the image of a convex set by an affine function (see [6, Sec. 3.2] for explanation), and the extremal points of this interval can be computed in polynomial time by solving a linear problem [6, Cor. 1]. We first transfer in the timed setting the characterisation of divergent games in terms of SCCs that we relied on in the untimed setting:

Proposition 5. *A weighted timed game $\mathcal{G}$ is divergent if and only if, in each SCC of $\mathcal{R}(\mathcal{G})$, simple cycles are either all positive, or all negative.*

The proof of the reciprocal follows the exact same reasoning as for weighted games (see Proposition 1). For the direct implication, the situation is more complex: we need to be more careful while composing cycles with each others, and weights in the timed game are no longer integers, forbidding the arithmetical reasoning we applied. To help us, we rely on the corner-point abstraction introduced in [8] to study multi-weighted timed automata. It consists in adding a weighted information to the edges $(s, r) \xrightarrow{\delta} (s', r')$ of the region automaton. Since the weights depend on the exact valuations ν and ν', taken in regions r and r', respectively, the weight of such an edge in the region automaton is computed for each pair of *corners* of the regions. Formally, corners of region r are valuations in $\overline{r} \cap \mathbb{N}^X$ (where $\overline{r}$ denotes the topological closure of r). Since corners do not necessarily belong to their regions, we must consider a modified version $\overline{\mathcal{G}}$ of the game $\mathcal{G}$ where all strict inequalities of guards have been replaced with non-strict ones. Then, for a path π in $\mathcal{R}(\mathcal{G})$, we denote by $\overline{\pi}$ the equivalent of path π in $\mathcal{R}(\overline{\mathcal{G}})$. In the following, our focus is on cycles of the region automaton, so we only need to consider the aggregation of all the behaviours following a cycle. Inspired by the *folded orbit graphs* (FOG) introduced in [23], we define the folded orbit graph $\mathsf{FOG}(\pi)$ of a cycle $\pi = (s_1, r = r_1) \xrightarrow{\delta_1} (s_2, r_2) \xrightarrow{\delta_2} \cdots \xrightarrow{\delta_n} (s_1, r)$ in $\mathcal{R}(\mathcal{G})$ as a graph whose vertices are corners of region r, and that contains an edge from corner v to corner v' if there exists a finite play $\overline{\rho}$ in $\overline{\mathcal{G}}$ from (s_1, v) to (s_1, v') following $\overline{\pi}$ jumping from corners to corners[9]. We fix such a finite play $\overline{\rho}$ arbitrarily and label the edge between v and v' in the FOG by this play: it is then denoted by $v \xrightarrow{\overline{\rho}} v'$. Moreover, since $\overline{\rho}$ jumps from corners to corners, its weight $\mathsf{Weight}(\overline{\rho})$ is an integer, conforming to the definitions of the corner-point abstraction of [8]. Following [8, Prop. 5], it is possible to find a play ρ in $\mathcal{G}$ close to $\overline{\rho}$, in the sense that we control the difference between their respective weights:

Lemma 1. *For all $\varepsilon > 0$ and edge $v \xrightarrow{\overline{\rho}} v'$ of $\mathsf{FOG}(\pi)$, there exists a play ρ in $\mathcal{G}$ following π such that $|\mathsf{Weight}(\rho) - \mathsf{Weight}(\overline{\rho})| \leqslant \varepsilon$.*

In order to prove the direct implication of Proposition 5, suppose now that $\mathcal{G}$ is divergent, and consider two simple cycles π and π' in the same SCC of $\mathcal{R}(\mathcal{G})$.

[9] Notice that if there is a play from (s_1, v) to (s_1, v') in $\overline{\mathcal{G}}$, there is another one that only jumps at corners of regions.

We need to show that they have the same sign. Lemma 2 will first take care of the case where π and π' share a state (s, r).

Lemma 2. *If $\mathcal{G}$ is divergent and two cycles π and π' of $\mathcal{R}(\mathcal{G})$ share a state (s, r), they are either both positive or both negative.*

Proof. Suppose by contradiction that π is negative and π' is positive. We assume that (s, r) is the first state of both π and π', possibly performing cyclic permutations of states if necessary. We construct a graph $\mathsf{FOG}(\pi, \pi')$ as the union of $\mathsf{FOG}(\pi)$ and $\mathsf{FOG}(\pi')$ (that share the same set of vertices), colouring in blue the edges of $\mathsf{FOG}(\pi)$ and in red the edges of $\mathsf{FOG}(\pi')$. A path in $\mathsf{FOG}(\pi, \pi')$ is said blue (respectively, red) when all of its edges are blue (respectively, red).

We assume first that there exists in $\mathsf{FOG}(\pi, \pi')$ a blue cycle C and a red cycle C' with the same first vertex v. Let k and k' be the respective lengths of C and C', so that C can be decomposed as $v \xrightarrow{\overline{\rho_1}} \cdots \xrightarrow{\overline{\rho_k}} v$ and C' as $v \xrightarrow{\overline{\rho'_1}} \cdots \xrightarrow{\overline{\rho'_{k'}}} v$, where $\overline{\rho_i}$ are plays following $\overline{\pi}$ and $\overline{\rho'_i}$ are plays following $\overline{\pi'}$, all jumping only on corners of regions. Let $\overline{\rho}$ be the concatenation of $\overline{\rho_1}, \ldots, \overline{\rho_k}$, and $\overline{\rho'}$ be the concatenation of $\overline{\rho'_1}, \ldots, \overline{\rho'_{k'}}$. Recall that $w = |\mathsf{Weight}(\overline{\rho})|$ and $w' = |\mathsf{Weight}(\overline{\rho'})|$ are integers. Since π is negative, so is π^k, the concatenation of k copies of π (the weight of a play following it is a sum of weights all below -1). Therefore, $\overline{\rho}$, that follows π^k, has a weight $\mathsf{Weight}(\overline{\rho}) \leqslant -1$. Similarly, $\mathsf{Weight}(\overline{\rho'}) \geqslant 1$. We consider the cycle C'' obtained by concatenating w' copies of C and w copies of C'. Similarly, we let $\overline{\rho''}$ be the play obtained by concatenating w' copies of $\overline{\rho}$ and w copies of $\overline{\rho'}$. By Lemma 1, there exists a play ρ'' in $\mathcal{G}$, following C'' such that $|\mathsf{Weight}(\rho'') - \mathsf{Weight}(\overline{\rho''})| \leqslant 1/3$. But $\mathsf{Weight}(\overline{\rho''}) = \mathsf{Weight}(\overline{\rho})w' + \mathsf{Weight}(\overline{\rho'})w = 0$, so $\mathsf{Weight}(\rho'') \in (-1, 1)$: this contradicts the divergence of $\mathcal{G}$, since ρ'' follows the cycle of $\mathcal{R}(\mathcal{G})$ composed of w' copies π^k and w copies of $\pi'^{k'}$ of $\mathcal{R}(\mathcal{G})$.

We now return to the general case, where C and C' may not exist. Since $\mathsf{FOG}(\pi)$ and $\mathsf{FOG}(\pi')$ are finite graphs with no deadlocks (every corner has an outgoing edge), from every corner of $\mathsf{FOG}(\pi, \pi')$, we can reach a blue simple cycle, as well as a red simple cycle. Since there are only a finite number of simple cycles in $\mathsf{FOG}(\pi, \pi')$, there exists a blue cycle C and a red cycle C' that can reach each other in $\mathsf{FOG}(\pi, \pi')$. In $\mathsf{FOG}(\pi, \pi')$, we let P be a path from the first vertex of C to the first vertex of C', and P' be a path from the first vertex of C' to the first vertex of C. Consider the cycle C'' obtained by concatenating P and P'. As a cycle of $\mathsf{FOG}(\pi, \pi')$, we can map it to a cycle π'' of $\mathcal{R}(\mathcal{G})$ (alternating π and π' depending on the colours of the traversed edges), so that C'' is a cycle (of length 1) of $\mathsf{FOG}(\pi'')$. By the divergence of $\mathcal{G}$, π'' is positive or negative. Suppose for instance that it is positive. Since (s, r) is the first state of both π and π'', we can construct the $\mathsf{FOG}(\pi, \pi'')$, in which C is a blue cycle and C'' is a red cycle, both sharing the same first vertex. We then conclude with the previous case. A similar reasoning with π' applies to the case that π'' is negative. Therefore, in all cases, we reached a contradiction. □

To finish the proof of the direct implication of Proposition 5, we suppose that the two simple cycles π and π' in the same SCC of $\mathcal{R}(\mathcal{G})$ do not share any

states. By strong connectivity, in $\mathcal{R}(\mathcal{G})$, there exists a path π_1 from the first state of π to the first state of π', and a path π_2 from the first state of π' to the first state of π. Consider the cycle of $\mathcal{R}(\mathcal{G})$ obtained by concatenating π_1 and π_2. By divergence of $\mathcal{G}$, it must be positive or negative. Since it shares a state with both π and π', Lemma 2 allows us to prove a contradiction in both cases. This concludes the proof of Proposition 5.

Value computation. We will now explain how to compute the values of a divergent weighted timed game $\mathcal{G}$. Remember that the function Val maps configurations of $S \times \mathbb{R}^X_{\geqslant 0}$ to a value in $\mathbb{R}_\infty = \mathbb{R} \cup \{-\infty, +\infty\}$. The semi-algorithm of [9] relies on the same principle as the value iteration algorithm used in the untimed setting, only this time we compute the greatest fixed point of operator $\mathcal{F} \colon \mathbb{R}_\infty^{S \times \mathbb{R}^X_{\geqslant 0}} \to \mathbb{R}_\infty^{S \times \mathbb{R}^X_{\geqslant 0}}$, defined by $\mathcal{F}(\boldsymbol{x})_{(s,\nu)} = 0$ if $s \in S_t$, and otherwise

$$\mathcal{F}(\boldsymbol{x})_{(s,\nu)} = \begin{cases} \sup\limits_{(s,\nu) \xrightarrow{d,\delta} (s',\nu')} d \times \mathsf{Weight}(s) + \mathsf{Weight}(\delta) + \boldsymbol{x}_{(s',\nu')} & \text{if } s \in S_{\mathsf{Max}} \\ \inf\limits_{(s,\nu) \xrightarrow{d,\delta} (s',\nu')} d \times \mathsf{Weight}(s) + \mathsf{Weight}(\delta) + \boldsymbol{x}_{(s',\nu')} & \text{if } s \in S_{\mathsf{Min}} \end{cases}$$

where $(s,\nu) \xrightarrow{d,\delta} (s',\nu')$ ranges over the edges of the infinite weighted game associated with $\mathcal{G}$ (the one defining its semantics). Then, starting from $\boldsymbol{x}^0$ mapping every configuration to $+\infty$, we let $\boldsymbol{x}^i = \mathcal{F}(\boldsymbol{x}^{i-1})$ for all $i > 0$. Since $\boldsymbol{x}^0$ is piecewise affine (even constant), and $\mathcal{F}$ preserves piecewise affinity, all iterates $\boldsymbol{x}^i$ are piecewise affine with a finite amount of pieces. In [1], it is proved that $\boldsymbol{x}^i$ has at most a number of pieces linear in the size of $\mathcal{R}(\mathcal{G})$ and exponential in i.[10]

First, we can compute the set of configurations having value $+\infty$. Indeed, the region automaton $\mathcal{R}(\mathcal{G})$ can be seen as a reachability two-player game $\mathcal{S}(\mathcal{G})$ by saying that (s,r) belongs to Min (Max, respectively) if $s \in S_{\mathsf{Min}}$ ($s \in S_{\mathsf{Max}}$, respectively). Notice that if $\mathsf{Val}(s,\nu) = +\infty$, then for all $\nu' \in [\nu], \mathsf{Val}(s,\nu') = +\infty$. Therefore, a configuration (s,ν) cannot reach the target states if and only if $(s,[\nu])$ is not in the attractor of Min to the targets in $\mathcal{S}(\mathcal{G})$. As a consequence, we can compute all such states of $\mathcal{S}(\mathcal{G})$ with complexity linear in the size of $\mathcal{R}(\mathcal{G})$.

We then decompose $\mathcal{R}(\mathcal{G})$ in SCCs. By Proposition 5, each SCC is either positive or negative (i.e. it contains only positive cycles, or only negative ones). Then, in order to find the sign of a component, it suffices to find one of its simple cycles, for example with a depth-first search, then compute the weight of one play following it.

As we did for weighted (untimed) games, we then compute values in inverse topological order over the SCCs. Once the values of all configurations in (s,r) appearing in previously considered SCCs have been computed, they are no longer modified in further computation. This is the case, in particular, for all pairs (s,r) that have value $+\infty$, that we precompute from the beginning. In order to resolve

[10] For divergent games with only non-negative weights, the fixed point is reached after a number of steps linear in the size of the region automaton [9]: overall, this leads to a doubly exponential complexity.

a positive SCC of $\mathcal{R}(\mathcal{G})$, we apply $\mathcal{F}$ on the current piecewise affine function, only modifying the pieces appearing in the SCC, until reaching a fixed point over these pieces. In order to resolve a negative SCC of $\mathcal{R}(\mathcal{G})$, we compute the attractor for Max to the previously computed SCCs: outside of this attractor, we set the value to $-\infty$. Then, we apply $\mathcal{F}$ for pieces appearing in the SCC, initialising them to $-\infty$ (equivalently, we compute in the dual game, that is a positive SCC), until reaching a fixed point over these pieces. The next proposition contains the correction and termination arguments that where presented in Propositions 2, 3, and 4 for the untimed setting:

Proposition 6. *Let $\mathcal{G}$ be a divergent game with no configurations of value $+\infty$.*

1. *The value iteration algorithm applied on a positive SCC of $\mathcal{R}(\mathcal{G})$ with n states stabilises after at most n steps.*
2. *In a negative SCC, states (s, r) of $\mathcal{R}(\mathcal{G})$ of value $-\infty$ are all the ones not in the attractor for Max to the targets.*
3. *The value iteration algorithm, initialised with $-\infty$, applied on a negative SCC of $\mathcal{R}(\mathcal{G})$ with n states, and no vertices of value $-\infty$, stabilises after at most n steps.*

By the complexity results of [1, Thm. 3], we obtain a doubly exponential time algorithm computing the value of a divergent weighted timed game. This shows that the value problem is in 2-$\mathsf{EXPTIME}$ for divergent weighted timed game. The proof for $\mathsf{EXPTIME}$-hardness comes from a reduction of the problem of solving timed games with reachability objectives [21]. To a reachability timed game, we simply add weights 1 on every transition and 0 on every state, making it a divergent weighted timed game. Then, Min wins the reachability timed game if and only if the value in the weighted timed game is lower than threshold $\alpha = |S| \times |\mathsf{Reg}(X, M)|$.

In an SCC of $\mathcal{R}(\mathcal{G})$, the value iteration algorithm of [1] allows us to compute an ε-optimal strategy for both players (for configurations having a finite value), that is constant (delay or fire a transition) over each piece of the piecewise affine value function. As in the untimed setting, we may then compose such ε-optimal strategies to obtain an ε'-optimal strategy in $\mathcal{G}$ (ε' is greater than ε, but can be controlled with respect to the number of SCCs in $\mathcal{R}(\mathcal{G})$).

Class decision. Deciding if a weighted timed game is divergent is PSPACE-complete. The proof is an extension of the untimed setting NL-complete result, but this time we reason on regions, hence the exponential blowup in complexity: it heavily relies on Proposition 5, as well as the corner-point abstraction to keep a compact representation of plays.

6 Conclusion

In this article, we introduced the first decidable class of weighted timed games with arbitrary weights, with no restrictions on the number of clocks. Future work include the approximation problem for a larger class of weighted timed games (divergent ones where we also allow cycles of weight exactly 0), already studied with only non-negative weights by [10].

References

1. Rajeev Alur, Mikhail Bernadsky, and P. Madhusudan. Optimal reachability for weighted timed games. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP'04)*, volume 3142 of *LNCS*, pages 122–133. Springer, 2004.
2. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
3. Rajeev Alur, Salvatore La Torre, and George J. Pappas. Optimal paths in weighted timed automata. *Theoretical Computer Science*, 318(3):297–322, 2004.
4. Eugene Asarin and Oded Maler. As soon as possible: Time optimal control for timed automata. In *Hybrid Systems: Computation and Control*, volume 1569 of *LNCS*, pages 19–30. Springer, 1999.
5. Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim G. Larsen, Judi Romijn, and Frits W. Vaandrager. Minimum-cost reachability for priced timed automata. In *Proceedings of the 4th International Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, volume 2034 of *LNCS*, pages 147–161. Springer, 2001.
6. Patricia Bouyer, Thomas Brihaye, Véronique Bruyère, and Jean-François Raskin. On the optimal reachability problem of weighted timed automata. *Formal Methods in System Design*, 31(2):135–175, 2007.
7. Patricia Bouyer, Thomas Brihaye, and Nicolas Markey. Improved undecidability results on weighted timed automata. *Information Processing Letters*, 98(5):188–194, 2006.
8. Patricia Bouyer, Ed Brinksma, and Kim G. Larsen. Optimal infinite scheduling for multi-priced timed automata. *Formal Methods in System Design*, 32(1):3–23, 2008.
9. Patricia Bouyer, Franck Cassez, Emmanuel Fleury, and Kim G. Larsen. Optimal strategies in priced timed game automata. In *Proceedings of the 24th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'04)*, volume 3328 of *LNCS*, pages 148–160. Springer, 2004.
10. Patricia Bouyer, Samy Jaziri, and Nicolas Markey. On the value problem in weighted timed games. In *Proceedings of the 26th International Conference on Concurrency Theory (CONCUR'15)*, volume 42 of *Leibniz International Proceedings in Informatics*, pages 311–324. Leibniz-Zentrum für Informatik, 2015.
11. Romain Brenguier, Franck Cassez, and Jean-François Raskin. Energy and mean-payoff timed games. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control (HSCC'14)*, pages 283–292. ACM, 2014.
12. Thomas Brihaye, Véronique Bruyère, and Jean-François Raskin. On optimal timed strategies. In *Proceedings of the Third international conference on Formal Modeling and Analysis of Timed Systems (FORMATS'05)*, volume 3829 of *LNCS*, pages 49–64. Springer, 2005.
13. Thomas Brihaye, Gilles Geeraerts, Axel Haddad, Engel Lefaucheux, and Benjamin Monmege. Simple priced timed games are not that simple. In *Proceedings of the 35th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'15)*, volume 45 of *LIPIcs*, pages 278–292. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2015.
14. Thomas Brihaye, Gilles Geeraerts, Axel Haddad, and Benjamin Monmege. To reach or not to reach? Efficient algorithms for total-payoff games. In *Proceedings of the 26th International Conference on Concurrency Theory (CONCUR'15)*, volume 42 of *LIPIcs*, pages 297–310. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2015.

15. Thomas Brihaye, Gilles Geeraerts, Axel Haddad, and Benjamin Monmege. Pseudopolynomial iterative algorithm to solve total-payoff games and min-cost reachability games. *Acta Informatica*, 2016.
16. Thomas Brihaye, Gilles Geeraerts, Shankara Narayanan Krishna, Lakshmi Manasa, Benjamin Monmege, and Ashutosh Trivedi. Adding negative prices to priced timed games. In *Proceedings of the 25th International Conference on Concurrency Theory (CONCUR'14)*, volume 8704, pages 560–575. Springer, 2014.
17. Damien Busatto-Gaston, Benjamin Monmege, and Pierre-Alain Reynier. Optimal reachability in divergent weighted timed games. Research Report 1701.03716, arXiv, January 2017.
18. Thomas Dueholm Hansen, Rasmus Ibsen-Jensen, and Peter Bro Miltersen. A faster algorithm for solving one-clock priced timed games. In *Proceedings of the 24th International Conference on Concurrency Theory (CONCUR'13)*, volume 8052 of *LNCS*, pages 531–545. Springer, 2013.
19. Neil Immerman. Number of quantifiers is better than number of tape cells. *Journal of Computer and System Sciences*, 22(3):384–406, 1981.
20. Neil Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17:935–938, 1988.
21. Marcin Jurdziński and Ashutosh Trivedi. Reachability-time games on timed automata. In *Proceedings of the 34th International Colloquium on Automata, Languages and Programming (ICALP'07)*, volume 4596 of *LNCS*, pages 838–849. Springer, 2007.
22. Leonid Khachiyan, Endre Boros, Konrad Borys, Khaled Elbassioni, Vladimir Gurvich, Gabor Rudolf, and Jihui Zhao. On short paths interdiction problems: Total and node-wise limited interdiction. *Theory of Computing Systems*, 43(2):204–233, 2008.
23. Anuj Puri. Dynamical properties of timed automata. *Discrete Event Dynamic Systems*, 10(1-2):87–113, 2000.
24. Michał Rutkowski. Two-player reachability-price games on single-clock timed automata. In *Proceedings of the Ninth Workshop on Quantitative Aspects of Programming Languages (QAPL'11)*, volume 57 of *EPTCS*, pages 31–46, 2011.
25. Róbert Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26(3):279–284, 1988.