Master research Internship

Bibliographic report

Stochastic Strategies in Quantitative or Timed Games

Domain: Computer Science and Game Theory

Author:
Julie Parreaux

Supervisor:
Benjamin Monmege
Pierre–Alain Reynier
Modelisation and Verification - LIS

Abstract: Games theory is well-established in the construction of complex reactive systems correct by construction. Two-player games model interaction between two antagonistic agents on a system: an environment and a controller. The controller strategy synthesis problem for the system requires we build a controller that ensures the system specification whenever the environment does. It can be modelled by the winning strategy synthesis on a game. We search to build a strategy as simple as possible. A natural criterion is the memory needed for this strategy. In the internship, we will focus on two classes of games : quantitative and priced timed games. This bibliographic review introduces theses games with their specificities. For each game, we give the complexity of the problem of finding a winning strategy and the memory needed for the strategies. These state of the art results give the context of the internship and justify the study of the trade-off between memory and probabilities for strategies in these two classes of games.

Contents

1 Introduction

The verification and correction of complex reactive systems are difficult. When an error is found, debug these systems is another hard problem. Game theory on graph is a well-established technique to design systems which are correct by construction. To build this system, we design a controller such that the systems ensure its specification. We solve the controller synthesis problem : build a correct controller for a given system model. In this problem, we can not ensure that the environment and the controller collaborate together. The controller is built such that properties are ensured whatever the environment performs. Games on graphs model the system behaviour by interactions between two antagonistic agents (called players) : an environment (the system's environment) and a controller (to control the system).

A two-player game models interaction between two players : Adam and Eve, with antagonistic objectives. In this situation, we focus on winning strategy problem for both players. A game is composed of a graph (usually named arena) and an objective. The graph gives the rules of the game: it describes the available behaviours of the players. An objective defines who wins the game: it models the system's specification. We suppose that the considered games have no tie : Eve or Adam always win, specification is satisfied or not. We must decide who wins the game when the game starts in a given vertex. Moreover, for the winner, we would decide if there exists a way for this player to play and ensure that he wins. When the system's specification is encoded in the objective, the controller synthesis problem can be approached by the winning strategy synthesis problem on a game.

There exist many games on graphs. The different instantiations of the graph (with or without weights, probability distributions on transitions, finite or infinite, ...) and objectives build distinct games. In this bibliographic review, we will focus on quantitative and timed games with a shortest path objective. Quantitative games can model quantitative parameters on the system. For example, these games can model energy consumption or functional requirement. It is a way to order the strategy for an objective and choose the best. Timed games model timed reactive systems or systems with time issues. These games are an example of infinite games. However, they can be represented with a timed automaton that is an extension of a finite automaton with time. When we add quantitative parameters in timed automata, we obtain priced automata. They describe priced timed games. For quantitative and priced timed games, we focus on the shortest path objective. It is an extension of a reachability objective as a combination of two objectives : a reachability objective and a minimisation of the system's execution's weight.

In the internship, we will focus on one particular problem : the synthesis of an optimal strategy. It searches to build this optimal strategy if it exists. It is a natural extension of the problem to the existence of an optimal strategy problem : we search if an optimal strategy exists. The existence problem asks whether there is a strategy for a player such that whatever the adversary plays the system's execution has a weight $\leq c$, for c a threshold. This problem allows one to solve other problems as the value problem that consists in computing the optimal value for all vertices of the arena. This value is the optimal weight that a player can be expected whatever the adversary plays. It is a way to compute who wins the game. The value problem asks whether the value for a vertex is $\leq c$, for c a threshold.

We study the synthesis problem with a question on the memory needed for this optimal strategy if we can build it. When a strategy is synthesised, we search the simplest one possible : it is a strategy without memory. However, in some games such as quantitative or priced timed games, memory is needed to obtain an optimal strategy. We search to quantify this memory for each game where it is necessary.

In Section 2, we introduce the basic notion for theses games and we present an example of an infinite game : a timed game. In Section 3, we introduce quantitative games. After presenting classical objectives for theses games, we present the synthesis of optimal strategy for the shortest path objective. In Section 4, we extend timed games with weights. We study the decidability classes of games for previous problems.

2 Game Theory on Graph

A two-player game between Adam and Eve on a graph is composed of two elements: an arena (a graph) and a winning condition called objective. Given these two elements, we design a particular game. The important problems on a game are the existence and the synthesis of a winning strategy for at least one player of this game.

2.1 Syntax and semantics

A game $\mathcal{G} = (\mathcal{A}, W)$ is defined by an arena $\mathcal{A}$ and an objective W. An *arena* of a game is a directed graph $\mathcal{A} = (V, E)$ (not necessarily finite, see Section 2.3 for an example) where $E \subseteq V \times V$ is the transition relation. A *run* is an infinite path in the arena. We denote $\pi = v_1 v_2 \dots$ a run in $\mathcal{G}$ where $v_i \in V$ and $(v_{i-1}, v_i) \in E$ for all i. A *finite run* is a run that finishes in a blocking vertex i.e. a vertex which has exit transitions. For the next, we suppose, without loss of generality, that there are no blocking vertices. Otherwise, blocking vertices can be transformed into an absorbing vertex by adding a self-loop. A *history* is a finite prefix of a run.

We consider a turn-based semantic to describe the choice of the players. A single token representing the current state of the game is moved by the players. The current player who is decided by the token vertex attribution chooses the next move of the token. Formally, to represent this semantic, we consider in the arena that $V = V_{Eve} \cup V_{Adam}$ where V_{Eve} is Eve's vertices and V_{Adam} Adam's vertices such that $V_{Eve} \cap V_{Adam} = \emptyset$. Figure 1 represents an arena of a turn-based game where V_{Eve} is represented by the circle and V_{Adam} by the rectangle. If the token is in $v_0 \in V_{Adam}$, Adam chooses to move to v_0 or v_1. As the same, in $v_6 \in V_{Eve}$, Eve chooses to move to v_3 or v_5. A run in this turn-based game is $\pi = v_6 v_3 v_2 (v_0 v_1)^\omega$.

Now, we define how Eve can win in a game $\mathcal{G} = (\mathcal{A}, W)$: the objective. More precisely, we focus on Eve's objective: $W \subseteq V^\omega$ (W can be an infinite set). Eve wins under W with a run π if and only if $\pi \in W$. When the game has some blocking vertices, we define a finite winning run π such that the token reaches a deadlock vertex for Adam. Moreover, we assume that Adam wins when Eve does not win, there is no tie in a game. This hypothesis implies that the Adam objective set is the complement of W.

There exist many objectives but we focus on a particular objective: the Reachability objective denoted $Reach(T)$ where $T \subseteq V$ is a subset of vertices to represent the target. Let $\pi = (v_i)_i$ a run in a game $\mathcal{G}$, Eve wins with π if there exists i such that $v_i \in T$. Adam solves the dual objective, i.e. the safety objective, he must not reach T. He wins with π if for all $v_i \notin T$. A game $\mathcal{G} = (\mathcal{A}, Reach(T))$ is a reachability game.

Example 1. *Consider the reachability game represented in Figure 1 with $T = \{v_0\}$. The run $\pi = v_6 v_3 v_2 (v_0 v_1)^\omega$ is winning for Eve, but the run $\pi' = v_6 v_3 v_2 (v_4 v_2)^\omega$ is winning for Adam.*

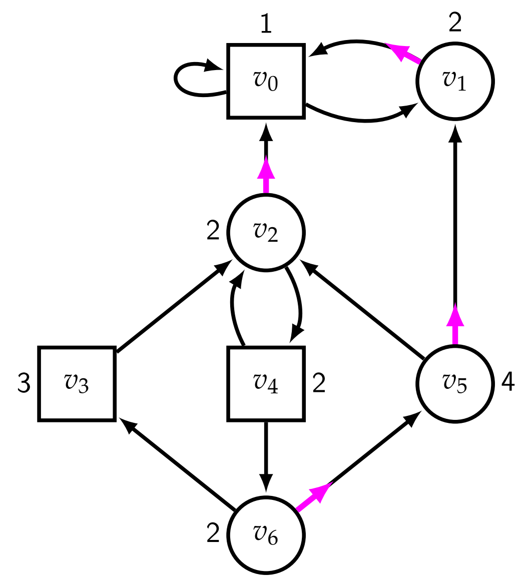


Figure 1: An arena for a regular game where Eve vertices are circle and Adam ones are rectangle.

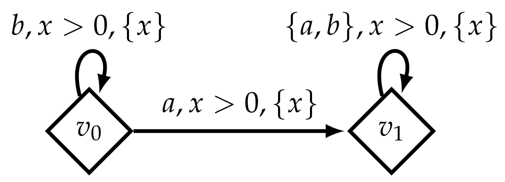


Figure 2: A concurrent timed game where Eve plays with $\{a\}$ and Adam with $\{b\}$.

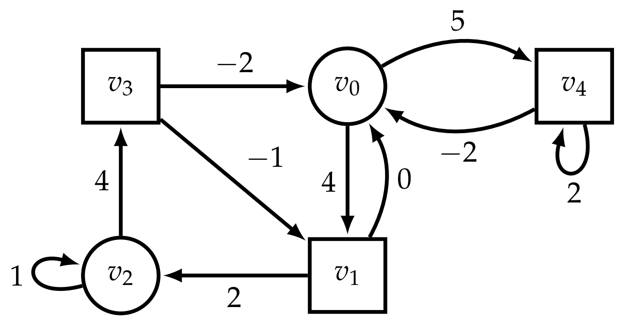


Figure 3: An arena of a quantitative game. Eve vertices are circle and Adam are rectangle one.

Related work on ω-regular objectives We can extend the reachability objective with ω-regular objectives [11]. These objectives are a concise syntax with the notion of vertex colours. Formally, let a finite set of colours $\mathcal{C}$ to be a finite subset of $\mathbb{N}$, we introduce the colouring function $\xi : V \to \mathcal{C}$. For example, in Figure 1, $\xi(v_0) = 1$. For a run π in G, the colour of π is a natural extension of the colouring matching $\xi : V^\omega \to \mathcal{C}^\omega$. The run colour is a word (over the colour) based on the colour of the run vertices (called ω-regular objective). A game with an objective of this class is a regular game.

A *Parity objective* accepts a run π such that its colour satisfies a parity condition. Let $p(\pi) = \min(\inf(\xi(\pi)))$ be the minimal colour that occurs infinitely often in an infinite run π. A parity objective accepts the run π if and only if $p(\pi)$ is even. Let $\mathcal{F}$ be a set of subsets of $\mathcal{C}$, a *Müller objective* accepts a run π such that its colour verifies $\inf(\xi(\pi)) \in \mathcal{F}$, i.e. the set of colours that occur infinitely often is an element of $\mathcal{F}$. The reachability objective is an ω-regular objective. If we consider $\mathcal{C} = \{0, 1\}$ such that all $\xi(v) = 1$ if and only if $v \in T$, then we accept a run π if this colour verifies $\max(\xi(\pi)) = 1$.

Example 2. *Figure 1 represents an arena of a regular game $\mathcal{G}$ with $\mathcal{C} = \{1, 2, 3, 4\}$. The colouring function ξ is represented by vertices labels on $\mathcal{G}$. Let $\pi = v_6 v_3 v_2 (v_0)^\omega$ and $\pi' = (v_2 v_4 v_6 v_3)^\omega$ be two runs of $\mathcal{G}$.*

Case of G with a Parity objective. The run π is wining for Adam because $p(\pi) = \min(\inf(\xi(\pi))) = \min(\inf(\xi(v_6 v_3 v_2 (v_0)^\omega))) = \min(\inf(231(1)^\omega)) = \min\{1\} = 1$ and $p(\pi)$ is odd. However, the run π' is winning for Eve because $p(\pi') = \min(\inf(\pi')) = \min\{2, 3\} = 2$ and $p(\pi')$ is even.

Case of G with a Müller objective with $\mathcal{F} = \{\{1, 2\}, \{1, 2, 3, 4\}\}$. The run π is wining for Eve because $\inf(\xi(\pi)) = \inf(\xi(v_6 v_3 v_2 (v_0 v_1)^\omega)) = \inf(232(12)^\omega) = \{1, 2\} \in \mathcal{F}$. However, the run π' is winning for Adam because $\inf(\xi(\pi')) = \{2, 3\} \notin \mathcal{F}$.

2.2 Strategies

To win the game $\mathcal{G} = (\mathcal{A}, W)$, Eve must make a good choice at each her turn. These choices are described by a *strategy*. It is a matching $\sigma : V^* V_{Eve} \to V$ mapping history in $\mathcal{G}$ to the vertex she wants to move the token to. A strategy for Adam is similarly defined by $\sigma : V^* V_{Adam} \to V$. Let π be a run and σ be a Eve's strategy (respectively Adam's strategy σ), we say that $\pi = v_0 v_1 \ldots$ is *conform* to σ if for all i such that $v_i \in V_{Eve}$ (respectively V_{Adam}), then $\sigma(v_0 v_1 \ldots v_i) = v_{i+1}$. For a game $\mathcal{G} = (\mathcal{A}, W)$, σ is a *winning strategy* for Eve if and only if all runs conform with σ are in W. For a given vertex v, Eve has a winning strategy σ if for all run π starting in v and conforms with σ are in W. We define analogously the winning strategy for Adam.

The determinacy of a game $\mathcal{G}$ gives a symmetrical argument to reason on $\mathcal{G}$. A game $\mathcal{G}$ is *determined* if and only if, for all vertices of $\mathcal{G}$, Eve or Adam have always a winning strategy. It justifies that we only focus on Eve's point of view.

Theorem 1 ([11]). *Every game $\mathcal{G}$ with an ω-regular objective is determined.*

Let $\mathcal{G}$ be a game, a classical problem is to decide the existence of a winning strategy for Eve. When we can decide that this strategy exists, we want to compute one (synthesis problem). An example of a simple strategy is a strategy built by an attractor. This notion is the base element for many objectives. We illustrate this notion on reachability games in the example 3.

Example 3. *Let $\mathcal{G}$ be a reachability game with a set of vertices T. In $\mathcal{G}$, we can compute the winning strategy for Eve in polynomial time with an attractor. It defines the set of vertices where Eve wins. We use the function Pre that defines the set of vertices such that whatever Adam plays, Eve reaches X on one step. Formally, Pre is defined for all X subset of vertices V, as $Pre(X) = \{v \in V_{Eve} | \exists (v, v') \in E, v' \in X\} \cup \{v \in V_{Adam} | \forall (v, v') \in E, v' \in X\}$. The Eve's attractor for T is the least fixed point of $: X \mapsto Pre(T \cup X)$. It can be computed iteratively as following in polynomial time [11]. We start with $X_0 = \emptyset$ and for each iteration based on X_i, we compute the set $X_{i+1} = Pre(T \cup X_i)$. Intuitively, at each iteration, we add to X_{i+1} all Eve's vertices such that it exists a transition to reach $X_i \cup T$ and all Adam's vertices, such that all transitions from it reach $X_i \cup T$. Eve's winning strategy is defined by the witness which allows to reach X_i.*

Let the reachability game represented in Figure 1 with $T = \{v_0\}$. Let $X_0 = \emptyset$, we describe the Eve's attractor computation. In the first step, $X_1 = Pre(T \cup X_0) = \{v_1, v_2\}$. We note that $v_2 \in X_1$ as v_2 reaches T with (v_2, v_0), but $v_0 \notin X_1$ because v_0 does not reach T from all edges: we can take (v_0, v_1). Then, we have $X_2 = Pre(T \cup X_1) = \{v_0, v_1, v_2, v_3, v_5\}$, $X_3 = Pre(T \cup X_2) = \{v_0, v_1, v_2, v_3, v_5, v_6\}$ and $X_4 = Pre(T \cup X_3) = V$ In the last step, $v_4 \in X_4$ because at this moment, all edges from v_4 reach $T \cup X_3$. Eve's winning strategy is represented by pink arrows in Figure 1.

Let the reachability game represented in Figure 1 with $T = \{v_1\}$. Eve's attractor computation converges in two steps when we add $\{v_5, v_6\}$. By the determinacy result, Eve has a winning strategy when the game starts from v_5 or v_6 and Adam has a winning strategy in the other cases.

Related work on different types of strategies To compute the simplest strategy, we need to order strategies. One criterion to compare strategies is the memory they use. In general, a strategy needs infinite memory since we must remember the whole history of the run. In several cases, knowing the whole history is not necessary through. A strategy is said *memoryless* (or positional) if for all two distinct history $v_0 v_1 \ldots v_i$ and $v'_0 v'_1 \ldots v'_i$ with $v_i = v'_i$, then $\sigma(v_0 v_1 \ldots v_i) = \sigma(v'_0 v'_1 \ldots v'_i)$. In this case, we denote $\sigma : V_{Eve} \to V$ a memoryless strategy for Eve. For example, a strategy based on an attractor (see example 3) is a memoryless strategy. We also consider *finite strategy memories*. They can be described by a Moore machine (M, m_0, up, dec) where M is a finite set representing the memory of the strategy, $m_0 \in M$, $up : M \times V \to M$ is the memory update function and $dec : M \times V \to V$ a decision function such that for all histories π and vertices v, $\sigma(\pi v) = dec(mem(\pi v), v)$ where $mem(\pi)$ is defined by induction on the length of the history π as follows : $mem(v_0) = m_0$ and $mem(\pi v) = up(mem(\pi), v)$. In this case, we say that $|\mathcal{M}|$ is the size of the strategy.

The memory is not the unique parameter to compare strategies. Stochastic games are games where the transition relation is a probabilistic distribution over vertices. For such games, it is natural to use a *probabilistic strategy*. A probabilistic strategies for Eve is defined as $\sigma : V^* V_{Eve} \to \mathcal{D}(V)$ where $\mathcal{D}$ is a probabilistic distribution over adjacent vertices [9].

2.3 An example of infinite game: timed games

Timed games are a particular class of infinite games (where the arena is infinite). The arena of this game is a graph where vertices are a location (from a finite set of locations) and the real value of some "clocks". However, it can be represented in a finite way. We use timed automata to represent finitely these games. We only study the reachability objective on these games so we call them, reachability timed games.

We store time with some variables called *clocks*. Let $\mathcal{C}$ be a set of clocks, we call *valuation* a function $\nu : \mathcal{C} \to \mathbb{R}_{\geq 0}$ such that for all clocks $c \in \mathcal{C}$, $\nu(c)$ is the value of c. We denote $V(\mathcal{C})$ the set of valuations of $\mathcal{C}$. Let $c \in \mathcal{C}$, there exist two actions on c: let some time elapse and reset c. If we would let pass $t \in \mathbb{R}_{\geq 0}$ time units, we denote, for all $\nu \in V(\mathcal{C})$, $(\nu + t)(c) = \nu(c) + t$. Time passes with the same speed for all clocks in $\mathcal{C}$. Let $C \subseteq \mathcal{C}$ be a set of clocks. The valuation $\nu[C := 0]$ returns 0 for all $c \in C$ and it returns $\nu(c)$ otherwise. This valuation models the reset of clocks of C. Moreover, we give *guards* that are some constraints of clocks. Let $c, c' \in \mathcal{C}$, $i \in \mathbb{N}$ and $\bowtie \in \{<, \leq, =, \geq, >\}$, a simple constraint over c and c' is the form $c \bowtie i$ or $c - c' \bowtie i$. A guard over $\mathcal{C}$ is a conjunction of simple constraints: this defines a convex set of clocks valuations. We denote $Guard(\mathcal{C})$ the set of guards in $\mathcal{C}$.

A *timed automaton* is a tuple $\mathcal{A} = (L, \mathcal{C}, \Sigma, \delta, Inv)$ where L is a finite set of locations, $\mathcal{C}$ is a finite set of clocks, Σ is a finite alphabet, $\delta \subseteq L \times \Sigma \times Guard(\mathcal{C}) \times L \times 2^{\mathcal{C}}$ is the set of transitions and $Inv : L \to Guard(\mathcal{C})$ is a function to assign an invariant defined by a guard for each location of $\mathcal{A}$. In this automaton, a transition is available if the valuation satisfies the guard on the transition and the invariant in the new location when we have reset the clocks. A *configuration* of a timed automaton is a location and the valuation of all clocks such that it satisfies this location's invariant. The *graph of configurations* of a timed automaton is an infinite graph such that the vertices of this graph are the configurations and the edges are given by δ. There exist an edge between two configurations (l_1, v_1) and (l_2, v_2) if and only if there exist $a \in \Sigma$ and $g \in Guard(\mathcal{C})$, such that $(l_1, a, g, l_2, v_2) \in \delta$.

Example 4. *Let $\mathcal{A}$ be the timed automaton represented in Figure 2 such that $L = \{v_0, v_1\}$, $\mathcal{C} = \{x\}$, $\Sigma = \{a, b\}$, $\delta = \{(v_0, a, \{x > 0\}, v_1, \nu[x := 0]), (v_0, b, \{x > 0\}, v_0, \nu[x := 0]), (v_1, a, \{x > 0\}, v_1, \nu[x := 0]), (v_1, b, \{x > 0\}, v_1, \nu[x := 0])\}$ and $Inv : Q \to true$.*

A *timed game* is based on the configuration graph of a timed automaton: it is its arena. In each location, players make two actions: choose a delay and an available transition with this delay. It chooses the delay to spend in the location (it can be null) before to take the chosen transition. This delay must satisfy the location invariant and the guard of the chosen transition. There exist two ways to make the vertices partition on the arena : concurrent games and turn-based games. An arena of a *turn-based timed game* is based on a timed automaton where L is split in two distinct sets L_{Eve} and L_{Adam} as already studied before. In a turn-based reachability timed game, attractors can be adopted do work in this timed setting.

An arena of a *concurrent game* is based on the timed automaton where Σ is split in two distinct sets Σ_{Eve} and Σ_{Adam}. Each set contains the actions that can perform each player. For each location v, we define $\Gamma_{Eve}(v) = \{(a, \delta) \mid a \in \Sigma_{Eve}$ is an available transition and δ is the delay$\} \cup \{(\perp, 0)\}$ the set of possible choices of Eve. We suppose that $(\perp, 0)$ describes a hidden transition to stay in v with the same valuation for all clocks (no reset and no delay). We define as the same $\Gamma_{Adam}(v)$. The player who chooses the smallest delay wins this round and applies his choice. When they propose the same delay, we use an external strategy called a scheduler to choose who wins the round. This game introduces an element of surprise. Turn-based games are a special case of concurrent games where action $(\perp, 0)$ is disallowed. In this class of games, we have a concurrent game where for all locations, we only have one player action to quit the location.

Example 5. *Let $\mathcal{G}$ be the concurrent timed game represented in Figure 2 where $\Sigma_{Eve} = \{a\}$ and $\Sigma_{Adam} = \{b\}$. We have $\Gamma_{Eve}(v_0) = \{(a, \delta) \mid \delta > 0\} \cup \{(\perp, 0)\}$.*

We focus on the reachability objective with T the set of target vertices for Eve in concurrent timed games. A natural Adam winning strategy always chooses $(\perp, 0)$. An infinite run conforms with this Adam strategy implies that the time converges for this run. An infinite run π is *time convergent* if its time is finite. It is *divergent* otherwise. To avoid this undesirable behaviour, the player responsible for this situation loses the run. A *time divergent strategy* is a strategy such that a run conform with this strategy is either time divergent or take only a finite number of transitions. This strategy is not responsible for the time convergent for all run conform with it. A winning strategy for Eve is a time divergent strategy that reaches the target for all time divergent run.

We need to carefully define the memory used by a strategy. In a timed game, we need to know infinitely precisely the value of the clocks (see example 6): storing a configuration requires infinite memory. Nevertheless, a strategy is generally said memoryless when we only to store the current configuration. A strategy is said finite-memory when it stores a finite number of configuration or an extra finite memory describe by a Moore machine. Otherwise, the strategy is said infinite memory. For example, if we need an extra clock with an infinitely precise valuation, it is an infinite memory strategy.

In a concurrent reachability timed games, the problem of the existence of a winning strategy is decidable. However, this strategy may require infinite memory to control the surprise's element of the game [10]. In the concurrent reachability timed game on Figure 2 with $T = \{v_1\}$, the winning strategy for Eve needs infinite memory. It maintains precisely a global clock to choose the good delay when she is in v_0: until Eve choice is not taken, she proposes a delay closer to 0. To make this choice, she needs to use a new global infinitely precise clock y : she chooses $(\frac{1}{2^{v(y)}}, a)$ at each turn. This strategy is a winning strategy because either Adam always chooses $(\perp, 0)$ and time is convergent or Adam can choose at each turn a delay non-null and there exists a moment where Eve chooses a smaller delay and the time diverges.

We can make a trade-off between memory used and probabilities. We reduce the memory used with a probabilistic strategy that chooses a delay uniformly at random in a given interval. A probabilistic strategy for reachability timed game reaches almost surely a target with finite memory.

Theorem 2 ([9]). *Let $\mathcal{G} = (\mathcal{A}, Reach(T))$ be a concurrent reachability timed game with T a set of locations. Eve has a winning, randomized, finite-memory strategy π_{Eve} such that for all locations where Eve can reach T, for all Adam's time divergent strategies π_{Adam}, $Pr_s^{\pi_{Eve}, \pi_{Adam}}(Reach(T)) = 1$.*

Example 6. *Let $\mathcal{G}$ be the concurrent reachability timed game in Figure 2 with $T = \{v1\}$. By Theorem 2, Eve has a winning randomized, finite-memory strategy. Let $\pi_{Eve}(p, \nu) = (a, Uniform(0, 1 - \nu(x)))$ a strategy that chooses the action a with a delay chosen uniformly at random in the interval $]0, 1 - \nu(x)]$. It is a winning strategy. Let δ_j be the delay proposed by Adam in the round j. The probability to never choose Eve's action is $\prod_{j=1}^{\infty}(1 - \delta_j) = 0$ if $\sum_{j=1}^{\infty} \delta_j = \infty$. As Adam uses a time-divergent strategy, $\sum_{j=1}^{\infty} \delta_j = \infty$. Thus, the probability to choose Eve's action is 1 and she reaches v_1 with probability 1.*

3 Quantitative Games

When a game has many strategies, we would like to compare them and choose the best one. A way to classify strategies is to introduce a measure with weights. For example, we can model energy used or created by a robot during a task with weights. We would minimise the energy consumed to solve the objective. Formally, we consider another sort of graphs for the arena of the game, weighted graph. We suppose that weights are in $\mathbb{Z}$. We begin by focusing on finite games with weights: quantitative games.

3.1 Quantitative games

A *quantitative game* is a turn-based game with an arena described by a finite weighted graph $\mathcal{A} = (V, E, w)$ where $V = V_{Adam} \cup V_{Eve}$, $E \subset V \times V$ and $w : E \to \mathbb{Z}$. Figure 3 represents a quantitative game. A run $\pi \in V^{\omega}$ in this game is defined as a run in the classical game.

Example 7. *In Figure 3 we represent an arena of a quantitative game where Eve has the circle vertices and Adam has the rectangle ones. We have, for example, $w(v_3, v_0) = -2$. A run in this game is $v_0 v_1 v_2 v_3 (v_1 v_0)^{\omega}$.*

An objective in quantitative games is given by a particular function : the *payoff*. Intuitively this function gives the weight of a run on the game. For a given payoff, Eve's objective is to maximise it and Adam's objective is to minimise it. For each vertex v of the game, we define $\text{val}^{\mathcal{G}}_{Eve}(v)$ and $\text{val}^{\mathcal{G}}_{Adam}(v)$ as the *value* for Eve and Adam. It is the best value that Eve and Adam can guarantee whatever the adversary plays when the game starts in v. When the game is not ambiguous we can skip $\mathcal{G}$ in the notation.

Under this objective, an *optimal strategy* is a winning strategy for this objective. In other words, an optimal strategy for Eve ensures that she reaches the value given by val_{Eve}. We have the same for Adam. For this game, we study a particular property : the determinacy. A game is *determined* if for all vertices v, $\text{val}_{Adam}(v) = \text{val}_{Eve}(v)$. When a game is determined, we denote by $\text{val}(v)$ the value $\text{val}_{Eve}(v) = \text{val}_{Adam}(v)$ of v and we search if we can solve the game : decide whenever $\text{val}(v) \geq x$ where $x \in \mathbb{Q}$ is a threshold. This problem is called the value problem. To answer it, we search when the strategy for Eve guarantees a payoff greater than x whatever Adam plays, we solve the existence of an optimal strategy problem. We also compute a vector $(\text{val}(v))_v$ that contains all values of the game.

Related work on classical payoff There exist many payoffs for a given quantitative game [12]. Each payoff can express a particular property. Quantitative games can express the ω-regular objective when colours are represented by weights. For example, the payoff $\mathrm{Sup}(\pi) = \sup w((\pi_i, \pi_{i+1}))$ expresses the reachability of T : if we take 1 in all incident edge of T and 0 otherwise. As Eve would maximise this payoff, we consider the maximum on a run π: she wins if the payoff is 1.

Now, we consider arithmetic operations for the payoff. A way to compute the weight of an infinite run is to take the average of the weights in the run. As this average may not be well-defined, we define the *Mean-payoff* objective for a run π as $\mathrm{MeanPayoff}(\pi) = \liminf_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} w((\pi_i, \pi_{i+1}))$. A game $\mathcal{G} = (\mathcal{A}, \mathrm{MeanPayoff})$ is a Mean-payoff game.

Example 8. *Consider the Mean-payoff game represented in Figure 3. The MeanPayoff of the run $\pi = v_0(v_1v_2v_3)^\omega$ is the average weight of the cycle $(v_1v_2v_3)$, i.e. $5/3$.*

Theorem 3 ([12])**.** *Mean-payoff games are determined and both players have memoryless optimal strategies.*

Example 9. *On the Mean-payoff game of Figure 3, Eve and Adam have memoryless optimal strategies, denoted by σ_{Eve} and σ_{Adam} respectively, such that $\sigma_{Eve}(v_0) = v_1$, $\sigma_{Eve}(v_2) = v_3$, $\sigma_{Adam}(v_1) = v_2$, $\sigma_{Adam}(v_3) = v_1$, and $\sigma_{Adam}(v_4) = v_0$.*

The Mean-payoff objective is a long-term objective : it is prefix independent. In economy, for example, the prefix is more important than the long-term. The *Discounted* payoff is an objective that can consider more the prefix. For a run π and a parameter $\lambda \in]0,1[$, the Discounted payoff is $\mathrm{DiscountedPayoff}_\lambda(\pi) = (1-\lambda)\sum_{i=0}^{\infty} \lambda^i w((\pi_i, \pi_{i-1}))$. A game $\mathcal{G} = (\mathcal{A}, \mathrm{DiscountedPayoff})$ is a discounted game.

Theorem 4 ([12])**.** *Discounted games are determined and both players have memoryless optimal strategies.*

Example 10. *Consider the discounted game in Figure 3. Let $\lambda = 0.9$, Eve and Adam have memoryless optimal strategies, denoted by σ_{Eve} and σ_{Adam} respectively, such that $\sigma_{Eve}(v_0) = v_1$, $\sigma_{Eve}(v_2) = v_3$, $\sigma_{Adam}(v_1) = v_2$, $\sigma_{Adam}(v_3) = v_3$, and $\sigma_{Adam}(v_4) = v_0$. We note that this strategy is the same as for the corresponding Mean-payoff game (see Example 9): when λ is close to 1 the discounted objective is the same as the Mean-payoff objective. Now, let $\lambda = 0.5$, only Adam changes his optimal choice : $\sigma_{Adam}(v_1) = v_0$. When $\lambda = 0.1$, the optimal strategy changes for both players: $\sigma_{Eve}(v_0) = v_4$, $\sigma_{Eve}(v_2) = v_3$, $\sigma_{Adam}(v_1) = v_0$, $\sigma_{Adam}(v_3) = v_0$, and $\sigma_{Adam}(v_4) = v_0$.*

Another interesting objective is simply to sum the weights along the run. We define the *Total-payoff* objective : for a run π we have $\mathrm{TotalPayoff}(\pi) = \limsup_{n\to\infty} \sum_{i=0}^{n-1} w((\pi_i, \pi_{i+1}))$. A game $\mathcal{G} = (\mathcal{A}, \mathrm{TotalPayoff})$ is a total-payoff game.

Example 11. *Consider the total-payoff game in Figure 3. For the run $\pi = v_0(v_1v_2v_3)^\omega$, the total-payoff is the sum of the weight of the cycle, i.e. $+\infty$.*

Theorem 5 ([12])**.** *Total-payoff games are determined and both players have a memoryless optimal strategy.*

3.2 Shortest path objective

We have seen that payoffs can be used to encode reachability objectives. Another way to extend this objective is the shortest path objective. In this case, Adam wants to reach a target with the

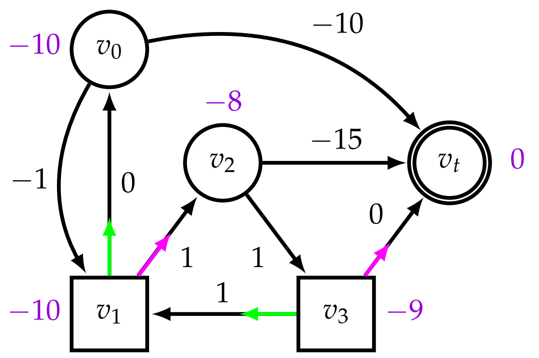
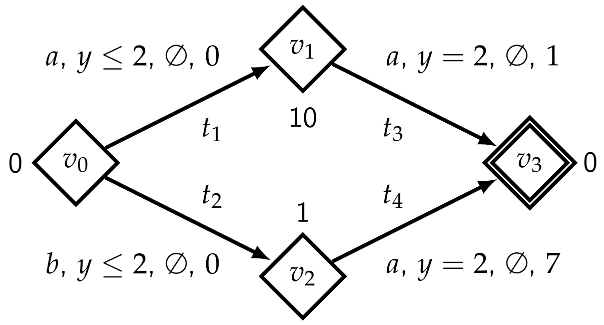


Figure 4: A shortest path game with $T = \{v_t\}$ where vertices are labelled by their value. Eve vertices are circle and Adam are rectangle one.



$a,\ y \leq 2,\ \emptyset,\ 0$
v_1
$a,\ y = 2,\ \emptyset,\ 1$
0
v_0
t_1
10
t_3
v_3
0
t_2
1
t_4
$b,\ y \leq 2,\ \emptyset,\ 0$
v_2
$a,\ y = 2,\ \emptyset,\ 7$

Figure 5: A concurrent shortest path timed game with $T = \{v_3\}$ with $\Sigma_{Adam} = \{a\}$ and $\Sigma_{Eve} = \{b\}$.

smallest total payoff possible. Eve wants to avoid this case: she would not reach the target and when it is not possible, she reaches it with the greatest total-payoff possible. Formally, we define the payoff *Shortest path* for a run π and a set of vertices' target T by :

$$\text{ShortestPath}(\pi) = \begin{cases} +\infty & \text{if } \pi_k \notin T \text{ for all } k \geq 0 \\ \sum_{i=0}^{k-1} w((\pi_i, \pi_{i+1})) & \text{if } k \geq 0 \text{ is the minimal index such that } \pi_k \in T \end{cases}$$

A game $\mathcal{G} = (\mathcal{A}, \text{ShortestPath})$ is a shortest path game.

We note that these games are determined [5]. Like the previous games, we solve the value problem for a game with the problem of optimal strategy synthesis for Adam.

Example 12. *Figure 4 represents a shortest path game where v_t is the only target ($T = \{v_t\}$). Next to each vertex, we write the optimal value when the game begins in this vertex.*

When we suppose that the weights in the game are non-negative, the optimal strategy for Adam can be computed with a sort of Dijkstra algorithm. So the value problem is solved in polynomial time [12]. However, when we add negative weights, we cannot use Bellman–Ford algorithm or other graph algorithms. A reason for this is that Adam's optimal strategy needs memory.

Example 13. *In the shortest path game represented in Figure 4, Adam's optimal strategy needs finite memory. In fact, when the game starts in the vertex v_1, Adam can ensure the value -10 if he chooses twelve times to go to the vertex v_0. He can take twelve times the negative cycle if Eve agrees with that. Then he chooses the vertex v_2 and, if needed, he finishes to choose the target. This is a finite-memory strategy because we need to remember the time spent on the negative cycle. If we suppose that Adam can only use a memoryless strategy, he has two choices in v_1. He can always choose v_0 to stay in the negative cycle. In this case, Eve chooses v_1 to stay in the cycle and never reach T. Or he can choose v_2 and does not enter in the negative cycle. In this case, Adam reaches T but with a payoff of 2. In fact Eve chooses v_3 in v_2 to maximise the payoff.*

In the rest of this section, we consider arbitrary weights. There is an iterative algorithm to build an optimal strategy for Adam with a pseudo-polynomial time complexity [4]. It computes a switching strategy: it is two memoryless strategies combined with some (pseudo-polynomial) memory. Intuitively, this algorithm computes the first strategy to reach the optimal value (reach a negative cycle or the smallest path to reach the target) and a second strategy based on an attractor to reach the target. We apply the first one until we have enough resources to go to the target and obtain the optimal value.

Theorem 6 ([5]). *Let $\mathcal{G}$ be a shortest path game. We can compute in pseudo-polynomial time the values in $\mathcal{G}$.*

Sketch of the proof. We will present the main idea of this algorithm. It computes the value as a fixed point of $F : \mathbb{R}^{|V|} \to \mathbb{R}^{|V|}$ such that for $x \in \mathbb{R}^{|V|}$ we define $y \in \mathbb{R}^{|V|}$ such that for all $v \in V$,

$$y_v = \begin{cases} 0 & \text{if } v \in T \\ \max_{(v,v') \in E} [w(v,v') + x_{v'}] & \text{if } v \in V_{Eve} \setminus T \\ \min_{(v,v') \in E} [w(v,v') + x_{v'}] & \text{if } v \in V_{Adam} \setminus T \end{cases}$$

We can iteratively compute the fixed point of F. We note that the presence of vertex with ∞ value can be done a computation that no terminate.

Find the vertices with value $+\infty$ By definition of the payoff, a vertex has a value $+\infty$ if and only if Adam cannot reach the target from it. We can use classical attractor technique to compute the set of vertices $V_{+\infty} = \{v \in V | val(v) = +\infty\}$ with the same equations as in Example 3.

Find the vertices with value $-\infty$ A vertex has a value $-\infty$ if and only if Adam controls a negative cycle. The presence of these vertices is the main technical difficulty. Adam would take a maximum of time this cycle that minimises the payoff. To reach the target, Adam must quit this cycle. The vertices with value $-\infty$ are exactly those with a value negative in the Mean-payoff game on the same arena. With this equivalence between shortest path and Mean-payoff game, we can deduce a threshold such that when the iteratively computed value of the game is less than of this threshold imply, we know that the value is $-\infty$. We remark that it can be used to compute the value in a Mean-payoff game [12].

The value is computed in pseudo-polynomial time by this algorithm. □

Now we study the synthesis problem of optimal strategies.

Theorem 7 ([4]). *Let $\mathcal{G}$ be a shortest path game.*

1. *Eve has an optimal memoryless strategy computable in pseudo-polynomial time.*

2. *Adam has an optimal pseudo-polynomial memory strategy computable in pseudo-polynomial time.*

Sketch of the proof. 1. For all vertices whose value is $+\infty$, the strategy for Eve is the attractor strategy where the target is $V_{+\infty}$. For all vertices with value $-\infty$ all strategies of Eve are equally bad. For other vertices, Eve's optimal strategy σ^*_{Eve} is $\sigma^*_{Eve}(\pi v) = \text{argmax}\{v' | \forall (v,v') \in E, w(v,v') + val(v')\}$. It is clearly memoryless and we can prove by induction the optimality.

2. For all vertices with value $+\infty$, all strategies of Adam are equally bad. Otherwise, Adam's optimal strategy $\sigma^*_{Adam} = (\sigma^t_{Adam}, \sigma^o_{Adam})$ is a combination of two memoryless strategies. The first one σ^t_{Adam} is a classical attractor for Adam to reach the target. The memoryless strategy σ^t_{Adam} is computed in polynomial time. The second one is based on the algorithm to find the vertices with value $-\infty$. We use the same technique as for Eve :

$\sigma^o_{Adam}(\pi v) = \text{argmin}\{v'|(v,v') \in E, w(v,v') + val(v')\}$. It is clearly a memoryless strategy. With these two strategies, we define σ_{Adam} such that σ_{Adam} is consistent with σ^o_{Adam} until the total payoff is at least $\text{val}(v) + c$ where c is the longest path in the arena. Next σ_{Adam} is consistent with σ^t_{Adam}. We prove by induction on $n = \text{val}(v) + c$ that the payoff of a run from v consistent with σ^n_{Adam} is at least $\max(n, \text{val}(v))$.

□

Example 14. *In the shortest path game of Figure 4, we have represented Adam's optimal strategy which is computed by the algorithm. In each Adam vertex, the green arrow gives the strategy to reach a negative cycle (σ^o_{Adam}) and the pink arrow gives the strategy to reach the target (σ^t_{Adam}).*

Related work on this objective A shortest path game is very close to a total payoff game. There exists a polynomial-time reduction from a total payoff game to a shortest path game. So we can use this result to compute the value of a total payoff game and its strategies [5].

If we consider a *divergent game*, the problem of the value is P-complete [7]. A divergent game is a game without a null cycle. By the structure of the game, we can ensure that the iterative algorithm in the general case converges at most $|V|$ steps.

In some cases, we can naturally introduce probabilities. When we do not know precisely the behaviour of the adversary, we model the behaviour of the adversary with a probabilistic strategy. A variant of the classical shortest path objective is the minimization of the expectation of the shortest path under a stochastic behaviour of the adversary [6].

4 Priced Timed Games

Quantitative games are finite games with weights. A way to consider infinite games is to add time. A timed game with weights is called a priced timed game. We add weights on transitions and locations in the timed automaton. The weight of a run depends on the chosen transitions and the time spend in each chosen locations. We study this game with a shortest path objective. Unfortunately, for the two semantics of the game (concurrent and turn-based) the value problem and the existence of an optimal strategy are undecidable.

4.1 Priced timed game with shortest path objective

A *priced timed game* is a timed game with weights in this arena. It is represented by a priced timed automaton. A *priced timed automaton* is a timed automaton with weights on theses transitions and locations. Formally, a priced timed automaton is a tuple $\mathcal{A} = (L, \mathcal{C}, \Sigma, \delta, Inv, w)$ where L is a finite set of locations, $\mathcal{C}$ a finite set of clocks, Σ a finite alphabet, $\delta \subseteq L \times \Sigma \times Guard(\mathcal{C}) \times L \times 2^{\mathcal{C}}$ a set of transitions, $Inv : L \to Guard(\mathcal{C})$ the invariant in each locations and $w : L \cup \delta \to \mathbb{Z}$ a priced function. As for a timed game, there exist two semantics for theses games : concurrent games and turn-based games. A run in a priced timed game is an infinite sequence of pairs of locations and delays spend in it, denoted $\pi = (v_i, \delta_i)_i \in (V, \mathbb{R}_{\geq 0})^\omega$.

Example 15. *Let $\mathcal{A}$ be the priced timed automaton represented in Figure 5 such that $V = \{v_0, v_1, v_2, v_3\}$, $\mathcal{C} = \{y\}$, $\Sigma = \{a, b\}$, $\delta = \{t_1 = (v_0, a, \{y \leq 2\}, v_1, \emptyset), t_2 = (v_0, b, \{y \leq 2\}, v_2, \emptyset), t_3 = (v_1, a, \{y = 2\}, v_3, \emptyset), t_4 = (v_2, a, \{y = 2\}, v_3, \emptyset)\}$ and $Inv : Q \to true$. Moreover, $w(v_0) = w(v_3) = w(t_1) = w(t_2) = 0$, $w(v_1) = 10$, $w(v_2) = w(t_2) = 1$ and $w(t_4) = 7$.*

The objectives on priced timed games are described by a payoff function as in quantitative games. This function computes the weight of a run in function of weights of taken transition and the weight of the time spent in each location. The payoff of a *Shortest path* objective, as in quantitative games, for a set T of vertices and for all runs $\pi = (v_i, \delta_i)_i$ is :

$$\text{ShortestPath}(\pi) = \begin{cases} +\infty & \text{if } \pi_k \notin T \text{ for all } k \geq 0 \\ \sum_{i=0}^{k-1} w((v_i, v_{i+1})) + \delta_i w(v_i) & \text{if } k \geq 0 \text{ is the minimal index such that } \pi_k \in T \end{cases}$$

A game $\mathcal{G} = (\mathcal{A}, \text{ShortestPath})$ is a shortest path timed game. For these games, as in the previous section, we can define val_{Adam} and val_{Eve} the values that Adam and Eve can ensure whatever the adversary plays. A shortest path timed game is determined if and only if $\text{val}_{Adam}(v) = \text{val}_{Eve}(v)$ for all vertices v. In this case, we denote $\text{val}(v) = \text{val}_{Adam}(v) = \text{val}_{Eve}(v)$.

4.2 Concurrent shortest path timed games

We consider a concurrent game with non-negative weights, i.e. we suppose that the priced function is $w : L \cup \delta \to \mathbb{N}$. In a concurrent game, Σ is split in two sets Σ_{Adam} and Σ_{Eve} as already studied before. In the general case, the value problem and the existence of an optimal strategy problem are undecidable.

Example 16. *Consider the concurrent shortest path timed game in Figure 5 with $\Sigma_{Adam} = \{a\}$ and $\Sigma_{Eve} = \{b\}$. The value of Adam in v_0 is $val_{Adam}(v_0) = \inf_{0 \leq t \leq 2} \max(10(2-t)+1, (2-t)+7) = \inf_{0 \leq t \leq 2} \max(21-10t, 9-t) = 1$. The optimal strategy for Adam in v_0 is $\sigma_{Adam}(v_0) = (a, 2)$. The value of Eve in v_0 is $val_{Eve}(v_0) = \sup_{0 \leq t \leq 2} \min(21-10t, 9-t) = 9$. The optimal strategy for Eve in v_0 is $\sigma_{Eve}(v_0) = (b, 0)$. In this concurrent game, in the first round Eve wins with the null delay, so Eve wins the game because she reaches the target with her value.*

Theorem 8 ([2])**.** *Let $\mathcal{G}$ be a concurrent game with non-negative weights.*

1. *Given a threshold $\bowtie c$, the value problem asks whether $\inf_{\sigma_{Adam}} ShortestPath(\pi) \bowtie c$ where π is conform with σ_{Adam}. It is undecidable.*

2. *Given a threshold $\bowtie c$, the existence problem asks whether there is a strategy σ_{Adam} for Adam such that for every strategy σ_{Eve} for Eve, it holds $ShortestPath(\pi) \bowtie c$ where π is conform with σ_{Adam} and σ_{Eve}. It is undecidable.*

Related work on decidable games There exists a subclass of games where the value problem [2] and the existence of an optimal strategy problem are decidable [1]. Theses games verify the non-Zenoness hypothesis. It permits to bound the game. Under this hypothesis, the payoff of an infinite run is infinite, otherwise, it can be finite. Formally, we use a technical restriction on the region automaton. For a game under the non-Zenoness hypothesis, the existence of an optimal strategy problem is decidable [1]. If we relax this hypothesis a little, the value problem is still approximable [2].

4.3 Turn-based shortest path games

We consider a turn-based game with arbitrary weights. In a turn-based game, L is split into two sets L_{Eve} and L_{Adam} as already studied before. Turn-based priced timed games are determined.

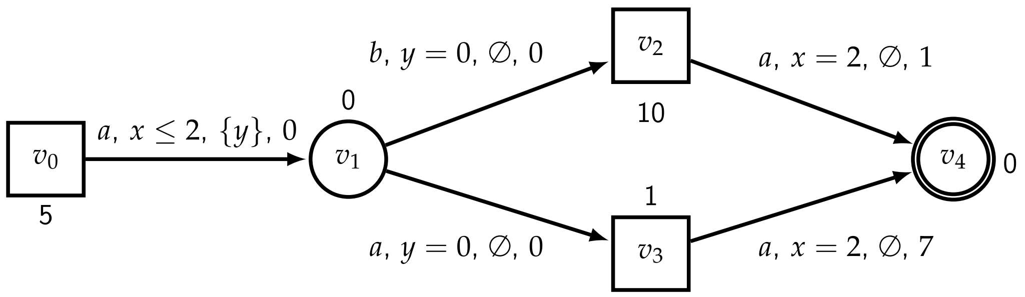

v_0
5
$a,\ x \le 2,\ \{y\},\ 0$
v_1
0
$b,\ y = 0,\ \emptyset,\ 0$
$a,\ y = 0,\ \emptyset,\ 0$
v_2
10
v_3
1
$a,\ x = 2,\ \emptyset,\ 1$
$a,\ x = 2,\ \emptyset,\ 7$
v_4
0

Figure 6: A turn-based shortest path timed game where Eve's vertices are circles ones and Adam's are rectangle ones.

For these games, the existence of an optimal strategy problem is undecidable for two clocks or more [5]. However, we do not know if the existence of an optimal strategy problem is decidable for only one clock.

Example 17. *Consider the turn-based shortest path timed game in Figure 6 where Adam's vertices are rectangle ones and Eve's vertices are circle ones. We can compute the value of Adam in v_0 as $val_{Adam}(v_0) = \inf_{0 \le t \le 2} \max(5t + 10(2-t) + 1, 5t + (2-t) + 7) = \inf_{0 \le t \le 2} \max(21 - 5t, 9 + 4t) = 14 + \frac{1}{3}$. The optimal strategy for Adam in v_0 is $\sigma_{Adam}(v_0) = (a, \frac{4}{3})$.*

Theorem 9 ([5])**.** *The existence of an optimal strategy problem for a turn-based shortest path game is undecidable for two clocks or more.*

Simple priced timed game *Simple priced timed games* are a subclass of games where the existence of an optimal strategy problem and the value problem are decidable [3]. It is a priced timed game with only one clock x that it is never reset and its valuation is bounded by 1. Moreover, all guards over x are $0 \le x \le 1$. Formally, a simple priced timed game is represented by a tuple $\mathcal{A} = (L, \mathcal{C}, \Sigma, \delta, w)$ where $L = L_{Eve} \cup L_{Adam}$, $\mathcal{C} = \{x\}$, $\delta \subseteq L \times \Sigma \times L$ and $w : L \cup \delta \to \mathbb{Z}$. In this game, the total time spent is between 0 and 1. When x reaches 1, the current player must play with a delay null.

The problems of the value and existence of optimal strategy are decidable for those games [3]. The idea of optimal strategies is based on quantitative games. The only difference is on the weight for the time spent in each location.

Theorem 10 ([3])**.** *Let $\mathcal{G}$ a simple priced timed game. The value problem for all vertices, as well as a pair of optimal strategy $(\sigma_{Adam}, \sigma_{Eve})$ can be computed in exponential time.*

Sketch of the proof. The main idea is to reduce the simple priced timed game on a quantitative game. We consider *urgent* locations where the current player must play with a delay null. When the locations of a game are all urgent locations, we obtain a quantitative game where time cannot pass. We can apply the quantitative game's algorithms to compute the value and optimal strategy. The idea of the transformation is the following. When we consider a minimal weight's location, Eve would spend minimal time in it and Adam would spend the maximal time. □

The optimal strategies computed by this algorithm has the same structure of the switching strategy in the case of quantitative games. It is computed in the game where all locations become urgent. An optimal strategy for Eve is always memoryless. However, an optimal strategy for Adam is finite-memory [3].

Related work on decidable games There exist other subclasses of decidable games. The previous result on simple priced timed games can be extended to games with one clock where the number of resets is bounded, i.e. there are no cycles in the region automaton that contains a reset. Based on the same technique that for the simple priced timed game, the value problem can be solved in exponential time [3]. We consider some other restrictions to give a decidable game.

A *bi-valued priced timed game* has only one clock that is bounded by the greatest constant in the guards. Moreover, we suppose that the weight on the vertices are in $\{-d, 0, d\}$ for $d \in \mathbb{N}$ [5]. Bi-valued priced timed games are determined. The value and the existence of optimal strategy problems are decidable for these games. Moreover, an optimal strategy for Adam uses finite memory and an optimal strategy for Eve may need infinite memory. The optimal strategy can be approximated in pseudo-polynomial time [5].

A *divergent priced timed game* is a game under a generalisation of the strictly non-Zenoness hypothesis in the case of non-negative weights. In this case, there are no restrictions on the number of clocks of the game, the restriction is under the payoff of a run. The problem of the value is in 2EXPTIME and EXPTIME-hard [7]. The problem of the existence of an optimal strategy is 2EXPTIME [8].

5 Conclusion

In this bibliographic review, we have presented some games on graphs. These games are based on the same formalism : an arena (a graph) and an objective. For a given game, we focus on the existence of a winning strategy for each player. During the strategy synthesis, we study the memory requirement. For the internship, we study two classes of games : quantitative and priced timed games with a shortest path objective for both. For these games, we study the existence of an optimal strategy problem: it consists in deciding if an optimal strategy exists for Adam. This problem solves another problem : the value problem. It decides the winning player in a determined game. Moreover, it can extend the optimal strategy synthesis problem when we search the memory needed.

In quantitative games with a shortest path objective, the value problem and the existence of an optimal strategy problem are both decidable in pseudo-polynomial time. The optimal strategy for Adam needs finite-memory: a pseudo-polynomial memory. One goal of the internship is to study the trade-off between memory and probability for theses games. We search if there exists a probabilistic and memoryless strategy. An idea is to combine with probability the two memoryless strategies of the switching strategy. Moreover, another question is whether there exists a probabilistic strategy better than a deterministic strategy.

In turn-based priced timed games with a shortest path objective, the value problem and the existence of an optimal strategy problem are both undecidable for at least two clocks. There exist subclasses of games where theses problems become decidable. When the existence of an optimal strategy problem is decidable, we extend this to the synthesis problem. In this case, we search the memory requirement for the optimal strategy. The second goal of the internship is to study a trade-off between memory and probabilistic for theses games. As in [10], we search if there exists a probabilistic strategy which needs less memory than in the deterministic strategy. To begin, we can focus on simple priced timed games and study if the probabilistic memoryless strategy can be used for these games. Moreover, we can study if probabilistic strategies, we do better for this class of games.

References

[1] Patricia Bouyer, Franck Cassez, Emmanuel Fleury, and Kim Guldstrand Larsen. Optimal strategies in priced timed game automata. In *FSTTCS 2004: Foundations of Software Technology and Theoretical Computer Science, 24th International Conference, Chennai, India, December 16-18, 2004, Proceedings*, pages 148–160, 2004.

[2] Patricia Bouyer, Samy Jaziri, and Nicolas Markey. On the value problem in weighted timed games. In *26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, Spain, September 1.4, 2015*, pages 311–324, 2015.

[3] Thomas Brihaye, Gilles Geeraerts, Axel Haddad, Engel Lefaucheux, and Benjamin Monmege. Simple priced timed games are not that simple. In *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India*, pages 278–292, 2015.

[4] Thomas Brihaye, Gilles Geeraerts, Axel Haddad, and Benjamin Monmege. Pseudopolynomial iterative algorithm to solve total-payoff games and min-cost reachability games. *Acta Informatica*, 54, 07 2016.

[5] Thomas Brihaye, Gilles Geeraerts, Shankara Narayanan Krishna, Lakshmi Manasa, Benjamin Monmege, and Ashutosh Trivedi. Adding negative prices to priced timed games. In *CONCUR 2014 - Concurrency Theory - 25th International Conference, CONCUR 2014, Rome, Italy, September 2-5, 2014. Proceedings*, pages 560–575, 2014.

[6] Véronique Bruyère, Emmanuel Filiot, Mickael Randour, and Jean-Francois Raskin. Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. *Leibniz International Proceedings in Informatics, LIPIcs*, 25, 03 2014.

[7] Damien Busatto-Gaston, Benjamin Monmege, and Pierre-Alain Reynier. Optimal reachability in divergent weighted timed games. In *Foundations of Software Science and Computation Structures - 20th International Conference, FOSSACS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*, pages 162–178, 2017.

[8] Damien Busatto-Gaston, Benjamin Monmege, and Pierre-Alain Reynier. Symbolic approximation of weighted timed games. In *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, December 11-13, 2018, Ahmedabad, India*, pages 28:1–28:16, 2018.

[9] Krishnendu Chatterjee, Luca de Alfaro, and Thomas A. Henzinger. Trading memory for randomness. In *Proceedings of the The Quantitative Evaluation of Systems, First International Conference*, QEST '04, pages 206–217, Washington, DC, USA, 2004. IEEE Computer Society.

[10] Krishnendu Chatterjee, Thomas A. Henzinger, and Vinayak S. Prabhu. Trading infinite memory for uniform randomness in timed games. In *Hybrid Systems: Computation and Control, 11th International Workshop, HSCC 2008, St. Louis, MO, USA, April 22-24, 2008. Proceedings*, pages 87–100, 2008.

[11] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata Logics, and Infinite Games: A Guide to Current Research*. Springer-Verlag New York, Inc., New York, NY, USA, 2002.

[12] Benjamin Monmege. *Games on Graphs*, chapter 4 : Payoffs. Unpublished, 2019.