Game Theory for Real-Time Synthesis: Decision, Approximation, and Randomness

Benjamin Monmege

Aix*Marseille université
Socialement engagée

HABILITATION À DIRIGER DES RECHERCHES

Benjamin MONMEGE

Game Theory for Real-Time Synthesis: Decision, Approximation, and Randomness

Théorie des jeux pour la synthèse temps-réel: décision, approximation et aléatoire

Discipline : Informatique

Laboratoire : Laboratoire d'Informatique et Systèmes

Composition du jury

Eugène Asarin Université de Paris	Examinateur
Béatrice Bérard Sorbonne Université	Examinatrice
Véronique Bruyère Université de Mons	Rapporteuse
Marcin Jurdziński University of Warwick, UK	Rapporteur
Nicolas Markey CNRS, Irisa	Rapporteur
Pierre-Alain Reynier Aix-Marseille Université	Examinateur
Yann Vaxès Aix-Marseille Université	Examinateur

Résumé

Les systèmes logiciels sont omniprésents et leur fiabilité est souvent cruciale, en particulier pour ceux qui fonctionnent dans du matériel embarqué, où la sobriété énergétique est aussi un paramètre crucial. Pour apporter des garanties sûres, les méthodes formelles sont utilisées depuis de nombreuses années, avec beaucoup de succès, avec entre autres des techniques basées sur les automates et le célèbre *model checking*. Au lieu de créer manuellement un logiciel, son modèle et de vérifier celui-ci dans un deuxième temps, un autre paradigme consiste à synthétiser automatiquement un logiciel correct par construction, en utilisant la théorie des jeux. Deux joueurs, le *contrôleur* et un *environnement*, s'affrontent dans un jeu à somme nulle joué sur un graphe de toutes les configurations du système. L'énergie, ou d'autres mesures quantitatives selon le problème envisagé, sont ensuite modélisées par l'ajout de poids dans les jeux. Certains résultats préliminaires avaient été obtenus en présence de pondérations positives uniquement, mon objectif a été d'autoriser l'usage de poids négatifs permettant de modéliser des données quantitatives plus riches. Ce manuscrit résume mes contributions dans l'étude d'une combinaison d'un objectif d'accessibilité avec une métrique de gain total : on s'intéresse donc à des *jeux de plus court* chemin où un joueur veut atteindre une cible du graphe tout en minimisant le poids accumulé. Notre étude commence par le cas des jeux sur graphes finis, avec la recherche d'algorithmes efficaces pour calculer les valeurs et stratégies optimales. Nous continuons ensuite en ajoutant des horloges et des contraintes de temps dans les jeux, afin de concevoir des programmes efficaces satisfaisant certaines spécifications temps-réel. Cela conduit à une classe de jeux notoirement difficiles où l'indécidabilité émerge très rapidement. Notre travail consiste en la recherche de fragments possédant des résultats décidables, soit en limitant le nombre d'horloges, soit les comportements cycliques du jeu. Un dernier chapitre étudie également l'utilisation de randomisation dans les stratégies afin de limiter la quantité de mémoire nécessaire pour que les joueurs jouent de manière optimale.

Abstract

Software systems are ubiquitous, and their reliability is often crucial, especially for the ones operating in embedded hardware, where the energetic sobriety moreover matters. To bring reliable guarantees, formal methods have been used for many years, with great success, among them automata based techniques and the renowned model checking approach. Instead of manually building a model, and verify it afterwards, another paradigm consists in automatically synthesising such correct by construction software, using game theory. Two players, the *controller* and an *environment*, are fighting one against the other in a zero-sum game played on a graph of all possible configurations of the system. Energy, or other quantitative metrics of interest, are then modelled by the addition of weights in the games. While some results have been obtained in the presence of non-negative weights only, my objective has been to incorporate negative weights in the picture, to model richer quantitative behaviours. This manuscript aims at studying a combination of reachability objective with the total-payoff metrics, i.e. *shortest-path games* where one player wants to reach a target of the graph while minimising the cumulated weight. Our study starts with the case of finite graph games, with the search for efficient algorithms to compute optimal values and strategies. We then continue by adding clocks and time constraints in the games, in order to design efficient programs satisfying some real-time specifications. This leads to a notoriously difficult class of games where undecidability emerges very quickly. Our work consists into the search for fragments with decidable results, either by limiting the number of clocks, or the cyclic behaviours in the game. A last chapter also studies the use of stochasticity in strategies in order to limit the amount of memory needed for players to play optimally.

Contents

Introduction

Formal methods to improve the quality of software

Software systems are ubiquitous, in our private equipments or industrial facilities, and sometimes perform as critical systems. Under these conditions, ensuring their reliability is crucial. Moreover, these systems must deal with larger and larger data, and must thus scale in terms of complexity. Finally, many of those systems operate in embedded hardware, where not only the speed of computation matters but also the energetic sobriety to work with limited batteries.

To solve these various issues, and bring reliable guarantees, formal methods have been used for many years, with great success, among them automata based techniques and the renowned model checking approach. Real-time and energy-aware extensions have also been studied thoroughly, allowing for studying and verifying time-sensitive embedded software.

Game theory for software synthesis

But verifying a model of a system requires that such a system has been designed previously, which is a harder and harder task, with the current complexity of the embedded systems. Another paradigm consists in automatically synthesise such software, that is correct by construction.

One of the famous techniques to do so consists in game theory, already theorised by John Nash for economic purposes. In computer science, two players, the *controller* and an *environment*, are fighting one against the other in a zero-sum game played on a graph of all possible configurations of the system. A *winning strategy* for the controller results in a correct program, while the environment is a player modelling all *uncontrollable* events that the program must face: signals incoming from the actual environment (human users of the program, weather conditions, etc.), uncertainty of the physical parameters, or exact timing of certain actions performed by the program. Many possible objectives have been studied in such two-player zero-sum games played on graphs: reachability of a target state, safety from a bug state, repeated reachability to model liveness of a program, and even all possible ω-regular objectives [GTW02].

Apart from such *qualitative* objectives, more *quantitative* ones are useful in order to select a particular strategy/program among all the ones that are correct with respect to a qualitative objective. Some metrics of interest, mostly studied in the quantitative game theory literature, are *mean-payoff*, *discounted-payoff*, or *total-payoff*. All these objectives have in common that, in finite games, both players have strategies using no memory or randomness to win or play optimally [GZ04].

Combining quantitative and qualitative objectives, enabling to select a good strategy among the valid ones for the selected metrics, often leads to the need of memory to play optimally.

Content of this manuscript

One of the most fundamental combinations showing this need for memory, and that will be thoroughly studied in this manuscript, consists in the *shortest-path games* combining a reachability objective with a total-payoff quantitative objective: they have been studied in [Kha+08] in the case of only non-negative weights, and we will further study this model in presence of negative weights in Chapter 1. Another case of interest is the combination of a parity qualitative objective (modelling every possible ω-regular condition), with a mean-payoff objective (aiming for a controller of good quality in the average long-run), where controllers need memory, and even infinite memory, to play optimally [CHJ05].

Chapter 1 also presents the use of techniques for shortest-path games in order to solve efficiently total-payoff games with both positive and negative weights.

Game-theoretic methods for controller synthesis is even more crucial in a real-time setting, where the design of programs satisfying some real-time specifications remains a notoriously difficult problem, prone to many bugs. Formal methods have incorporated the notion of time, by considering *timed automata* [AD94] that extend finite state automata with timing constraints, providing an automata-theoretic framework to model and verify real-time systems. While this has lead to the development of mature *verification* tools, the synthesis requires to extend this setting to the two-player game-theoretic case. Reachability timed games have thus been introduced, and shown decidable [AM99], and even EXPTIME-complete [JT07].

For many real-time applications, this qualitative setting is again too coarse to model faithfully the system, which again motivates a shift to a quantitative setting. Weighted extensions of timed automata [Beh+01; ALP04] and timed games [Bou+04a; ABM04] have thus been considered in order to measure the quality of the winning strategy for the controller. There, weights are twofold: transitions are equipped with weights as in the untimed setting, but locations (i.e. states) are also equipped with rates of weights, the cost being then proportional to the time spent in this location, with the rate as proportional coefficient. In this setting, the possibility to use negative weights on transitions and locations is crucial when one wants to model energy or other resources that can grow or decrease (linearly with respect to the time) during the execution of the system under study. We will recall useful definitions and classical techniques like the region or corner-point abstractions in Chapter 2.

While solving the optimal reachability problem on weighted timed automata has been shown to be PSPACE-complete [Bou+07] (i.e. the same complexity as the non-weighted version), weighted timed games are known to be undecidable [BBR05]. Several restrictions have then been considered in order to regain decidability. A first one consists in limiting the number of clocks of the underlying timed automaton. When in presence of only non-negative weights (both in transitions and locations) and with a single clock, weighted timed games become decidable [Bou+06], with an exponential-time complexity [Rut11; HIM13]. We will present our extension of this positive result to the case where negative weights are allowed in Chapter 3. Technical restrictions have to be considered limiting the presence of resets in some cycles of the game, letting the unconstrained case of weighted timed games with one clock still open. We are currently investigating this problem, with Julie Parreaux and Pierre-Alain Reynier, hoping to close the decidability gap (since the problem indeed becomes undecidable with two clocks in the negative setting [Bri+14], contrary to the non-negative case where the undecidability is only known with at least

three clocks).

Another very interesting restriction to recover decidability consists in not restraining the number of clocks, but the weights allowed on cycles of the game. The class of *strictly non-Zeno cost* weighted timed games with only non-negative weights has thus been introduced and studied [Bou+04a]: this hypothesis requires 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). In this context, applying unfolding techniques and using results of [ABM04], the class becomes decidable. Chapter 4 aims at presenting our work to extend this setting in the presence of negative weights: in the so-called *divergent* weighted timed games, each execution that follows a cycle (of the region automaton) has a weight in $(-\infty, -1] \cup [1, +\infty)$. We proposed a triply-exponential-time algorithm allowing one to compute the values and almost-optimal strategies, while deciding the divergence is PSPACE-complete.

Even in the case where weighted timed games are undecidable, for instance for non-divergent ones with at least two clocks, it is tempting to still aim for some positive results. Such research direction has been introduced in [BJM15] where authors relax the strictly non-Zeno cost condition (for games with only non-negative weights) to allow for cycles of weight exactly 0 (while still preventing those of weight arbitrarily close to 0). For such games, the value cannot be computed precisely in most generality (since the corresponding decision problem is shown undecidable), but the value can be *approximated*. We present, also in Chapter 4, a highly non-trivial extension of this line of work in presence of negative weights.

As said earlier, in shortest-path games with both positive and negative weights, it is known that the player desiring to reach the target requires memory to play optimally. This is of course even more true in the context of weighted timed games. More precisely, the memory needed is pseudo-polynomial for untimed games (i.e. polynomial if constants are encoded in unary) and even exponential for timed games, which can be prohibitive from an applicability point of view. An important challenge is thus to find ways to avoid using such complex strategies, e.g. by proposing alternative classes of strategies that are more easily amenable to implementation.

One way to do so, that we explore in Chapter 5 is to trade memory for randomness. Such trade-off is inspired by some work on Street or Müller games, both in the untimed [CAH04] or timed setting [CHP08a]. Memory or randomness is also crucial in multi-dimensional objectives [CRR14]: for instance, in mean-payoff parity games, if there exists a deterministic finite-memory winning strategy, then there exists a randomised memoryless almost-sure winning strategy. In our context of shortest-path games, we explain how to trade memory for randomness, by using the particular shape of optimal strategies for the player who wants to reach the target: they can always search for negative cycles of the game, play for a sufficiently long time in such cycles if possible, before switching to a strategy attracting the game to the target. Optimal strategies are thus obtained by the combination of two memoryless strategies. Randomisation is thus done, roughly speaking, by choosing between the two memoryless strategies with a well-chosen distribution of probabilities.

To lift such randomisation results in the timed setting, we first propose an original definition of the expected payoff, by considering necessary conditions on the strategies for this expectation to be well-defined (intuitively, the controller not only must reach the

target, they must do it quickly enough). We strongly rely on the presence of switching strategies, that we only know of in divergent weighted timed games, the setting that we thus consider in the second part of Chapter 5. It has to be noticed that randomisation only comes from the strategies of both players: in the presence of randomness also in the game itself, the situation becomes much more complex, even undecidable with at least two clocks [BCJ09].

Biographical sketch since the PhD

My PhD thesis, defended in October 2013, addressed the specification of quantitative properties with logical and automata frameworks. This manuscript presents some work I achieved since the beginning of my post-doctorate position in Université libre de Bruxelles, and then as a Maître de Conférences in Aix-Marseille Université, with many co-authors. I refer to the corresponding published articles in the beginning of each chapter.

For space issues, I decided not to present some other results obtained during this period, that I hereby shortly summarise.

— With Serge Haddad [HM15; HM18], we proposed a new algorithm, called interval iteration, to improve the usual value iteration technique for Markov chains and (interval) Markov decision processes.

— With Thomas Brihaye, Gilles Geeraerts, Axel Haddad, Guillermo A. Pérez, and Gabriel Renault [Bri+15c], we investigated the presence of failures in quantitative games.

— With Thomas Brihaye, Morgane Estiévenart, Gilles Geeraerts, Hsi-Ming Ho, and Nathalie Sznajder [Bri+16b], we studied real-time synthesis based on the MITL logic, and explained why this is a hard problem, and under which assumptions it can become tractable.

— With Thomas Brihaye, Gilles Geeraerts, Hsi-Ming Ho, and Arthur Milchior [Bri+18], we proposed a new tool to translate MITL formula into timed automata, in a compositional fashion, both in the classical timed-words semantics [Bri+17b], and in the signal semantics [Bri+17c].

— With Damien Busatto-Gaston, Pierre-Alain Reynier, and Ocan Sankur [Bus+19], we synthesised robust controllers in timed Büchi automata.

— With Théodore Lopez, and Jean-Marc Talbot [LMT19], we determinised finitely-ambiguous copyless cost register automata.

— With Thomas Brihaye, Gilles Geeraerts, Marion Hallet, and Bruno Quoitin [Bri+19], we initiated a work on the dynamics in games, using simulation-based techniques with an application to routing.

1 Shortest-Path and Total-Payoff Games

This chapter presents some results obtained with Thomas Brihaye, Damien Busatto-Gaston, Amit Kumar Dhar, Gilles Geeraerts, Axel Haddad, Engel Lefaucheux and Pierre-Alain Reynier, published in the conferences CONCUR [Bri+15b] and FoSSaCS [BMR17], the journal Acta Informatica [Bri+17a], the workshop Cassting [Bri+16a], as well as a submitted journal article [Bri+21].

Table of contents

Games played on graphs are nowadays a well-studied and well-established model for the computer-aided design of computer systems, as they enable *automatic synthesis* of systems that are *correct-by-construction*. Of particular interest are *quantitative games*, that allow one to model precisely *quantitative* parameters of the system, such as energy consumption. In this setting, the game is played by two players on a directed weighted graph, where the edge weights model, for instance, a cost or a reward associated with the moves of the players. Each vertex of the graph belongs to one of the two players who compete by moving a token along the graph edges, thereby forming an infinite path called a *play*. With each play is associated a real-valued *payoff* computed from the sequence of edge weights along the play. The traditional payoffs that have been considered in the literature include total-payoff [GZ04], mean-payoff [EM79] and discounted-payoff [ZP96]. In this quantitative setting, one player aims at maximising the payoff while the other tries to minimise it. So one wants to compute, for each player, the best payoff that they [1] can guarantee from each vertex, and the associated optimal strategies (i.e. that guarantee the optimal payoff no matter how the adversary is playing).

Such quantitative games have been extensively studied in the literature. Their associated decision problems (*is the value of a given vertex above a given threshold?*) are known to be in $\mathsf{NP} \cap \mathsf{coNP}$. Mean-payoff games have arguably been best studied from the algorithmic point of view. A landmark is Zwick and Paterson's pseudo-polynomial time (i.e. polynomial in the weighted graph when weights are encoded in unary) algorithm [ZP96], using the *value iteration* paradigm that consists in computing a sequence of vectors of values that converges towards the optimal values of the vertices. After a fixed, pseudo-polynomial, number of steps, the computed values are precise enough to deduce the actual values of all vertices. Better pseudo-polynomial time algorithms have later been proposed, e.g., by

1. In this document, we will use the singular *they* pronouns to describe the players.

[BV07; Bri+11; CR15], also achieving sub-exponential expected running time by means of randomisation.

Our first objective in this chapter is to focus on *total-payoff games in the presence of both positive and negative weights*[2]. Given an infinite play ρ, we denote by $\rho[k]$ the prefix of ρ of length k, and by $\mathbf{TP}(\rho[k])$ the (finite) sum of all edge weights along this prefix. The *total-payoff* of ρ, $\mathbf{TP}(\rho)$, is the inferior limit of all those sums, i.e.

$$\mathbf{TP}(\rho) = \liminf_{k\to\infty} \mathbf{TP}(\rho[k])$$

Compared to mean-payoff (and discounted-payoff) games, the literature on total-payoff games is less extensive. Gimbert and Zielonka [GZ04] have shown that optimal memoryless strategies always exist for both players and the best algorithm known before our works to compute the values runs in exponential time [GS09], and consists in iteratively improving strategies. Other related works include *energy games* where one player tries to optimise its energy consumption (computed again as a sum), keeping the energy level always above 0. Note that it differs in essence from total-payoff games where no condition on the energy level is required: in particular, the optimal total-payoff could be negative, and even $-\infty$, and it is a priori not possible to simply lift all the weights by a constant to solve total-payoff games by solving a related energy games. Moreover, this difference makes difficult to apply techniques solving energy games in the case of total-payoff games. Probabilistic variants of total-payoff games have also been studied, but the weights are restricted to be non-negative [Che+13].

We argue that the total-payoff objective is interesting as a *refinement* of the mean-payoff. Indeed, the total-payoff is finite if and only if the mean-payoff is null. Then, the computation of the total-payoff enables a finer, two-stage analysis of a game $\mathcal{G}$: (i) compute the mean payoff $\mathbf{MP}(\mathcal{G})$; (ii) subtract $\mathbf{MP}(\mathcal{G})$ from all edge weights, and scale the resulting weights if necessary to obtain integers. At that point, one has obtained a new game $\mathcal{G}'$ with null mean-payoff; (iii) compute $\mathbf{TP}(\mathcal{G}')$ to *quantify the amount of fluctuation around the mean-payoff* of the original game. Unfortunately, no efficient (i.e. polynomial, or at least pseudo-polynomial time) algorithms for total-payoff games was known before our works, and straightforward adaptations of Zwick and Paterson's value iteration algorithm for mean-payoff do not work, as we demonstrate at the end of Section 1.1. Our goal was to fill in this gap by introducing the first pseudo-polynomial time algorithm for computing the values in total-payoff games.

Our solution is a non-trivial value iteration algorithm that proceeds through nested fixed points (see Algorithm 2). A play of a total-payoff game is infinite by essence. We transform the game so that one of the players (the minimiser) must ensure a *reachability objective*: we assume that the game ends once this reachability objective has been met. The intuition behind this transformation, that stems from the use of an inferior limit in the definition of the total-payoff, is as follows: in each play ρ whose total-payoff is *finite*, there is a position ℓ in the play after which all the partial sums $\mathbf{TP}(\rho[i])$ (with $i \geqslant \ell$) will be larger than or equal to the total-payoff $\mathbf{TP}(\rho)$ of ρ, and infinitely often both will be equal. For example, consider the game depicted in Figure 1.1(a), where the maximiser player (henceforth called Max) plays with the rectangular vertices and the minimiser (Min) with the circular vertices. For both players, the optimal value when

2. Note that those games are different from *total-reward games* as studied in [TV87].

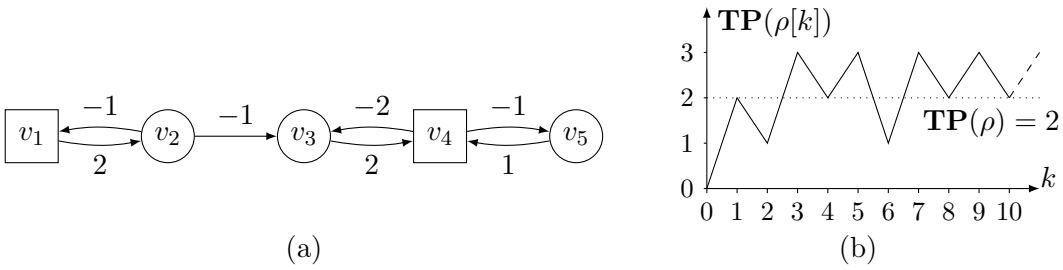

Figure 1.1 – (a) A total-payoff game, and (b) the evolution of the partial sums in ρ.

playing from v_1 is 2, and the play $\rho = v_1v_2v_3\ v_4v_5\ v_4v_3\ (v_4v_5)^\omega$ reaches this value (i.e. $\mathbf{TP}(\rho) = 2$). Moreover, for all $k \geqslant 7$: $\mathbf{TP}(\rho[k]) \geqslant \mathbf{TP}(\rho)$, and infinitely many prefixes $(\rho[8], \rho[10], \rho[12], \ldots)$ have a total-payoff of 2, as shown in Figure 1.1(b).

Based on this observation, we transform a total-payoff game $\mathcal{G}$, into a new game that has the same value as the original total-payoff game but incorporates a reachability objective for Min. Intuitively, in this new game, we allow a new action for Min: after each play prefix $\rho[k]$, they can ask to *stop the game*, in which case the payoff of the play is the payoff $\mathbf{TP}(\rho[k])$ of the prefix. However, allowing Min to stop the game at every moment would not allow us to obtain the same value as in the original total-payoff game: for instance, in the example of Figure 1.1(a), Min could secure value 1 by asking to stop after $\rho[2]$, which is strictly smaller than the actual total-payoff (2) of the whole play ρ. So, we allow Max to *veto* to stop the game, in which case both must go on playing. Again, allowing Max to turn down all of Min's requests would be unfair, so we parameterise the game with a natural number K, which is the maximal number of vetoes that Max can ask (and we denote by $\mathcal{G}^K$ the resulting game). For the *play* depicted in Figure 1.1(b), letting $K = 3$ is sufficient: trying to obtain a better payoff than the optimal, Min could request to stop after $\rho[0]$, $\rho[2]$ and $\rho[6]$, and Max can veto these three requests. After that, Max can safely accept the next request of Min, since the total payoff of all prefixes $\rho[k]$ with $k \geqslant 6$ are larger than or equal to $\mathbf{TP}(\rho) = 2$. Our key technical contribution is to show that *for all total-payoff games, there exists a finite, pseudo-polynomial, value of K such that the values in $\mathcal{G}^K$ and $\mathcal{G}$ coincide* (assuming all values are finite in $\mathcal{G}$: we treat values $+\infty$ and $-\infty$ separately). Now, assume that, when Max accepts to stop the game (possibly because they have exhausted the maximal number K of vetoes), the game moves to a *target vertex*, and stops. By doing so, we effectively reduce the computation of the values in the total-payoff game $\mathcal{G}$ to the computation of the values in the total-payoff game $\mathcal{G}^K$ *with an additional reachability objective* (the target vertex) for Min.

In the following, such refined total-payoff games—where Min *must* reach designated target vertices—will be called *games with shortest-path-payoff*, shortly *shortest-path games*, (originally called *min-cost reachability games* in [Bri+15b; Bri+17a] and simply *weighted games* in [BMR17]). Failing to reach the target vertices is the worst situation for Min, so the payoff of all plays that do not reach the target is $+\infty$, irrespective of the weights along the play. Otherwise, the payoff of a play is the sum of the weights up to the first occurrence of the target. As such, this problem nicely generalises the classical shortest-path problem in a weighted graph. This class of games is indeed very natural and interesting in its own, especially in the presence of both positive and negative weights. They represent a traditional extension of (qualitative) games with reachability objectives, and may therefore

model many situations interesting in practice. We have demonstrated this by applying this framework to the efficient energy distribution in a smart grid [Bri+16a]. We will also see in next chapters how to make shortest-path games even more useful, by adding real-time constraints.

In the one-player setting (considering the point of view of Min for instance), shortest-path games can be solved in polynomial time by Dijkstra's and Floyd-Warshall's algorithms when the weights are non-negative and arbitrary, respectively. Khachiyan et al [Kha+08] have proposed an extension of Dijkstra's algorithm to handle the two-player case with only non-negative weights. However, in our more general setting (two players, arbitrary weights), this problem had never been studied as such, except that the associated decision problem was known to be in NP ∩ coNP [FGR12]. [3]

Thus, as a second contribution in this chapter, we present a *value iteration* algorithm enabling us to compute in pseudo-polynomial time the values of a shortest-path game. In addition, we show how to compute optimal strategies for both players and characterise them: there is always a memoryless strategy for Max, but we exhibit an example (see Figure 1.2) where Min needs (finite) memory. Those results on shortest-path games are exploited in Section 1.3 where we present our efficient algorithm for total-payoff games. We briefly present a prototype implementation in Section 1.4, using as a core the numerical model-checker PRISM. This allows us to describe some heuristics able to improve the practical performances of our algorithms for total-payoff games and shortest-path games on certain subclasses of graphs.

As a last contribution, we introduce in Section 1.5 the class of *divergent* shortest-path games, for which we are able to compute the values and optimal strategies *in polynomial time*. Divergence is inspired by the *strictly non-Zeno cost* property introduced in [Bou+04b] to obtain the decidability of the same problems in weighted timed games, that we will study in the next chapters. It consists in intuitively ensuring that long enough plays will have a value that *diverges* towards $+\infty$ or $-\infty$: for that, we require the absence of cycles of accumulated weight 0. We can even extend slightly the result by requiring *almost-divergence*, meaning that every cycle of weight 0 must only contain smaller cycles of weight 0: therefore, we allow for cycle of weight 0 as long as they are not obtained by alternating positive and negative cycles. This paves the way of Chapter 4 where similar fragments are studied as a way to compute (for divergent games) or approximate (for almost-divergent games) the value of weighted *timed* games.

1.1 Weighted games with arbitrary weights

In this section, we formally introduce the (untimed) game model we consider throughout this manuscript. We denote by $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{R}_{\geqslant 0}$ the set of natural numbers, integers, rational and real numbers, and non-negative real numbers respectively. We let $\mathbb{Z}_\infty = \mathbb{Z} \cup \{-\infty, +\infty\}$ and $\mathbb{R}_\infty = \mathbb{R} \cup \{-\infty, +\infty\}$. The set of vectors indexed by V with values in S is denoted by S^V: the components of such a vector $x \in S^V$ are written x_v or $x(v)$ depending on the context. We let $\preccurlyeq$ be the pointwise order over ${\mathbb{R}_\infty}^V$, where $x \preccurlyeq y$

3. A pseudo-polynomial time algorithm to solve a very close problem, called the *longest shortest path problem* has been introduced by [BV07] to eventually solve mean-payoff games. However, because of this peculiar context of mean-payoff games, their definition of the length of a path differs from our definition of the payoff and their algorithm can not be easily adapted to solve our shortest-path problem.

if and only if $x(v) \leqslant y(v)$ for all $v \in V$.

We consider two-player turn-based games played on weighted graphs and denote the two *players* by Max and Min.

Definition 1.1. A *weighted game* is a tuple $\langle V_{\mathsf{Min}}, V_{\mathsf{Max}}, E, \mathsf{wt}\rangle$ where $V = V_{\mathsf{Min}} \uplus V_{\mathsf{Max}}$ is a set of vertices partitioned into the sets V_{Min} and V_{Max} of Min and Max respectively, $E \subseteq V \times \Sigma \times V$ is a set of *directed labelled edges* such that for all vertices v and labels a, there is at most one edge $(v, a, v') \in E$, and $\mathsf{wt}\colon E \to \mathbb{Z}$ is the *weight function*, associating an integer weight with each edge. The weighted game is called *finite* if V and Σ are finite.

In our drawings, Max vertices are depicted by rectangles; Min vertices by circles. In this chapter, we will not draw labels on edges, for simplicity, and will more generally suppose that the label of the edge (v, a, v') is always $a = v'$ so that we remove them from the presentation[4]. For every vertex $v \in V$, the set of labels of outgoing edges is denoted by $\Sigma(v) = \{a \in \Sigma \mid \exists v' \in V, (v, a, v') \in E\}$, and the set of successor vertices by $E(v) = \{v' \in V \mid \exists a \in \Sigma, (v, a, v') \in E\}$. We will denote an edge (v, a, v') with the arrow $v \xrightarrow{a} v'$. Finally, we let $W = \max_{e\in E} |\mathsf{wt}(e)|$ be the greatest edge weight (in absolute value) in the game graph.

A *finite play* ρ is a finite path in the graph of the game, i.e. $\rho = v_0 \xrightarrow{a_0} v_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{k-1}} v_k$ such that for all $0 \leqslant i < k$, $v_i \xrightarrow{a_i} v_{i+1} \in E$. We let FPlays be the set of finite plays, and $\mathsf{FPlays}^{\mathsf{Min}}$ (resp. $\mathsf{FPlays}^{\mathsf{Max}}$) the subset of finite plays ending in a vertex of Min (resp. Max). A *play* is a maximal (i.e. infinite or that cannot be extended by one more edge) sequence $\rho = v_0 \xrightarrow{a_0} v_1 \xrightarrow{a_1} \cdots$ such that every finite prefix $v_0 \xrightarrow{a_0} v_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{k-1}} v_k$, denoted by $\rho[k]$, is a finite play. We let Plays be the set of plays.

The *cumulated weight* of a finite play $\rho = v_0 \xrightarrow{a_0} v_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{k-1}} v_k$ is obtained by summing up the weights along ρ, i.e.

$$\mathsf{wt}(\rho) = \sum_{i=0}^{k-1} \mathsf{wt}(v_i, a_i, v_{i+1})$$

Based on this simple definition, we may define three classical payoff functions. In order for the two first definitions to define a payoff to all plays, we forbid maximal plays that would be finite. This is done by classically assuming that the games we consider are deadlock-free, i.e. for all vertices v, $\Sigma(v) \neq \emptyset$.

— The *total-payoff* of an (infinite) play ρ is given by[5]

$$\mathbf{TP}(\rho) = \liminf_{k\to\infty} \mathsf{wt}(\rho[k])$$

— The *mean-payoff* computes the average weight of ρ, i.e.

$$\mathbf{MP}(\rho) = \liminf_{k\to\infty} \frac{\mathsf{wt}(\rho[k])}{k}$$

4. We keep labels in the definition since we will give the semantics of *timed* games in the next chapter, by using the labels as a way to give the time spent in a location, as well as the chosen transition.

5. Our results can easily be extended by substituting a lim sup for the lim inf. The lim inf is more natural since we adopt the point of view of the maximiser Max, where the lim inf is the *worst* partial sum seen infinitely often.

— The *shortest-path-payoff* (we will often shorten this to *shortest-path*, in particular speaking about *shortest-path games*) of a play $\rho = v_0 \xrightarrow{a_0} v_1 \xrightarrow{a_1} \cdots$ computes the cumulated weight before reaching a given set of target vertices $V_t \subseteq V$. In this context, we suppose that V is the disjoint union of V_{Min}, V_{Max} and V_t, and that V_t are the only deadlock vertices of the game: therefore, a play is finite if and only if it visits (and then ends) in a target vertex. Formally, the shortest-path-payoff of a play ρ is given by

$$\mathbf{SP}(\rho) = \begin{cases} +\infty & \text{if } \rho \text{ is infinite} \\ \mathsf{wt}(\rho[k]) & \text{if } v_k \in V_t \end{cases}$$

The choice to give a payoff $+\infty$ to a play that does not reach the target is a corollary of the decision to give the reachability of the game to Min, that wants to *minimise* the payoff: therefore, not reaching the target is necessarily the worst possible situation for Min.

A *strategy* for Min (resp. Max) is a recipe dictating how to play, i.e. a mapping $\sigma \colon \mathsf{FPlays}^{\mathsf{Min}} \to \Sigma$ (resp. $\sigma \colon \mathsf{FPlays}^{\mathsf{Max}} \to \Sigma$) such that for all finite plays $\rho = v_0 \xrightarrow{a_0} v_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{k-1}} v_k$ in $\mathsf{FPlays}^{\mathsf{Min}}$ (resp. $\mathsf{FPlays}^{\mathsf{Max}}$), $\sigma(\rho) \in \Sigma(v_k)$. When we want to emphasise on the players, we will denote σ_{Min} and σ_{Max} the strategies of Min and Max, respectively. Moreover, we let Σ_{Min} and Σ_{Max} be the sets of strategies of Min and Max, respectively. A (finite or infinite) play $\rho = v_0 \xrightarrow{a_0} v_1 \xrightarrow{a_1} \cdots$ conforms to a strategy σ of Min (resp. σ of Max) if for all k such that $v_k \in V_{\mathsf{Min}}$ (resp. $v_k \in V_{\mathsf{Max}}$), we have that $a_k = \sigma(\rho[k])$.

A strategy σ is *memoryless* if for all finite plays ρ, ρ' ending in the same vertex, $\sigma(\rho) = \sigma(\rho')$. It is then sufficient to describe the value of σ on vertices only (and not all finite plays). A strategy σ is said to be *finite-memory* if it can be described in a memory with a finite number of bits. A usual formal definition goes through an encoding with deterministic Moore machines. We will not use it in this manuscript but we refer interested readers to the definition we use in [Bri+17a].

For all strategies σ_{Min} and σ_{Max}, for all vertices v, we let $\mathsf{play}(v, \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}})$ be the outcome of σ_{Min} and σ_{Max}, defined as the unique play conforming to σ_{Min} and σ_{Max} and starting in v. Naturally, the objective of Max is to maximise the payoff. In this model of zero-sum game, Min then wants to minimise the payoff. We let $\mathsf{Val}(v, \sigma_{\mathsf{Max}})$ and $\mathsf{Val}(v, \sigma_{\mathsf{Min}})$ be the respective values of the strategies, being the payoff obtained as a best-response of the opponent. Formally, they are defined as

$$\begin{cases} \mathsf{Val}(v, \sigma_{\mathsf{Max}}) = \inf_{\sigma_{\mathsf{Min}} \in \Sigma_{\mathsf{Min}}} \mathbf{P}(\mathsf{play}(v, \sigma_{\mathsf{Max}}, \sigma_{\mathsf{Min}})) \\ \mathsf{Val}(v, \sigma_{\mathsf{Min}}) = \sup_{\sigma_{\mathsf{Max}} \in \Sigma_{\mathsf{Max}}} \mathbf{P}(\mathsf{play}(v, \sigma_{\mathsf{Max}}, \sigma_{\mathsf{Min}})) \end{cases}$$

where $\mathbf{P}$ is a payoff function, either $\mathbf{TP}$, $\mathbf{MP}$, or $\mathbf{SP}$. Each vertex v of the game is then associated with two possible values. The *upper value* of v is the best Min can hope for when choosing first their strategy:

$$\overline{\mathsf{Val}}(v) = \inf_{\sigma_{\mathsf{Min}} \in \Sigma_{\mathsf{Min}}} \mathsf{Val}(v, \sigma_{\mathsf{Min}})$$

while the *lower value* of v is the best Max can hope for:

$$\underline{\mathsf{Val}}(v) = \sup_{\sigma_{\mathsf{Max}} \in \Sigma_{\mathsf{Max}}} \mathsf{Val}(v, \sigma_{\mathsf{Max}})$$

We may easily see that $\underline{\mathsf{Val}} \preccurlyeq \overline{\mathsf{Val}}$. When needed, we may index the various sets and mappings associated with a game $\mathcal{G}$ by its name, for instance writing $\overline{\mathsf{Val}}_{\mathcal{G}}$ and $\underline{\mathsf{Val}}_{\mathcal{G}}$ for the values of the game. We say that strategies $\sigma^\star_{\mathsf{Max}}$ of Max and $\sigma^\star_{\mathsf{Min}}$ of Min are *optimal* if, for all vertices v, $\mathsf{Val}(v, \sigma^\star_{\mathsf{Max}}) = \underline{\mathsf{Val}}(v)$ and $\mathsf{Val}(v, \sigma^\star_{\mathsf{Min}}) = \overline{\mathsf{Val}}(v)$, respectively.

A game $\mathcal{G}$ is said to be *determined* when for all vertices v, its lower and upper values are equal. In that case, we write $\mathsf{Val}(v) = \underline{\mathsf{Val}}(v) = \overline{\mathsf{Val}}(v)$, and refer to it as the *value* of v in $\mathcal{G}$. Mean-payoff and total-payoff games are known to be determined, with the existence of optimal memoryless strategies [ZP96; GZ04]. Using an indirect consequence of Martin's theorem [Mar75], we can show with an ad-hoc proof that

Theorem 1.2. *Shortest-path games are determined, even if the set of vertices is infinite, countable or not.*

Proof. We reproduce here the proof of the unpublished article [Bri+21]. Consider a quantitative game $\mathcal{G}$ and a vertex $v \in V$. We will prove the determinacy result by using the Borel determinacy result of [Mar75]. First, notice that the payoff mapping $\mathbf{SP}$ is Borel measurable since the set of plays with finite shortest-path-payoff is a countable union of cylinders. Then, for an integer M, consider Win_M to be the set of plays with a payoff less than or equal to M. It is a Borel set, so that the qualitative game defined over the same graph with winning condition Win_M is determined. We now use this preliminary result to show our determinacy result.

We first consider cases where either the lower or the upper values is infinite. Suppose first that $\underline{\mathsf{Val}}(v) = -\infty$. We have to show that $\overline{\mathsf{Val}}(v) = -\infty$ too. Let M be an integer. Since $\underline{\mathsf{Val}}(v) < M$, we know that for all strategies σ_{Max} of Max, there exists a strategy σ_{Min} for Min, such that $\mathbf{SP}(\mathsf{play}(v, \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}})) \leqslant M$. In particular, Max has no winning strategies in the qualitative game equipped with Win_M as a winning condition, hence, by determinacy, Min has a winning strategy, i.e. a strategy σ_{Min} such that every strategy σ_{Max} of Max verifies $\mathbf{SP}(\mathsf{play}(v, \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}})) \leqslant M$. This exactly means that $\overline{\mathsf{Val}}(v) \leqslant M$. Since this holds for every value M, we get that $\overline{\mathsf{Val}}(v) = -\infty$. The proof goes exactly in a symmetrical way to show that $\overline{\mathsf{Val}}(v) = +\infty$ implies $\underline{\mathsf{Val}}(v) = +\infty$.

Consider then the case where both $\overline{\mathsf{Val}}(v)$ and $\underline{\mathsf{Val}}(v)$ are finite values. For the sake of contradiction, assume that $\underline{\mathsf{Val}}(v) < \overline{\mathsf{Val}}(v)$ and consider a real number r strictly in-between those two values. From $r < \overline{\mathsf{Val}}(v)$, we deduce that Min has no winning strategy from v in the qualitative game with winning condition Win_r. Identically, from $\underline{\mathsf{Val}}(v) < r$, we deduce that Max has no winning strategy from v in the same game. This contradicts the determinacy of this qualitative game. Hence, $\underline{\mathsf{Val}}(v) = \overline{\mathsf{Val}}(v)$. □

Natural problems on weighted games are of three kinds, considering here the point of view of Min though the same questions can be asked by taking the point of view of Max too:

— the *value problem*: given a game $\mathcal{G}$, an initial vertex v and a threshold $\alpha \in \mathbb{Z}_\infty$, do we have $\overline{\mathsf{Val}}(v) \leqslant \alpha$?

— the *existence problem*: given a game $\mathcal{G}$, an initial vertex v and a threshold $\alpha \in \mathbb{Z}_\infty$, does there exist a strategy σ_{Min} of Min such that $\mathsf{Val}(v, \sigma_{\mathsf{Min}}) \leqslant \alpha$?

— the *synthesis problem*: given a game $\mathcal{G}$, compute an optimal strategy for Min.

In the context of weighted games on a finite set of vertices, like this chapter focuses on (we now drop the labels from edges in particular), value and existence problems are almost

equivalent: in particular, we will see that infimum/supremum in the value definitions are indeed minimum/maximum, except if the values are $-\infty$ where Min will need an infinite sequence of strategies to target a value as low as they want. To decide the value and existence problems, we will rely on the *computation* of the value function, which will also enable the computation of optimal strategies, solving the synthesis problem too. The situation will be much different (and more complex) in the context of timed games (that have an infinite, uncountable, set of vertices) we will consider in the next chapters.

Before our contributions, total-payoff games have been mainly considered as a refinement of mean-payoff games [GZ04]. Indeed, if the mean-payoff value of a game is positive (resp. negative), its total-payoff value is necessarily $+\infty$ (resp. $-\infty$). When the mean-payoff value is 0 however, the total-payoff is necessarily different from $+\infty$ and $-\infty$, hence total-payoff games are particularly useful in this case, to refine the analysis of the game. Deciding the value problem for total-payoff and mean-payoff games can be achieved in $\mathsf{NP} \cap \mathsf{coNP}$ [ZP96]. Gawlitza and Seidl [GS09] refined the complexity to $\mathsf{UP} \cap \mathsf{coUP}$, and values are shown to be effectively computable solving nested fixed point equations with a strategy iteration algorithm working in exponential time in the worst case. Because of this strong relationship between mean-payoff games and total-payoff games, we can show that total-payoff games are, in some sense, as hard as mean-payoff games, for which the existence of a (strongly) polynomial time algorithm is a long-standing open question.

Our contribution is to improve on this state-of-the-art and present a pseudo-polynomial time algorithm for total-payoff games. In many cases (e.g. mean-payoff games), a successful way to obtain such a more efficient algorithm is the *value iteration paradigm*. Intuitively, value iteration algorithms compute successive approximations $x_0, x_1, \ldots, x_i, \ldots$ of the game value by restricting the number of turns that the players are allowed to play: x_i is the vector of optimal values achievable when the players play at most i turns. The sequence of values is computed by means of an operator $\mathcal{F}$, letting $x_{i+1} = \mathcal{F}(x_i)$ for all i. Good properties (Scott-continuity and monotonicity) of $\mathcal{F}$ ensure convergence towards its smallest or greatest fixed point (depending on the value of x_0), which, in some cases, is the value of the game.

Let us briefly explain why, unfortunately, a straightforward application of this approach fails with total-payoff games. In our case, the most natural operator $\mathcal{F}$ is such that

$$\mathcal{F}(X)(v) = \begin{cases} \max_{v' \in E(v)} \big(\mathsf{wt}(v,v') + X(v')\big) & \text{if } v \in V_{\mathsf{Max}} \\ \min_{v' \in E(v)} \big(\mathsf{wt}(v,v') + X(v')\big) & \text{if } v \in V_{\mathsf{Min}} \end{cases}$$

Indeed, this definition matches the intuition that X_N is the optimal value after N turns. Then, consider the example of Figure 1.1(a), limited to vertices $\{v_3, v_4, v_5\}$ for simplicity. Observe that there are two simple cycles with weight 0, hence the total-payoff value of this game is finite. Max has the choice between cycling into one of these two cycles. It is easy to check that Max's optimal choice is to enforce the cycle between v_4 and v_5, securing a payoff of -1 from v_4 (because of the $\liminf$ definition of **TP**). Hence, the values of v_3, v_4 and v_5 are respectively 1, -1 and 0. In this game, we have $\mathcal{F}(X) = \big(2 + X(v_4), \max(-2 + X(v_3), -1 + X(v_5)), 1 + X(v_4)\big)$, and the vector $(1, -1, 0)$ is indeed a fixed point of $\mathcal{F}$. However, it is neither the greatest nor the smallest fixed point of $\mathcal{F}$. Indeed, it is easy to check that, *if* X is a fixed point of $\mathcal{F}$, *then* $X + (\alpha, \alpha, \alpha)$ is also a fixed point, for all $\alpha \in \mathbb{Z} \cup \{-\infty, +\infty\}$. If we try to initialise

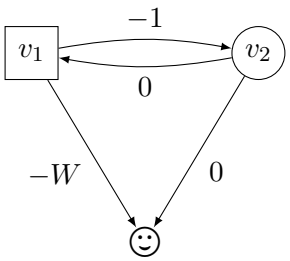

Figure 1.2 – A shortest-path game (with W a positive integer) where Min needs memory to achieve its optimal strategy.

the value iteration algorithm with value $(0, 0, 0)$, which could seem a reasonable choice, the sequence of computed vectors is: $(0, 0, 0)$, $(2, -1, 1)$, $(1, 0, 0)$, $(2, -1, 1)$, $(1, 0, 0)$, ... that is not stationary, and does not even contain $(1, -1, 0)$. Notice that $(-\infty, -\infty, -\infty)$ and $(+\infty, +\infty, +\infty)$ are fixed points, so that they do not allow us to find the correct answer too. Thus, it seems difficult to compute the actual game values with an iterative algorithm relying on the operator $\mathcal{F}$, as in the case of mean-payoff games. [6] Notice that, in the previous example, the Zwick and Paterson's algorithm [ZP96] to solve mean-payoff games would easily conclude from the sequence above, since the vectors of interest are then $(0, 0, 0)$, $(1, -0.5, 0.5)$, $(0.33, 0, 0)$, $(0.5, -0.25, 0.25)$, $(0.2, 0, 0)$, ... indeed converging towards $(0, 0, 0)$, the mean-payoff values of this game.

Instead, as explained in the introduction, we propose a different approach that consists in reducing total-payoff games to shortest-path games. The aim of the next section is to study these games, and we reduce total-payoff games to them in Section 1.3.

1.2 Shortest-path games

In this section, we solve shortest-path games. We start by studying a fundamental example giving hints about the difficulty of the problem.

Example 1.3. Consider the shortest-path game displayed in Figure 1.2, where W is a positive integer and ☺ is the target. We claim that the values of vertices v_1 and v_2 are both $-W$. Indeed, consider the following strategy for Min: during each of the first W visits to v_2 (if any), go to v_1; else, go to ☺. Clearly, this strategy ensures that the target will eventually be reached, and that either (i) the edge from v_1 to ☺ (with weight $-W$) will eventually be traversed; or (ii) the edge from v_1 to v_2 (with weight -1) will be traversed at least W times. Hence, in all plays following this strategy, the payoff will be at most $-W$. This strategy allows Min to secure $-W$, but they cannot ensure a lower payoff, since Max always has the opportunity to take the edge from v_1 to ☺ (with weight $-W$) instead of cycling between v_1 and v_2. Hence, Max's optimal choice is to follow this edge as soon as v_1 is reached, securing a payoff of $-W$. The strategy of Min we have just given is optimal, and there is *no optimal memoryless strategy* for Min. Indeed, in v_2,

6. In the context of stochastic models like Markov decision processes, [Str66] already noticed that in the presence of arbitrary weights, the value iteration algorithm does not necessarily converge towards the accurate value: [Put94] gives a more detailed explanation in Ex. 7.3.3.

always going to ☺ does not ensure a payoff at most $-W$; and, always going to v_1 does not guarantee to reach the target, and this strategy has thus value $+\infty$.

Shortest-path games (with both positive and negative weights) are one of the simplest games where one of the player needs memory to play optimally. This need stems from the combination of two objectives, a reachability one and a total-payoff one, that is one of the possible known sources of the need for memory.

We now show how to solve those games, i.e. how to compute $\mathsf{Val}(v)$ for all vertices v, in pseudo-polynomial time:

Theorem 1.4. *Let $\mathcal{G}$ be a shortest-path game.*

1. *For all $v \in V$, deciding whether $\mathsf{Val}(v) = +\infty$ can be done in polynomial time.*
2. *For all $v \in V$, deciding whether $\mathsf{Val}(v) = -\infty$ is as hard as solving mean-payoff games, in $\mathsf{NP} \cap \mathsf{coNP}$ and can be achieved in pseudo-polynomial time.*
3. *Computing all values $\mathsf{Val}(v)$ (for $v \in V$), as well as optimal strategies (if they exist) for both players, can be done in (pseudo-polynomial) time $\mathcal{O}(|V|^2|E|W)$.*
4. *If $\mathsf{Val}(v) \neq -\infty$ for all vertices $v \in V$, then both players have optimal strategies. Moreover, Max always has a memoryless optimal strategy, while Min may require finite (pseudo-polynomial) memory in their optimal strategy.*
5. *In the presence of only non-negative weights, both players have memoryless optimal strategies, and the above complexity drops down to polynomial time.*

This shows that the value problem, the existence problem, and the synthesis problem can all be solved in pseudo-polynomial time, and even in polynomial time when weights are non-negative. We now give the ingredients needed to prove the theorem.

Values $+\infty$ To prove the first item of Theorem 1.4, it suffices to notice that vertices with value $+\infty$ are exactly those from which Min cannot reach the target. Therefore the problem reduces to deciding the winner in a classical reachability game, that can be solved in polynomial time [Tho95], using an *attractor* construction.

Observe that we can safely *remove* from the game graph all those vertices with value $+\infty$, without changing the values of the other vertices. Hence, when need be, we can assume that the shortest-path games we consider contain no vertices with value $+\infty$, as they can be removed by this polynomial-time preprocessing. In those games, one can construct in polynomial time a memoryless strategy, called *an attractor strategy*, ensuring to reach the target in less than $|V|$ steps from every vertex. In the following, we therefore assume that all vertices have a value different from $+\infty$.

Values $-\infty$ To prove the second item, we notice that vertices with value $-\infty$ in a shortest-path game are exactly those with a value < 0 in the mean-payoff game obtained by adding a self-loop of weight 0 on the target vertices.

On the other hand, we can show that every mean-payoff game can be transformed (in polynomial time) into a shortest-path game such that a vertex has value < 0 in the mean-payoff game if and only if the value of its corresponding vertex in the shortest-path game is $-\infty$. To do so, without loss of generality, we may suppose that the graph of

Algorithm 1: Value iteration for shortest-path games. Gray lines correspond to the computation of optimal strategies for both players

Input: Shortest-path game $\mathcal{G}$, W greatest weight in absolute value

1 **foreach** $v \in V$ **do**
2 **if** $v \in V_t$ **then** $\mathsf{X}(v) := 0$
3 **else** $\mathsf{X}(v) := +\infty$
4 **repeat**
5 $\mathsf{X}_{pre} := \mathsf{X}$
6 **foreach** $v \in V_{\mathsf{Max}}$ **do**
7 $\mathsf{X}(v) := \max_{v' \in E(v)} \left(\mathsf{wt}(v, v') + \mathsf{X}_{pre}(v')\right)$
8 $\sigma_{\mathsf{Max}}(v) := \operatorname{argmax}_{v' \in E(v)} \left(\mathsf{wt}(v, v') + \mathsf{X}_{pre}(v')\right)$
9 **foreach** $v \in V_{\mathsf{Min}}$ **do**
10 $\mathsf{X}(v) := \min_{v' \in E(v)} \left(\mathsf{wt}(v, v') + \mathsf{X}_{pre}(v')\right)$
11 **if** $\mathsf{X}(v) \neq \mathsf{X}_{pre}(v)$ **then**
12 $\sigma^1_{\mathsf{Min}}(v) := \operatorname{argmin}_{v' \in E(v)} \left(\mathsf{wt}(v, v') + \mathsf{X}_{pre}(v')\right)$
13 **if** $\mathsf{X}_{pre}(v) = +\infty$ **then** $\sigma^2_{\mathsf{Min}}(v) = \sigma^1_{\mathsf{Min}}(v)$
14 **foreach** $v \in V \setminus V_t$ **do**
15 **if** $\mathsf{X}(v) < -(|V| - 1)W$ **then** $\mathsf{X}(v) := -\infty$
16 **until** $\mathsf{X} = \mathsf{X}_{pre}$
17 **return** X

the game is bipartite, in the sense that $E \subseteq V_{\mathsf{Max}} \times V_{\mathsf{Min}} \cup V_{\mathsf{Min}} \times V_{\mathsf{Max}}$.[7] The associated shortest-path game is the same game with one more vertex ☺ that is the only target. We add all edges from v to ☺ with $v \in V_{\mathsf{Min}}$, of weight 0. The crucial property is that Min can guarantee a value $-\infty$ in this shortest-path game if and only if they can force a negative cycle of the original mean-payoff game (thus, if and only if the mean-payoff is < 0) as long as they want before reaching the target (which they can always do as long as they have the opportunity to play, reason why we first assumed the game to be bipartite).

Computing all values Now that we have discussed the case of vertices with value in $\{-\infty, +\infty\}$, let us present our core contribution on shortest-path games, which is a pseudo-polynomial time, value iteration algorithm to compute the values of those games. Note that this algorithm is correct even when some vertices have value in $\{-\infty, +\infty\}$, as we will argue later.

Our value iteration algorithm for shortest-path games is given in Algorithm 1. As it can be seen, this algorithm consists in computing a sequence of vectors X. Initially (line 1), $\mathsf{X}(v) = +\infty$ for all vertices but the target vertices v where $\mathsf{X}(v) = 0$. Then, a new value of X is obtained by optimising locally the value of each vertex, and changing to $-\infty$ the value $\mathsf{X}(v)$ of all vertices v such that the computed value $\mathsf{X}(v)$ has gone below a given threshold $-(|V| - 1)W$ (line 7). The following proposition states the correctness of Algorithm 1.

7. It suffices to add a vertex of the opponent in-between two vertices of the same player related by an edge: in this vertex, the opponent has no choice but to follow the transition chosen by the first player.

Proposition 1.5. *If a shortest-path game $\mathcal{G}$ is given as input (possibly with values $+\infty$ or $-\infty$), Algorithm 1 outputs the value vector, after at most $(2|V|-1)W|V|+2|V|$ iterations.*

To establish this proposition, we consider the sequence of values that vector X takes along the execution of the algorithm. The intuition behind this sequence is that *the i-th iterate is the value vector of the game if we impose that* Min *must reach the target within i steps* (and gets a payoff of $+\infty$ if they fail to do so). In order to formalise this intuition, we define, for a play $\rho = v_0 \xrightarrow{a_0} v_1 \xrightarrow{a_1} \cdots$:

$$\mathbf{SP}^{\leqslant i}(\rho) = \begin{cases} \mathbf{SP}(\rho) & \text{if } v_k \in V_t \text{ for some } k \leqslant i \\ +\infty & \text{otherwise} \end{cases}$$

and we then let

$$\overline{\mathsf{Val}}^{\leqslant i}(v) = \inf_{\sigma_{\mathsf{Min}} \in \Sigma_{\mathsf{Min}}} \sup_{\sigma_{\mathsf{Max}} \in \Sigma_{\mathsf{Max}}} \mathbf{SP}^{\leqslant i}(\mathsf{play}(v, \sigma_{\mathsf{Max}}, \sigma_{\mathsf{Min}}))$$

Successive values of $\overline{\mathsf{Val}}^{\leqslant i}$ can be obtained by the iterative application of an operator $\mathcal{F} \colon \mathbb{Z}_\infty^V \to \mathbb{Z}_\infty^V$ mapping every vector $X \in \mathbb{Z}_\infty^V$ to the vector $\mathcal{F}(X)$ defined, for all vertices v, by:

$$\mathcal{F}(X)(v) = \begin{cases} 0 & \text{if } v \in V_t \\ \max\limits_{v' \in E(v)} \big(\mathsf{wt}(v,v') + X(v')\big) & \text{if } v \in V_{\mathsf{Max}} \\ \min\limits_{v' \in E(v)} \big(\mathsf{wt}(v,v') + X(v')\big) & \text{if } v \in V_{\mathsf{Min}}\,. \end{cases} \tag{1.1}$$

Then, we can indeed show that for all vertices v:

$$\overline{\mathsf{Val}}^{\leqslant 0}(v) = \begin{cases} 0 & \text{if } v \in V_t \\ +\infty & \text{otherwise} \end{cases} \qquad \text{and} \qquad \forall i \geqslant 1 \quad \overline{\mathsf{Val}}^{\leqslant i} = \mathcal{F}(\overline{\mathsf{Val}}^{\leqslant i-1})$$

Moreover, the computation performed in each loop of Algorithm 1 is exactly the same as the one in $\mathcal{F}$. Therefore, by induction, $\overline{\mathsf{Val}}^{\leqslant i}$ is the vector X obtained after i iterations of the main loop.

Notice that operator $\mathcal{F}$ is monotonic (i.e. $\mathcal{F}(X) \preccurlyeq \mathcal{F}(Y)$ for all $X \preccurlyeq Y$), and that $\overline{\mathsf{Val}}^{\leqslant 1} = \mathcal{F}(\overline{\mathsf{Val}}^{\leqslant 0}) \preccurlyeq \overline{\mathsf{Val}}^{\leqslant 0}$, so that the sequence $(\overline{\mathsf{Val}}^{\leqslant i})_{i \geqslant 0}$ is non-increasing: for all $i \geqslant 0$, $\overline{\mathsf{Val}}^{\leqslant i+1} \preccurlyeq \overline{\mathsf{Val}}^{\leqslant i}$. Observe also that for all $v \in V$, for all $i \geqslant 0$, and for all strategies σ_{Min} and σ_{Max}:

$$\mathbf{SP}^{\leqslant i}(\mathsf{play}(v, \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}})) \geqslant \mathbf{SP}(\mathsf{play}(v, \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}}))$$

Indeed, if a target vertex is reached within i steps, then payoffs are equal. Otherwise, $\mathbf{SP}^{\leqslant i}(\mathsf{play}(v, \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}})) = +\infty$. Thus, for all $i \geqslant 1$,

$$\overline{\mathsf{Val}}^{\leqslant i} \succcurlyeq \overline{\mathsf{Val}} = \mathsf{Val}$$

The main question is now to characterise the limit of the sequence $(\overline{\mathsf{Val}}^{\leqslant i})_{i \geqslant 0}$ that is returned by the algorithm, and more precisely, to prove that it is the value Val of the game. Indeed, at this point, it would not be too difficult to show that Val is a fixed

point of the operator $\mathcal{F}$, but it would be more difficult to show that it is the *greatest* fixed point of $\mathcal{F}$, that is indeed the limit of the sequence $(\overline{\mathsf{Val}}^{\leqslant i})_{i\geqslant 0}$ (by Kleene's theorem, applicable since $\mathcal{F}$ is Scott-continuous). Instead, we need to study refined properties of this sequence, namely its stationarity and the speed of its convergence, to deduce that Val is the greatest fixed point of $\mathcal{F}$. Here are the main steps of the proof:

— The first $|V|+1$ iterations emulate the classical attractor computation: if $v \in V$ is a vertex discovered after $k \leqslant |V|$ steps of attractor, then for all $j < k$, we have $\overline{\mathsf{Val}}^{\leqslant j}(v) = +\infty$ and $\overline{\mathsf{Val}}^{\leqslant k}(v) \leqslant kW$. In particular, for all $v \in V$, $\overline{\mathsf{Val}}^{\leqslant |V|}(v) \leqslant |V|W$. Therefore, the presence of vertices with value $+\infty$ does not interfere.

— The sequence $(\overline{\mathsf{Val}}^{\leqslant i})_{i\geqslant 0}$ stabilises after $(2|V|-1)W|V|+|V|$ steps, when all values of the game are finite. Moreover, the limit value is equal to the vector Val of values of $\mathcal{G}$.

— If the game contains vertices with value $-\infty$, $\overline{\mathsf{Val}}^{\leqslant i}$ will not converge for these vertices, but when it reaches an integer below $-(|V|-1)W$, we are sure that its value is indeed $-\infty$, which proves correct the line 7 of the algorithm.

Example 1.6. We close this discussion on the computation of the values by an example of execution of Algorithm 1. Consider the shortest-path game in Figure 1.2. The successive values for vertices (v_1, v_2) computed by the value iteration algorithm are the following: $(+\infty, +\infty)$, $(+\infty, 0)$, $(-1, 0)$, $(-1, -1)$, $(-2, -1)$, $(-2, -2), \ldots, (-W, -W+1)$, $(-W, -W)$. This requires $2W$ steps to converge (hence a pseudo-polynomial time).

Strategies of Min As we have seen earlier, Min does not always have optimal memoryless strategies. However, we will see that one can always construct so-called *negative cycle strategies* (NC-strategies), which are memoryless strategies that have a meaningful structure for Min, in the sense that they allow them either: (i) to reach the target by means of a play whose value is lower than the value of the game; or (ii) to decrease arbitrarily the partial sums along the play, when it does not reach the target (in other words, the partial sums tend to $-\infty$ as the play goes on). So NC-strategies are not optimal in general as they do not guarantee to reach the target, but in this case they guarantee that Min will play consistently with their objective, by decreasing the value of the play prefixes.

Formally, an NC-strategy is a memoryless strategy σ_{Min} for player Min such that, for all cyclic finite plays $v_0 \to \cdots v_{k-1} \to v_k = v_0$ conforming to σ_{Min},

$$\mathsf{wt}(v_0, v_1) + \cdots + \mathsf{wt}(v_{k-1}, v_k) < 0$$

We define the *fake value*, $\mathsf{fake}(v, \sigma_{\mathsf{Min}})$, of an NC-strategy σ_{Min} as the supremum of the cumulated weight of the *finite* plays that conform to it and start in v:

$$\mathsf{fake}(v, \sigma_{\mathsf{Min}}) = \sup\left\{\mathbf{SP}(v_0 \to \cdots v_{k-1} \to v_k) \mid v_0 = v, v_k \in V_t, \forall v_i \in V_{\mathsf{Min}}, \sigma_{\mathsf{Min}}(v_i) = v_{i+1}\right\}$$

Notice that the fake value is not necessarily equal to the value of σ_{Min}, since, in the definition of the fake value, we only consider plays that *do reach* the target, and ignore those that do not (in the computation of the actual value of σ_{Min}, these plays would yield a payoff $+\infty$). However, the fake value of a vertex is always smaller than or equal to its value. We say that an NC-strategy is *fake-optimal* if its fake value is smaller than or equal to the

optimal value of the game from all vertices. In particular, if the optimal value of a vertex v is $-\infty$, the set $\{\mathbf{SP}(v_0 \to \cdots v_{k-1} \to v_k) \mid v_0 = v, v_k \in V_t, \forall v_i \in V_{\mathsf{Min}}, \sigma_{\mathsf{Min}}(v_i) = v_{i+1}\}$ is empty for all fake-optimal strategies σ_{Min}, hence the supremum of this set is indeed $-\infty$.

Example 1.7. On the game of Figure 1.2, the memoryless strategy σ_{Min} mapping v_2 to v_1 is an NC-strategy as the only two possible cycles $v_1 \to v_2 \to v_1$ and $v_2 \to v_1 \to v_2$ have weight -1. The value of σ_{Min} from v_2 is $+\infty$ as the play $v_2 \to v_1 \to v_2 \to v_1 \to v_2 \to \cdots$ agrees with it and does not reach the target, but the fake value of σ_{Min} is $-W$, since the play $v_2 \to v_1 \to ☺$ is the finite play that agrees with σ_{Min} with the biggest cumulated weight possible. Since we know that the actual optimal value is $-W$, the strategy σ_{Min} is fake-optimal.

By combining a fake-optimal NC-strategy with a memoryless strategy reaching target vertices, we may obtain optimal (when possible) finite-memory strategies of a very special shape. We call such strategies *switching strategies*, described by two deterministic memoryless strategies σ^1_{Min} and σ^2_{Min}, as well as a switching parameter $\alpha \in \mathbb{N}$. It allows us to build a (finite-memory) strategy σ_{Min} playing along σ^1_{Min}, until eventually switching to σ^2_{Min} when the length of the current finite play is greater than α.

Indeed, if we let σ^1_{Min} be a fake-optimal NC-strategy, and σ^2_{Min} be any strategy obtained from an attractor computation (thus reaching a target vertex in a most $|V|$ steps), then we can show that the switching strategy σ_{Min} described by $(\sigma^1_{\mathsf{Min}}, \sigma^2_{\mathsf{Min}}, (2W(|V|-1) + N)|V| + 1)$ guarantees a value $\mathsf{Val}(v, \sigma_{\mathsf{Min}}) \leqslant \max(-N, \mathsf{Val}(v))$. In particular, if the value of all vertices is finite, then one can construct an optimal finite-memory strategy by considering $N = -(|V|-1)W$. If the value of a vertex is $-\infty$, this also says that there is an infinite family of strategies that allows one to secure a value which is arbitrarily low (remember that, by definition, $-\infty$ cannot be the value that corresponds to a single strategy).

It only remains to explain how to build a fake-optimal NC-strategy σ^1_{Min}. Without loss of generality, we suppose that no vertices have value $+\infty$, since for these vertices, all strategies are equivalent. For vertices of value $-\infty$, we can obtain σ^1_{Min} as an optimal strategy for Min in the associated mean-payoff game (obtained by adding a self-loop of weight 0 on target vertices). Since the mean-payoff value is negative, this strategy guarantees that it does not reach a target, thus generating a fake value $-\infty$, equal to the optimal value of the vertex. Moreover, since it is a memoryless strategy, we know that, as soon as Max plays a memoryless strategy that necessarily reaches a cycle, this cycle must have a negative weight (at most the optimal value of the initial vertex): this strategy is thus a fake-optimal NC-strategy.

Consider finally the case where no vertices have value $+\infty$ or $-\infty$. For all vertices $v \in V_{\mathsf{Min}}$, we let $\sigma^1_{\mathsf{Min}}(v)$ be a vertex $v' \in E(v)$ that minimises $\mathsf{wt}(v, v') + \mathsf{Val}(v')$. We can show that this construction indeed yields a fake-optimal NC strategy σ^1_{Min}, by proving (by induction) that all cycles it creates have a negative weight and that if it reaches the target, it does so with a payoff at most the value of the initial vertex. Notice that this definition is exactly what is computed in line 12 of Algorithm 1. Note also that line 13 indeed computes an attractor strategy. This ends the search for strategies for Min.

Strategies of Max Let us now show that Max always has a *memoryless optimal strategy*. For vertices with value $+\infty$, we already know a memoryless optimal strategy for Max,

namely every strategy that remains outside of the attractor of the target vertices. For vertices with value $-\infty$, all strategies are equally bad for Max. We now explain how to define a memoryless optimal strategy σ_{Max} for Max in case of a game containing only finite values. For every finite play ρ ending in a vertex $v \in V_{\mathsf{Max}}$ of Max, we let $\sigma_{\mathsf{Max}}(\rho)$ be a vertex $v' \in E(v)$ that maximises $\mathsf{wt}(v, v') + \mathsf{Val}(v')$. This is clearly a memoryless strategy, and we can show that it is optimal. Notice that line 8 of Algorithm 1 indeed computes such a strategy σ_{Max}.

Non-negative case Finally, to show the last item of Theorem 1.4, notice that, in the absence of negative cycles, the (optimal) switching strategy obtained for Min cannot loop in negative cycles, and thus the first (memoryless) strategy σ^1_{Min} cannot cycle and thus must reach a target vertex in a most $|V| + 1$ edges. Therefore, the second strategy σ^2_{Min}, and the switch threshold, are never used, and Min has thus an optimal memoryless strategy. Moreover, consider $\overline{\mathsf{Val}}^{\leqslant |V|}$, the value with bounded horizon $|V|$. For every vertex v, we have $\overline{\mathsf{Val}}^{\leqslant |V|}(v, \sigma^1_{\mathsf{Min}}) = \mathsf{Val}(v)$, so that $\overline{\mathsf{Val}}^{\leqslant |V|} \preccurlyeq \mathsf{Val}$. As $\mathsf{Val} \preccurlyeq \overline{\mathsf{Val}}^{\leqslant i}$ holds for all $i \geq 0$, we have $\mathsf{Val} = \overline{\mathsf{Val}}^{\leqslant |V|}$, which means that Algorithm 1 converges in at most $|V|$ iterations, thus running in an overall polynomial time.

1.3 An efficient algorithm to solve total-payoff games

We now turn our attention back to total-payoff games (without reachability objective). Building on the results of the previous section, we present a pseudo-polynomial time algorithm for solving those games in the presence of arbitrary weights, thanks to a reduction from total-payoff games to shortest-path games. The shortest-path game produced by the reduction has size pseudo-polynomial in the size of the original total-payoff game. Then, we show how to compute the values of the total-payoff game without building the entire shortest-path game, and explain how to deduce memoryless optimal strategies from the computation of our algorithm.

Reduction to shortest-path games We provide a transformation from a total-payoff game $\mathcal{G}$ to a shortest-path game $\mathcal{G}^K$ (where K is a parameter in $\mathbb{N}$) such that the values of $\mathcal{G}$ can be extracted from the values in $\mathcal{G}^K$ (as formalised below). Intuitively, $\mathcal{G}^K$ simulates the game where players play in $\mathcal{G}$; Min may propose to stop playing and reach a fresh vertex ☺ acting as the target; Max can then accept, in which case we reach the target, or refuse at most K times, in which case the game continues. Structurally, $\mathcal{G}^K$ consists of a sequence of copies of $\mathcal{G}$ along with some new states that we now describe formally. For all $n \geqslant 1$, we define the shortest-path game $\mathcal{G}^n$ with vertices of Min being

$$V^n_{\mathsf{Min}} = \{(v, j) \mid 1 \leqslant j \leqslant n, v \in V_{\mathsf{Min}}\} \cup \{(\mathsf{stop}, v, j) \mid 1 \leqslant j \leqslant n, v \in V\}$$

vertices of Max being

$$V^n_{\mathsf{Max}} = \{(v, j) \mid 1 \leqslant j \leqslant n, v \in V_{\mathsf{Max}}\} \cup \{(\mathsf{veto}, v, j) \mid 1 \leqslant j \leqslant n, v \in V\}$$

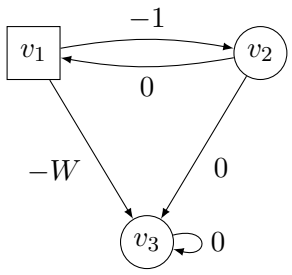

Figure 1.3 – A total-payoff game

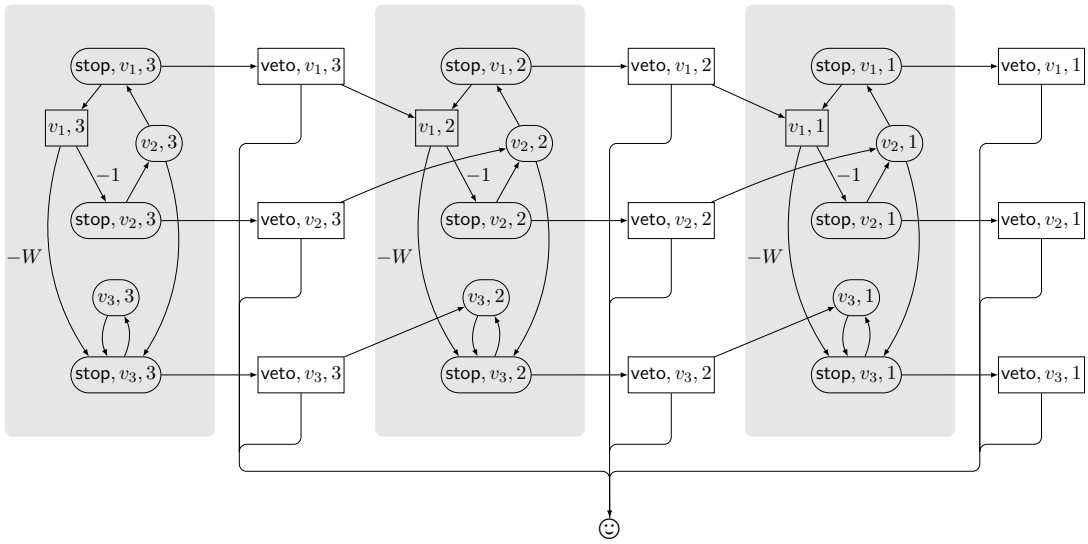

Figure 1.4 – Shortest-path game $\mathcal{G}^3$ associated with the total-payoff game of Figure 1.2

and edges given by

$$\begin{aligned} E^n = \quad & \big\{(v,j) \to (\mathsf{stop}, v', j) \mid v \to v' \in E, 1 \leqslant j \leqslant n\big\} \\ & \cup \big\{(\mathsf{stop}, v, j) \to (v, j) \mid v \in V, 1 \leqslant j \leqslant n\big\} \\ & \cup \big\{(\mathsf{stop}, v, j) \to (\mathsf{veto}, v, j) \mid v \in V, 1 \leqslant j \leqslant n\big\} \\ & \cup \big\{(\mathsf{veto}, v, j) \to (v, j-1) \mid v \in V, 1 < j \leqslant n\big\} \\ & \cup \big\{(\mathsf{veto}, v, j) \to \text{☺} \mid v \in V, 1 \leqslant j \leqslant n\big\} \end{aligned}$$

all of weight zero, except edges $(v,j) \to (\mathsf{stop}, v', j)$ that have weight $\mathsf{wt}(v, v')$.

Example 1.8. For example, considering the total-payoff game of Figure 1.3, the corresponding shortest-path game $\mathcal{G}^3$ is depicted in Figure 1.4 (where weights 0 have been removed).

The next proposition formalises the relationship between the two games:

Proposition 1.9. *Let $K = |V|(2(|V|-1)W+1)$. For all $v \in V$ and $k \geqslant K$,*

— *$\mathsf{Val}_{\mathcal{G}}(v) \neq +\infty$ if and only if $\mathsf{Val}_{\mathcal{G}}(v) = \mathsf{Val}_{\mathcal{G}^k}(v,k)$;*

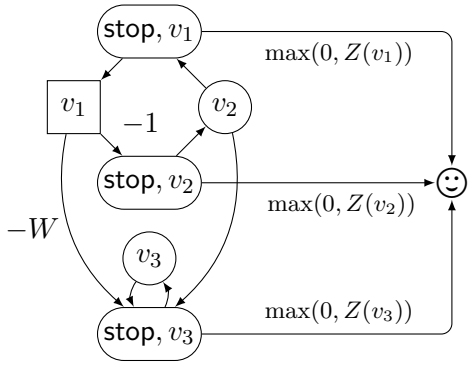

Figure 1.5 – Shortest-path game $\mathcal{G}_Z$ associated with the total-payoff game of Figure 1.2

— $\mathsf{Val}_{\mathcal{G}}(v) = +\infty$ *if and only if* $\mathsf{Val}_{\mathcal{G}^k}(v, k) \geqslant (|V| - 1)W + 1$.

The bound K is found by using the fact (informally described in the previous section) that if not infinite, the value of a shortest-path game belongs in $[-(|V|-1)\times W+1, |V|\times W]$, and that after enough visits of the same vertex, an adequate loop ensures that $\mathcal{G}^k$ verifies the above properties.

Value iteration algorithm for total-payoff games By Proposition 1.9, an immediate way to obtain a value iteration algorithm for total-payoff games is to build game $\mathcal{G}^K$, run Algorithm 1 on it, and map the computed values back to $\mathcal{G}$. We take advantage of the structure of $\mathcal{G}^K$ to provide a better algorithm that avoids building $\mathcal{G}^K$. Intuitively, we first compute the values of the vertices in the *last copy of the game* (vertices of the form $(v, 1)$, $(\mathsf{stop}, v, 1)$ and $(\mathsf{veto}, v, 1)$), then of those in the penultimate (vertices of the form $(v, 2)$, $(\mathsf{stop}, v, 2)$ and $(\mathsf{veto}, v, 2)$), and so on.

We sketch the intuitions that lead to the formalisation of these ideas. Let Z_j be a vector mapping each vertex v of $\mathcal{G}$ to the value $Z_j(v)$ of vertex (v, j) in $\mathcal{G}^K$. Then, we define an operator $\mathcal{H}$ such that $Z_{j+1} = \mathcal{H}(Z_j)$. Thus, assuming that we have computed the values of all vertices in the jth copy, $\mathcal{H}$ returns the value of the vertices in the $(j+1)$th copy. In other words, the definition of $\mathcal{H}(Z)$ for some vector Z is to extract from $\mathcal{G}^K$ one copy of the game (that we call $\mathcal{G}_Z$), and make Z appear in the weights of some edges as illustrated in Figure 1.5. This game, $\mathcal{G}_Z$, simulates a play in $\mathcal{G}$ in which Min can opt for 'stopping the game' at each round (by moving to the target), obtaining $\max(0, Z(v))$, if v is the current vertex. Then, we can define $\mathcal{H}(Z)(v)$ as the value of v in $\mathcal{G}_Z$. By construction, it is easy to see that $Z_{j+1} = \mathcal{H}(Z_j)$ holds for all $j \geqslant 1$, i.e. that $\mathcal{H}$ indeed corresponds to computing the values of the vertices in the $(j+1)$th copy, given the values in the jth copy. Furthermore, letting $Z_0(v) = -\infty$ for all v, and $Z_1 = \mathcal{H}(Z_0)$, we can prove that:

— $\mathcal{H}$ is monotonic (but may not be Scott-continuous);

— the sequence $(Z_j)_{j\geqslant 0}$ converges towards $\mathsf{Val}_{\mathcal{G}}$.

These arguments are the main ideas justifying Algorithm 2 to solve total-payoff games. Intuitively, the outer loop computes, in variable Y, a non-decreasing sequence of vectors whose limit is $\mathsf{Val}_{\mathcal{G}}$, and that is stationary (this is not necessarily the case for the sequence $(Z_j)_{j\geqslant 0}$). Line 1 initialises Y to Z_0. Each iteration of the outer loop amounts to running Algorithm 1 to compute $\mathcal{H}(\mathsf{Y}_{pre})$ (lines 10 to 13), then detecting if some vertices have

Algorithm 2: A value iteration algorithm for total-payoff games

Input: Total-payoff game $\mathcal{G}$, W largest weight in absolute value

1 **foreach** $v \in V$ **do** $\mathsf{Y}(v) := -\infty$
2 **repeat**
3 | **foreach** $v \in V$ **do** $\mathsf{Y}_{pre}(v) := \mathsf{Y}(v)$; $\mathsf{Y}(v) := \max(0, \mathsf{Y}(v))$; $\mathsf{X}(v) := +\infty$
4 | **repeat**
5 | | $\mathsf{X}_{pre} := \mathsf{X}$
6 | | **foreach** $v \in V_{\mathsf{Max}}$ **do** $\mathsf{X}(v) := \max_{v' \in E(v)} \left[\mathsf{wt}(v, v') + \min(\mathsf{X}_{pre}(v'), \mathsf{Y}(v'))\right]$
7 | | **foreach** $v \in V_{\mathsf{Min}}$ **do** $\mathsf{X}(v) := \min_{v' \in E(v)} \left[\mathsf{wt}(v, v') + \min(\mathsf{X}_{pre}(v'), \mathsf{Y}(v'))\right]$
8 | | **foreach** $v \in V$ such that $\mathsf{X}(v) < -(|V| - 1)W$ **do** $\mathsf{X}(v) := -\infty$
9 | **until** $\mathsf{X} = \mathsf{X}_{pre}$
10 | $\mathsf{Y} := \mathsf{X}$
11 | **foreach** $v \in V$ such that $\mathsf{Y}(v) > (|V| - 1)W$ **do** $\mathsf{Y}(v) := +\infty$
12 **until** $\mathsf{Y} = \mathsf{Y}_{pre}$
13 **return** Y

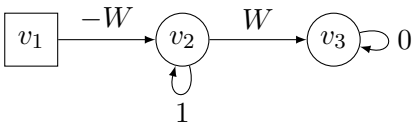

Figure 1.6 – An example total-payoff game where each execution of the inner loop requires two iterations while the outer loop takes W iterations to stabilise.

value $+\infty$, updating Y accordingly (line 11, following the second item of Proposition 1.9). It can then be shown that, for all $j > 0$, if we let Y_j be the value of Y after the jth iteration of the main loop, then $Z_j \preccurlyeq Y_j \preccurlyeq \mathsf{Val}_{\mathcal{G}}$, which ensures the correctness of the algorithm.

Proposition 1.10. *If a total-payoff game $\mathcal{G}$ is given as input, Algorithm 2 outputs the vector of optimal values, after at most $K = |V|(2(|V|-1)W+1)$ iterations of the external loop. The complexity of the algorithm is $\mathcal{O}(|V|^4|E|W^2)$.*

The number of iterations in each internal loop is controlled by Theorem 1.4. On the example of Figure 1.2, only 2 external iterations are necessary, but the number of iterations of each internal loop would be $2W$. By contrast, for the total-payoff game depicted in Figure 1.6, each internal loop requires 2 iterations to converge, but the external loop takes W iterations to stabilise. A combination of both examples would experience a pseudo-polynomial number of iterations to converge in both the internal and external loops, matching the W^2 term of the above complexity: this gives rise to the parametric example of Figure 1.7.

Optimal strategies In Section 1.2, we have shown, for all shortest-path games, the existence of a fake-optimal NC-strategy permitting to reconstruct an optimal (finite-memory) switching strategy for Min (if every vertex has value different from $-\infty$, or a

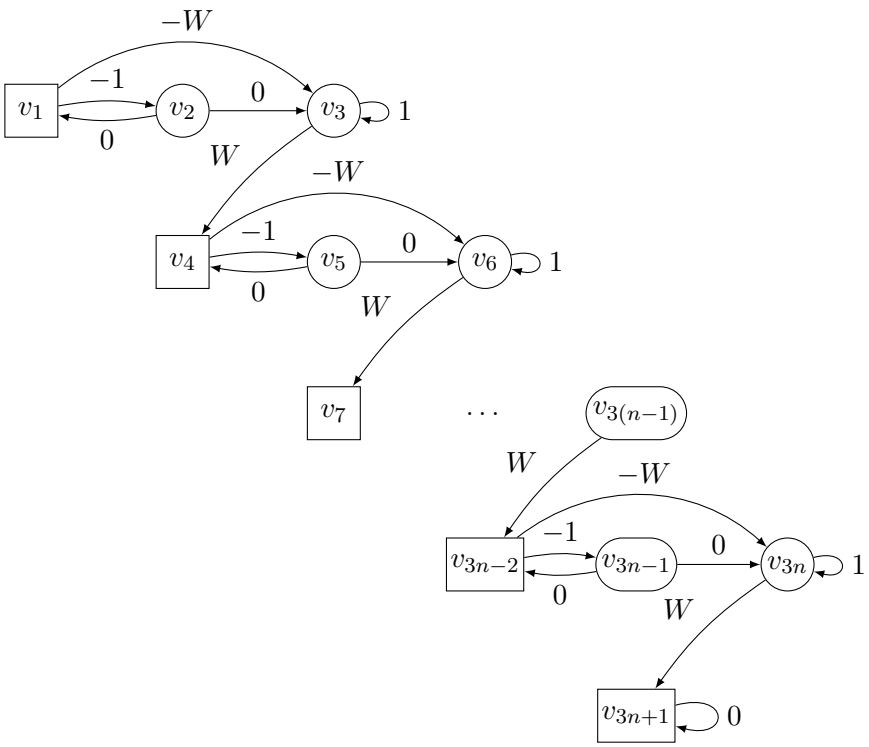

Figure 1.7 – Parametric total-payoff game

strategy ensuring every possible negative threshold for vertices with value $-\infty$). Given a total-payoff game $\mathcal{G}$, if we apply this construction to the game $\mathcal{G}_Y$ with $Y = \mathsf{Val}_{\mathcal{G}}$, we obtain an NC-strategy σ^1_{Min}. Consider the memoryless strategy σ^*_{Min}, obtained by keeping the decisions of σ^1_{Min} on non-stop vertices. Strategy σ^*_{Min} can indeed be shown to be optimal for Min in $\mathcal{G}$. Notice that σ^1_{Min}, and hence σ^*_{Min}, can be computed during the last iteration of the value iteration algorithm, as explained in the case of shortest-path games. A similar construction can be done to compute an optimal strategy for Max.

We finally summarise the results we have obtained in the context of total-payoff games:

Theorem 1.11. *Let $\mathcal{G}$ be a total-payoff game. Computing all values of $\mathcal{G}$, as well as optimal memoryless strategies for both players, can be done in (pseudo-polynomial) time $\mathcal{O}(|V|^4|E|W^2)$.*

1.4 Implementation and heuristics

In this section, we report on a prototype implementation of our algorithms.[8] For convenience reasons, we have implemented them as an add-on to PRISM-games [Che+13], although we could have chosen to extend another model-checker as we do not rely on the probabilistic features of PRISM models (i.e. we use the PRISM syntax of *stochastic multi-player games*, allowing arbitrary rewards, and forbidding probability distributions different of Dirac ones). We then use rPATL specifications of the form $\langle\langle C\rangle\rangle \mathsf{R}^{\min\,/\,\max=?}[\mathsf{F}^{\infty}\varphi]$ and

8. Source and binary files, as well as some examples, can be downloaded from http://www.ulb.ac.be/di/verif/monmege/tool/TP-MCR/.

Table 1.1 – Results of value iteration on a parametric example

		without heuristics			with heuristics		
W	n	t	k_e	k_i	t	k_e	k_i
50	100	0.52s	151	12 603	0.01s	402	1 404
50	500	9.83s	551	53 003	0.42s	2,002	7 004
200	100	2.96s	301	80 103	0.02s	402	1 404
200	500	45.64s	701	240 503	0.47s	2 002	7 004
500	1 000	536s	1 501	1 251 003	2.37s	4 002	14 004

$\langle\langle C\rangle\rangle \mathsf{R}^{\min\,/\,\max=?}[\mathsf{F}^c \bot]$ to model respectively shortest-path games and total-payoff games, where C represents a coalition of players that want to minimise/maximise the payoff, and φ is another rPATL formula describing the target set of vertices (for total-payoff games, such a formula is not necessary). We have tested our implementation on toy examples. On the parametric one of Figure 1.7, results obtained by applying our algorithm for total-payoff games are summarised in Table 1.1, where for each pair (W, n), we give the time t in seconds, the number k_e of iterations in the external loop, and the total number k_i of iterations in the internal loop.

Notice that due to the very little memory consumption of the algorithm, there is no risk of running out of memory. However, the execution time can become very large. For instance, in case $W = 500$ and $n = 1000$, the execution time becomes $536s$ whereas the total number of iterations in the internal loop is greater than a million.

We close this section by sketching two techniques that can be used to speed up the computation of the fixed point in Algorithms 1 and 2. Both accelerations rely on a topological order of the strongly connected components (SCC for short) of the game graph, given as a function $\mathsf{cp}\colon V \to \mathbb{N}$, mapping each vertex to its *component*, satisfying that

— $\mathsf{cp}(V) = \{0, \dots, p\}$ for some $p \geqslant 0$,

— $\mathsf{cp}^{-1}(q)$ is a maximal SCC for all q,

— and $\mathsf{cp}(v) \geqslant \mathsf{cp}(v')$ for all $(v, a, v') \in E$.[9]

In case of a shortest-path game with ☺ the unique target, $\mathsf{cp}^{-1}(0) = \{☺\}$. Intuitively, cp induces a directed acyclic graph whose vertices are the sets $\mathsf{cp}^{-1}(q)$ for all $q \in \mathsf{cp}(V)$, and with an edge (S_1, S_2) if and only if there are $v_1 \in S_1, v_2 \in S_2$ such that $(v_1, v_2) \in E$.

The *first acceleration heuristic*, very classical, is a divide-and-conquer technique that consists in applying Algorithm 1 (or the inner loop of Algorithm 2) iteratively on each $\mathsf{cp}^{-1}(q)$ for $q = 0, 1, 2, \dots, p$, using at each step the information computed during steps $j < q$ (since the value of a vertex v depends only on the values of the vertices v' such that $\mathsf{cp}(v') \leqslant \mathsf{cp}(v)$).

The *second acceleration heuristic*, more ad-hoc, consists in studying more precisely each component $\mathsf{cp}^{-1}(q)$. Having already computed the optimal values $\mathsf{Val}(v)$ of vertices $v \in \mathsf{cp}^{-1}(\{0, \dots, q-1\})$, we ask an oracle to precompute a finite set $S_v \subseteq \mathbb{Z}_\infty$ of possible optimal values for each vertex $v \in \mathsf{cp}^{-1}(q)$. For shortest-path games and the inner iteration of the algorithm for total-payoff games, one way to construct such a set S_v

9. Such a mapping is computable in linear time, e.g., by Tarjan's algorithm [Tar72].

is to consider that possible optimal values are the one of non-looping paths inside the component exiting it: this stems from the existence, in shortest-path games, of optimal strategies for both players whose outcome is a non-looping path (see Section 1.2).

Finally, we note that we can identify classes of weighted graphs for which there exists an oracle that runs in polynomial time and returns, for all vertices v, a set S_v of polynomial size. On such classes, Algorithms 1 and 2, enhanced with our two acceleration techniques, *run in polynomial time*. For instance, for all fixed positive integers L, the class of weighted graphs where every component $\mathsf{cp}^{-1}(q)$ uses at most L distinct weights (that can be arbitrarily large in absolute value) satisfies this criterion. Table 1.1 contains the results obtained with the heuristics on the parametric example presented before. Observe that the acceleration technique permits here to drastically decrease the execution time, the number of iterations in both loops not even depending anymore on W. Even though the number of iterations in the external loop increases with heuristics, due to the decomposition, less computation is required in each internal loop since we only apply the computation for the active component.

1.5 Divergent shortest-path games

So far, we have seen that solving shortest-path games can be performed in pseudo-polynomial time, though the best computational lower-bound is PTIME for the value or existence problems (since it is already the case for unweighted reachability games [Imm81]). It is thus tempting to try to close this gap. In this last section, we introduce a subclass of shortest-path games that we are able to solve in polynomial time. The same core techniques are then lifted in Chapter 4 in the timed setting. It is based on the restriction on the allowed cycles (i.e. cyclic finite plays) in the game:

Definition 1.12. A shortest-path game $\mathcal{G}$ is divergent if every cycle ρ of $\mathcal{G}$ satisfies $\mathsf{wt}(\rho) \neq 0$.

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 1.13. *The value problem over divergent shortest-path games is* PTIME*-complete. Moreover, deciding if a shortest-path game is divergent is an* NL*-complete problem when weights are encoded in unary, and is in* PTIME *when they are encoded in binary.*

We now sketch the proof of this result, based on a careful analysis of the SCCs.

SCC analysis A finite play ρ in $\mathcal{G}$ is said to be positive (respectively, negative) if $\mathsf{wt}(\rho) > 0$ (respectively, $\mathsf{wt}(\rho) < 0$). It follows that a cycle in a divergent shortest-path 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, whose proof

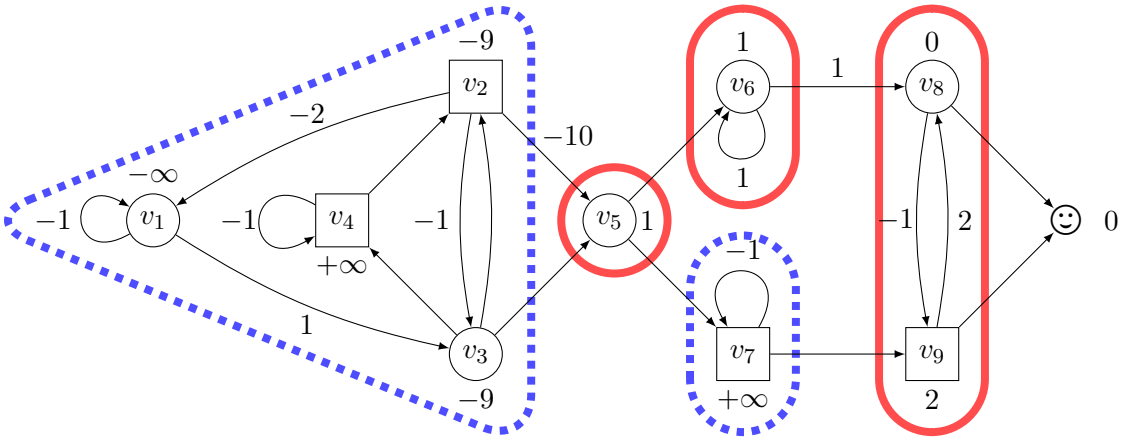

Figure 1.8 – SCC decomposition of a divergent shortest-path game: $\{v_1, v_2, v_3, v_4\}$ and $\{v_7\}$ are negative SCCs, $\{v_6\}$ and $\{v_8, v_9\}$ are positive SCCs, and $\{v_5\}$ is a trivial positive SCC.

relies on the fact that weights of cycles are integers and thus two cycles of weight $-p < 0$ and $p' > 0$ can be concatenated enough times to obtain a cycle of weight 0.

Proposition 1.14. *A shortest-path 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.*

Computing the values We already know how to compute vertices of value $+\infty$ in polynomial time, thanks to attractors. We now assume that all values are in $\mathbb{Z} \cup \{-\infty\}$. The first heuristics studied before explains that the value of vertices can safely be computed SCC by SCC in a bottom-up fashion (starting with target vertices).

Example 1.15. Consider the shortest-path game of Figure 1.8. Near each vertex is placed its value. By a computation of the attractor of the target 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 ☺ 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 the value iteration operator $\mathcal{F}$ 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. [10] Notice that this game is divergent, since, in each SCC, all simple cycles have the same sign.

For a divergent game $\mathcal{G}$, Proposition 1.14 allows us to know in polynomial time if a given SCC is positive or negative, i.e. if all cycles it contains are positive 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 have already seen in page 23 that if a game does not contain any negative cycle (like in a positive SCC), the value iteration algorithm stabilises after a number of steps at most the number of vertices of the SCC.

10. 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.

Example 1.16. For the SCC $\{v_8, v_9\}$ of the game in Figure 1.8, starting from a vector mapping v_8 and v_9 to $+\infty$, and ☺ to 0, after one iteration, the value of v_8 changes for value 0, and after the second iteration, the value of 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 (contrary to the general case of (non divergent) games where the problem of deciding if a vertex has value $-\infty$ is as hard as solving mean-payoff games, by Theorem 1.4). These are all vertices where Max cannot unilaterally guarantee to reach a target vertex:

Proposition 1.17. *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.*

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:

— switching vertices of Min and Max;

— 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 shortest-path games are determined (Theorem 1.2), the value of $\mathcal{G}$ is the opposite of the value of $\widetilde{\mathcal{G}}$, that can be computed in polynomial time by the value iteration procedure, as observed before. Alternatively said, we may interpret this result as follows:

Proposition 1.18. *The value iteration algorithm, initialised with all values being $-\infty$, applied on a negative SCC with n vertices, and no vertices of value $+\infty$ or $-\infty$, stabilises after at most n steps.*

Example 1.19. Consider the SCC $\{v_1, v_2, v_3, v_4\}$ of the game in Figure 1.8, 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 the vector mapping v_2 and v_3 to $-\infty$, the value iteration algorithm computes this sequence of vectors: $(-9, -\infty)$ (Max tries to maximise the payoff, so they prefer to jump to the target to obtain $-10+1$ than going to v_3 where they get $-1-\infty$, while Min chooses v_2 to still guarantee $0-\infty$), and then $(-9, -9)$ (now, Min has a choice between the target giving $0+1$ or v_3 giving $0-9$) which stabilises.

Class decision We explain why deciding the divergence of a shortest-path game is an NL-complete problem when weights are encoded in unary. First, to prove the membership in NL, notice that a shortest-path 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. [11] 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,

11. If the game is not divergent, there exists an SCC containing a negative simple cycle and a positive one by Proposition 1.14. This implies the existence of a negative cycle and a positive cycle in the same SCC, both of length at most $|V|$. Reciprocally, this property implies the non-divergence, by the same proof as for Proposition 1.14.

keeping in memory their cumulated 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 $\mathcal{G}$, if weights are encoded in unary), and stop the on-the-fly exploration when the length of the cycles exceeds $|V|$. Therefore testing divergence is in $\mathsf{coNL} = \mathsf{NL}$ [Imm88; Sze88].

The NL-hardness (indeed coNL-hardness, which is equivalent [Imm88; Sze88]) 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 shortest-path game by distributing all states to Min, and equipping all edges with weight 1. We also add a loop with weight -1 on the target state and an edge from the target vertex 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.

When weights are encoded in binary, the previous decision procedure gives NP membership. However, we can achieve a PTIME upperbound with the following procedure. For every vertex v, let $C_v = \{\mathsf{wt}(\rho) \mid \rho \text{ cycle containing } v\}$. Using Floyd-Warshall's algorithm, it is possible to compute in polynomial time $\inf C_v$ (in particular, it detects if $C_v \neq \emptyset$), as well as $\sup C_v$ in a dual fashion. Then, Proposition 1.14 allows us to guarantee that a shortest-path game $\mathcal{G}$ is divergent if and only if $0 \notin [\inf C_v, \sup C_v]$, for all vertices v such that $C_v \neq \emptyset$.

Almost-divergence extension With divergent shortest-path games, we described a class where the value problem is polynomial instead of pseudo-polynomial. This gain in complexity came at a cost, the absence of cycles of weight 0. However, one could argue that such cycles are useful from a modelling point of view. We argue that some of those cycles can be allowed, without losing too much:

Definition 1.20. A shortest-path game $\mathcal{G}$ is *almost-divergent* if every cycle ρ of weight 0 satisfies the following property: for every decomposition of ρ into the concatenation of two smaller cycles ρ' and ρ'', ρ' and ρ'' have weight 0 too.

Intuitively, a game is almost-divergent if its cycles of weight different from 0 cannot be combined to create a cycle of weight 0. Almost-divergence is a weaker property than divergence, and thus every divergent weighted game is almost-divergent. Theorem 1.13 can be extended as follows:

Theorem 1.21. *The value problem over almost-divergent shortest-path games is* PTIME*-complete. Moreover, deciding if a given shortest-path game is almost-divergent is an* NL*-complete problem when weights are encoded in unary, and is in* PTIME *when they are encoded in binary.*

The proof is more involved than in the divergent case, since 0-cycles must be taken care of in a special way. We will only describe the machinery to solve this case in the timed setting of Chapter 4, but we refer readers interested to knowing further details on the untimed setting to the article [BMR21].

Conclusion

In this chapter, we have provided pseudo-polynomial time algorithms to solve total-payoff games with arbitrary (positive and negative) weights, as well as shortest-path games.

These algorithms are variations of the classical value iteration technique. The result on total-payoff games relies on a reduction to shortest-path games. We have characterised the optimal strategies that one can extract in total-payoff and shortest-path games.

In the specific case of shortest-path games, we have introduced divergence and almost-divergence, two natural restrictions that allow us to recover a polynomial-time (even linear time) algorithm, more amenable to results in the presence of weights of large amplitude in the graph.

As future works, we would like to push further the shortest-path games in a context of non-zero sum games where each player wants to optimise their cumulated payoff until reaching their own target. As a possible direction, the search for Nash equilibria in this context will most likely benefit from our better understanding of optimal strategies for both players in the underlying zero-sum games. We have initiated this study in [Bri+16a], with an application to the efficient energy distribution in a smart grid. This bridge from zero-sum to non-zero-sum games has also been investigated for concurrent priced games by [Kli+12], and by [BDS13] to find simple Nash equilibria for large classes of multiplayer cost games.

Another long-term perspective is to continue searching for polynomial-time algorithms for the whole class of shortest-path and total-payoff games. We have seen that it strongly relates to mean-payoff games, for which the search for such polynomial-time algorithms is a long-standing open question. We claim that shortest-path games, with the presence of the special target, gives a decomposition of the game that is not present in mean-payoff games, and could lead to new algorithms. As a next direction, the use of stochasticity (as we will study in Chapter 5) could give us new insights towards polynomial-time algorithm, based on a parallel with interior-point methods to solve linear programs.

2 Weighted Timed Games: Models and Problems

This chapter follows the presentation of the submitted journal article [BMR21] (written with Damien Busatto-Gaston and Pierre-Alain Reynier), and recalls results obtained with Thomas Brihaye, Gilles Geeraerts, Axel Haddad, Shankara Narayanan Krishna, Engel Lefaucheux, Lakshmi Manasa, and Ashutosh Trivedi, published in the conferences CONCUR [Bri+14] and FSTTCS [Bri+15a], as well as a submitted journal article [Bri+21].

Table of contents

We have studied in the previous chapters shortest-path games and total-payoff games played on finite graphs equipped with integer weights. From a modelling point of view, these discrete games do not allow us to consider thinner behaviours due to real-time evolution. For instance, consider a model of costs for electrical power consumption. Fixed costs, like the ones in shortest-path games, are useful (e.g. contract costs) but not sufficient. Indeed, most of the cost depends (linearly) on the time a given appliance is used. Discretising the time is always a possibility, but this makes the model size blow out, and we do not know if we lose too much precision, thus missing possibly interesting behaviours. We will instead model such situations with *timed automata* [AD94] equipped with weights, both on transitions (to model fixed costs) and on vertices (to model costs depending linearly on the time spent in a vertex).

In this chapter, we introduce the models of weighted timed games we will use, based on the presentation of [BMR21] (that itself benefits from the historic introduction of this model [ABM04; Bou+04b]), focusing only on the shortest-path objective we have studied in the previous chapter.

2.1 Modelling real-time constraints

We first introduce notions that let us express timing constraints, useful to then define weighted timed games, and introduce classical tools for their study.

Let $\mathcal{X} = \{\mathsf{x}_1, \dots, \mathsf{x}_n\}$ be a finite, non-empty set of variables called clocks. A *valuation* $\nu \colon \mathcal{X} \to \mathbb{R}_{\geqslant 0}$ is a mapping from clocks to non-negative real numbers, such that $\nu(\mathsf{x}_1), \dots, \nu(\mathsf{x}_n)$ are called the coordinates of ν. Equivalently, ν can be seen as a point in

space $\mathbb{R}^{\mathcal{X}}_{\geqslant 0}$. We denote $\mathbf{0}$ the valuation such that for all $\mathsf{x} \in \mathcal{X}$, $\nu(\mathsf{x}) = 0$. Given a real number $t \in \mathbb{R}$, we define $\nu + t$ as the valuation such that $\forall \mathsf{x} \in \mathcal{X}, (\nu + t)(\mathsf{x}) = \nu(\mathsf{x}) + t$ if it exists.[1] If t is non-negative, we say that we performed a time elapse of delay t. The *time-successors* of ν are the valuations $\nu + t$ with $t \geqslant 0$. Similarly, we refer to all $\nu + t$ in $\mathbb{R}^{\mathcal{X}}_{\geqslant 0}$ with $t \leqslant 0$ as *time-predecessors* of ν. The set of points that are either time-predecessors or time-successors of a valuation ν form the unique "diagonal" line in $\mathbb{R}^{\mathcal{X}}_{\geqslant 0}$ that contains ν. If $\mathcal{Y}$ is a subset of $\mathcal{X}$, we define $\nu[\mathcal{Y} := 0]$ as the valuation such that $\forall \mathsf{x} \in \mathcal{Y}, (\nu[\mathcal{Y} := 0])(\mathsf{x}) = 0$ and $\forall \mathsf{x} \in \mathcal{X} \backslash \mathcal{Y}, (\nu[\mathcal{Y} := 0])(\mathsf{x}) = \nu(\mathsf{x})$. This operation is called a *reset* of clocks $\mathcal{Y}$.

We extend those notions to sets of valuations in a natural way. The set of time-successors of $Z \subseteq \mathbb{R}^{\mathcal{X}}_{\geqslant 0}$, denoted $\mathsf{PostTime}(Z)$, contains the valuations that are time-successors of valuations in Z. The reset of $Z \subseteq \mathbb{R}^{\mathcal{X}}_{\geqslant 0}$ by $\mathcal{Y}$, denoted $Z[\mathcal{Y} := 0]$, contains the valuations $\nu[\mathcal{Y} := 0]$ such that $\nu \in Z$.

The term *atomic constraint* will refer to an affine inequality in one of the following forms:

— A strict (resp. non-strict) *non-diagonal* atomic constraint over clock $\mathsf{x} \in \mathcal{X}$ and constant $c \in \mathbb{Q}$ is an inequality of the form $\mathsf{x} \bowtie c$ with $\bowtie \in \{>, <\}$ (resp. $\bowtie \in \{\geqslant, \leqslant\}$).

— A strict (resp. non-strict) *diagonal* atomic constraint over clocks x and $\mathsf{y} \in \mathcal{X}$ and constant $c \in \mathbb{Q}$ is an inequality of the form $\mathsf{x} - \mathsf{y} \bowtie c$ with $\bowtie \in \{>, <\}$ (resp. $\bowtie \in \{\geqslant, \leqslant\}$).

Let $\top$ and $\bot$ denote two special atomic constraints, defined as $\mathsf{x} \geqslant 0$ and $\mathsf{x} < 0$ for an arbitrary $\mathsf{x} \in \mathcal{X}$. A *guard* g over $\mathcal{X}$ is a finite conjunction of atomic constraints over clocks in $\mathcal{X}$. In particular, guards let us define $\mathsf{x} = c$ as shorthand for $\mathsf{x} \leqslant c \wedge \mathsf{x} \geqslant c$, and $c_1 < \mathsf{x} < c_2$ as shorthand for $\mathsf{x} > c_1 \wedge \mathsf{x} < c_2$. A guard is said strict (resp. non-strict, diagonal, non-diagonal) if all of its atomic constraints are strict (resp. non-strict, diagonal, non-diagonal). We let $\mathsf{Guards}(\mathcal{X})$ denote the set of all guards over $\mathcal{X}$, and $\mathsf{Guards}^{\mathrm{nd}}(\mathcal{X})$ the subset of non-diagonal guards. For all constants $c \in \mathbb{Q}$ and $\bowtie \in \{\geqslant, \leqslant, >, <\}$, we say that valuation $\nu \in \mathbb{R}^{\mathcal{X}}_{\geqslant 0}$ satisfies the atomic constraint $\mathsf{x} \bowtie c$ (resp. $\mathsf{x} - \mathsf{y} \bowtie c$), and write $\nu \models \mathsf{x} \bowtie c$ (resp. $\nu \models \mathsf{x} - \mathsf{y} \bowtie c$), if $\nu(\mathsf{x}) \bowtie c$ (resp. $\nu(\mathsf{x}) - \nu(\mathsf{y}) \bowtie c$). We say that valuation $\nu \in \mathbb{R}^{\mathcal{X}}_{\geqslant 0}$ satisfies guard g, and write $\nu \models g$, if ν satisfies all atomic constraints in g. For $g \in \mathsf{Guards}(\mathcal{X})$, let $[\![g]\!]$ denote the set of all $\nu \in \mathbb{R}^{\mathcal{X}}_{\geqslant 0}$ such that $\nu \models g$. Such sets are called *zones* and form convex polyhedra of $\mathbb{R}^{\mathcal{X}}_{\geqslant 0}$. A guard g is said satisfiable when the zone $[\![g]\!]$ is non-empty, and a zone is called rectangular when the associated guard is non-diagonal. The universal zone refers to $[\![\top]\!] = \mathbb{R}^{\mathcal{X}}_{\geqslant 0}$ and the empty zone refers to $[\![\bot]\!] = \emptyset$. Guard $\overline{g}$ is the closed version of a satisfiable guard g where every strict constraint of comparison operator $<$ or $>$ is replaced by its non-strict version $\leqslant$ or $\geqslant$. The zone $[\![\overline{g}]\!]$ is the topological closure of $Z = [\![g]\!]$, and is also denoted $\overline{Z}$.

2.2 Weighted timed games

We now turn our attention to weighted timed two-player games with a shortest-path objective towards a set of target locations. The semantics will be given in terms of an *infinite* shortest-path game.

1. If t is negative, $\nu + t$ may not belong to $\mathbb{R}^{\mathcal{X}}_{\geqslant 0}$

Definition 2.1. A *weighted timed game* (WTG) is a tuple $\mathcal{G} = \langle L_{\mathsf{Min}}, L_{\mathsf{Max}}, L_{\mathsf{t}}, L_{\mathrm{u}}, \mathcal{X}, \Delta, \mathsf{wt}, \mathsf{fwt}\rangle$ with

— $L = L_{\mathsf{Min}} \uplus L_{\mathsf{Max}} \uplus L_{\mathsf{t}}$ a finite set of locations split between players Min and Max (in drawings, locations belonging to Min are depicted by circles and the ones belonging to Max by squares) and a set of target locations;

— $L_{\mathrm{u}} \subseteq L_{\mathsf{Min}} \uplus L_{\mathsf{Max}}$ a set of *urgent* locations where time cannot be delayed;

— $\mathcal{X}$ a finite set of clocks;

— $\Delta \subseteq L \times \mathsf{Guards}^{\mathrm{nd}}(\mathcal{X}) \times 2^{\mathcal{X}} \times L$ a finite set of transitions $\ell \xrightarrow{g,\mathcal{Y}} \ell'$ from location ℓ to location ℓ', labelled by a non-diagonal guard g and a set $\mathcal{Y}$ of clocks to reset (in drawings, the reset of a clock $\mathsf{x} \in \mathcal{Y}$ will be denoted by $\mathsf{x} := 0$);

— $\mathsf{wt}\colon \Delta \uplus L \to \mathbb{Z}$ a weight function associating an integer weight with each transition and location;

— and $\mathsf{fwt}\colon L_{\mathsf{t}} \times \mathbb{R}^{\mathcal{X}}_{\geqslant 0} \to \mathbb{R}_{\infty}$ is a function mapping each target configuration to a final weight.

The addition of final weights is not standard, but we will use it in the process of solving those games: in any case, it is possible to simply map a given target location to the weight 0, allowing us to recover the standard definitions of the literature.

The presence of urgent locations is also unusual: in a timed automaton, urgency can be modelled with an additional clock u that is reset just before entering the urgent location and with guards $\mathsf{u} = 0$ on outgoing transitions. However, when limiting the number of clocks as we will do in Chapter 3, we regain modelling capabilities by allowing for such urgent locations. The weight of an urgent location is never used, and will thus not be given in drawings: instead, urgent locations will be displayed with a u inside.

The semantics of a weighted timed game $\mathcal{G}$ is defined in terms of an (infinite) shortest-path game $[\![\mathcal{G}]\!]$ whose vertices are configurations $(\ell, \nu) \in L \times \mathbb{R}^{\mathcal{X}}_{\geqslant 0}$. Configurations are split into players according to the location ℓ, and a configuration (ℓ, ν) is a target if $\ell \in L_{\mathsf{t}}$. The labels of $[\![\mathcal{G}]\!]$ are given by $\mathbb{R}_{\geqslant 0} \times \Delta$ and will encode the delay that a player wants to spend in the current location, before firing a certain transition. For every delay $t \in \mathbb{R}_{\geqslant 0}$, transition $\delta = \ell \xrightarrow{g,\mathcal{Y}} \ell' \in \Delta$ and valuation ν, $[\![\mathcal{G}]\!]$ contains an edge $(\ell, \nu) \xrightarrow{t,\delta} (\ell', \nu')$ if

— $\nu + t \models g$;

— $\nu' = (\nu + t)[\mathcal{Y} := 0]$;

— and $t = 0$ if $\ell \in L_{\mathrm{u}}$.

The weight of such an edge takes into account both discrete and continuous costs: it is given by $t \times \mathsf{wt}(\ell) + \mathsf{wt}(e)$.

As usual in related work [ABM04; Bou+04b; BJM15], we will make some hypotheses on the games.

— First, we will assume that weighted timed games have only non-diagonal guards where all constants are integers: this is without loss of generality since we can apply the methods of [Bér+98] to remove diagonal guards.

— Moreover, we will assume that all clocks are *bounded* by the greatest constant M that appears in guards, and we restrict $[\![\mathcal{G}]\!]$ to configurations in $L \times [0, M)^{\mathcal{X}}$. This is known to be without loss of generality for (weighted) timed automata [Beh+01,

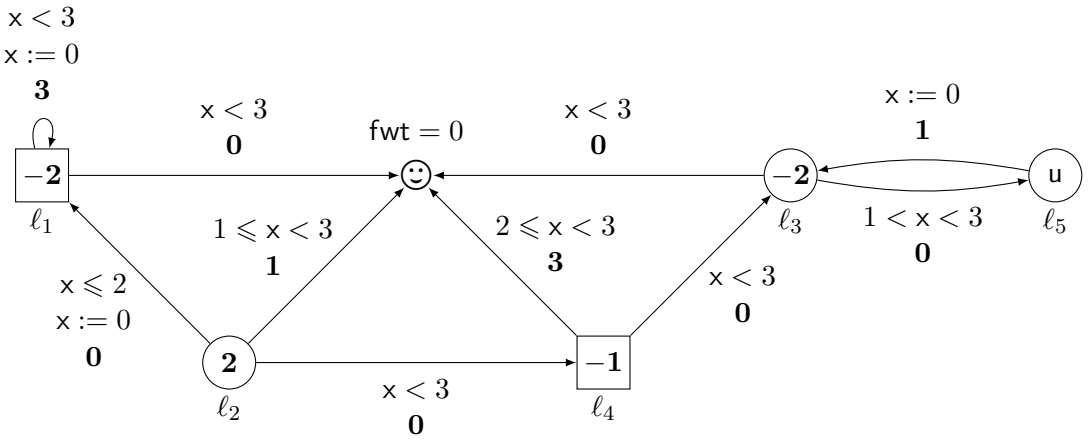

Figure 2.1 – A weighted timed game with a single clock x. Weights are indicated in bold font on locations and transitions. The target location is ☺, whose final weight function is zero. Location ℓ_5 is urgent.

Theorem 2]: it suffices to replace transitions with unbounded delays with self-loop transitions periodically resetting the clocks. We do not know if it is the case for the weighted timed games defined above. Indeed, the technique of [Beh+01] cannot be directly applied. This would give too much power to player Max that would then be allowed to loop in a location where an unbounded delay could originally be taken before going to the target. In [Bou+04b], the situation is simpler since the game is *concurrent*, and thus Min always has a chance to move outside of such a situation. Trying to detect and avoid such situations in our turn-based case seems difficult in the presence of negative weights, since the opportunities of Max crucially depend on the configurations of value $-\infty$ that Min could control afterwards: we will see in Proposition 4.15 that detecting such configurations is undecidable, which is an additional evidence to motivate the decision to focus only on bounded weighted timed games.

— Without loss of generality, we also suppose the absence of deadlocks in $[\![\mathcal{G}]\!]$ except on target locations, i.e. for each location $\ell \in L \backslash L_{\mathsf{t}}$ and valuation $\nu \in [0, M)^{\mathcal{X}}$, there exist $t \in \mathbb{R}_{\geqslant 0}$ and $\delta = (\ell, g, \mathcal{Y}, \ell') \in \Delta$ such that $(\ell, \nu) \xrightarrow{t,\delta} (\ell', \nu')$, and no transitions start from L_{t}.

— We finally assume that the final weight functions satisfy a sufficient property ensuring that they can be encoded in finite space: they must be piecewise affine with a finite number of pieces and are continuous on each region (we will recall in Section 2.4 the formal definition of regions). In particular, infinite final weights are constant over regions, i.e. if some configuration (ℓ_{t}, ν) has final weight $+\infty$ or $-\infty$, then for every valuation ν' in the same region as ν, $\mathsf{fwt}(\ell_{\mathsf{t}}, \nu) = \mathsf{fwt}(\ell_{\mathsf{t}}, \nu')$. The standard final weight function $(\ell_{\mathsf{t}}, \nu) \mapsto 0$ satisfies this property. Moreover, the computations we will perform in the following maintain this property as an invariant.

An example of weighted timed game satisfying those assumptions is depicted on Figure 2.1.

Notions of plays, strategies and values are obtained for $\mathcal{G}$, by considering the similar

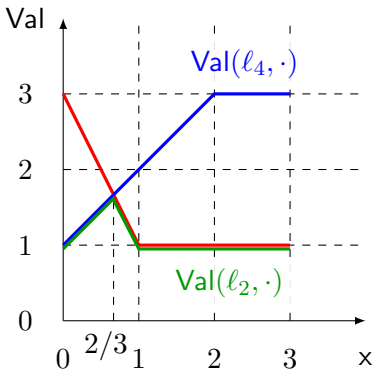

Figure 2.2 – Value functions

notions on $[\![\mathcal{G}]\!]$, defined in Chapter 1. Because of the infinite nature of the timed games, optimal strategies may no longer exist: for example, a player may want to let time elapse as much as possible, but with a delay $t < 1$ because of a strict guard, preventing them to obtain the optimal value. We naturally extend the definition to *almost-optimal strategies*, taking into account small possible errors: we say that a strategy $\sigma^\star_{\mathsf{Min}}$ of Min is ε-optimal if, for all initial configurations (ℓ_0, ν_0)

$$\mathsf{Val}((\ell_0,\nu_0),\sigma^\star_{\mathsf{Min}}) \leqslant \overline{\mathsf{Val}}(\ell_0,\nu_0) + \varepsilon$$

Symmetrically, a strategy $\sigma^\star_{\mathsf{Max}}$ of Max is ε-optimal if, for all initial configurations (ℓ_0, ν_0)

$$\mathsf{Val}((\ell_0,\nu_0),\sigma^\star_{\mathsf{Max}}) \geqslant \underline{\mathsf{Val}}(\ell_0,\nu_0) - \varepsilon$$

Example 2.2. In the weighted timed game of Figure 2.1, observe that the location ℓ_1 of Max has value $+\infty$ no matter the initial valuation in $[0, 3)$, since Max can avoid the target. Moreover, locations ℓ_3 and ℓ_5 of Min have value $-\infty$ no matter the initial valuation in $[0, 3)$: indeed, Min can loop in-between the two locations, cumulating a weight $\leqslant -1$ each time, before switching to the target when the weight is low enough. The value in ℓ_4 is only determined by the transition to ☺ (since Max tries to avoid ℓ_3), and depicted in blue in Figure 2.2. In location ℓ_2, the value that Min can get while jumping through the transition to ☺ is depicted in red. While jumping to ℓ_2 after a delay 0 (since $\mathsf{wt}(\ell_2) > 0$, Min wants to minimise the time spent in this location), Min can also obtain the value of ℓ_4. Therefore, the value of location ℓ_2 is obtained as the minimum of these two curves, depicted in green. Observe the intersection point in $\mathsf{x} = 2/3$ requiring to refine the usual regions (to be formally defined in Section 2.4).

As a corollary of Theorem 1.2 (that was stated even in infinite shortest-path games), we obtain that:

Theorem 2.3. *All weighted timed games are determined, i.e.* $\underline{\mathsf{Val}}(\ell,\nu) = \overline{\mathsf{Val}}(\ell,\nu)$ *for each location* ℓ *and valuation* ν.

In the rest of this manuscript, we therefore use the notation Val to refer to both values. We also introduce the notation $\mathsf{Val}(\ell)$ regrouping all values of configurations in location ℓ, i.e. a mapping $\mathbb{R}^{\mathcal{X}}_{\geqslant 0} \to \mathbb{R}_\infty$ such that $\mathsf{Val}(\ell)(\nu) = \mathsf{Val}(\ell,\nu)$ for all $\nu \in \mathbb{R}^{\mathcal{X}}_{\geqslant 0}$.

We denote by $w^L_{\max}$ (resp. $w^\Delta_{\max}$) the maximal weight in absolute values of locations (resp. of transitions) in $\mathcal{G}$:

$$w^L_{\max} = \max_{\ell \in L} |\mathsf{wt}(\ell)| \qquad \text{and} \qquad w^\Delta_{\max} = \max_{\delta \in \Delta} |\mathsf{wt}(\delta)|$$

Moreover, we denote by $w_{\max}$ a bound on the weight of edges in $[\![\mathcal{G}]\!]$, that exists since clocks are bounded by M:

$$w_{\max} = M w^L_{\max} + w^\Delta_{\max}$$

The integer $w_{\max}$ is at most exponential in the size of $\mathcal{G}$, and can thus be stored in polynomial space.

2.3 Problems and first results

We consider several problems related to weighted timed games throughout this manuscript, as already stated in the previous chapter: the existence problem, the value problem and the strategy synthesis problem. Because optimal strategies may no longer exist, the synthesis problem now takes as an input a weighted timed game and an error paramater ε, and asks for the computation of an ε-optimal strategy for Min.

It is well-known that the existence problem is undecidable, even in the presence of only non-negative weights in locations and transitions. We sharpened this result in [Bri+14], by reducing the necessary number of clocks to obtain this undecidability result:

Theorem 2.4. *The existence problem is undecidable in weighted timed games:*

- *with three or more clocks and only non-negative weights [BBR05; BBM06];*
- *with two or more clocks and arbitrary weights [Bri+14].*

Indeed, in [Bri+14], we also show that this undecidability result is very robust since it continues to hold (with the same number of clocks) if we want to find a strategy for Min whose value v satisfies $v \bowtie r$ with $\bowtie \in \{<, =, >, \geqslant\}$. We also studied the bounded-time restriction, hoping to recover decidability as it is the case in many related problems [OW10; Bri+13]. However, with a sufficient number of clocks we still get undecidability:

Theorem 2.5 ([Bri+14])**.** *Let $\mathcal{G}$ be a weighted timed game, and constants $K, T \in \mathbb{N}$. The decision problem corresponding to the existence of a strategy of* Min *reaching the target with a shortest-path payoff at most K, while keeping the total time elapsed within T units, is undecidable with five or more clocks (and even with weights 0 and 1).*

In [Bri+14], we were also able to modify the proof in order to show undecidability of a slightly different problem, asking for Min to reach target locations (that is thus not a deadlock) infinitely often each time with an accumulated weight in a given interval $[-\eta, \eta]$ (with $\eta > 0$).

It is less trivial to formally relate the existence problem with the value problem (asking whether $\mathsf{Val}(\ell, \nu) \leqslant \alpha$). Bouyer, Jaziri and Markey [BJM15] have formally shown that, already in the non-negative case:

Theorem 2.6 ([BJM15])**.** *The value problem is undecidable.*

We will study in the following chapters classes of weighted timed games where value and existence problems are decidable, and also study the synthesis problem in this case.

When the value problem is undecidable, we also consider the *value approximation problem* that consists, given a precision $\varepsilon \in \mathbb{Q}_{>0}$, in computing an ε-approximation of $\mathsf{Val}(\ell, \mathbf{0})$. More generally, we will try to compute an ε-approximation of the whole value function (and not only for the particular initial configuration $(\ell, \mathbf{0})$): this means that we want to compute (in a format that we will detail when needed) a function $\mathbf{V} \colon L \times [0, M)^{\mathcal{X}} \to \mathbb{R}_\infty$ such that $\|\mathsf{Val} - \mathbf{V}\|_\infty \leqslant \varepsilon$, where $\|\cdot\|_\infty$ denotes the classical ∞-norm of mappings, so that $\|f\|_\infty = \sup_x |f(x)|$.

2.4 Region abstraction

The analysis of timed systems often rely on the crucial notion of regions, as introduced in the seminal work on timed automata [AD94]. We will define an extension of regions that will be useful in the following, and see regions as a special case.

Given a finite set of rational numbers $\mathcal{S} \subseteq \mathbb{Q}$, $\mathcal{S}$ is said to be of granularity $1/N$ if $\mathcal{S} \subseteq \mathbb{Q}_N = \{c/N \mid c \in \mathbb{Z}\}$. Such N always exists, and one can find the smallest one by decomposing elements of $\mathcal{S}$ as irreducible fractions c/c' with $c \in \mathbb{Z}$, $c' \in \mathbb{N} \setminus \{0\}$ and use the least common multiple of all c' as N. A guard g is said to be of granularity $1/N$ if all constants in the atomic constraints of g form a set of granularity $1/N$. A zone is of granularity $1/N$ if it can be described by a guard of granularity $1/N$. Let $\mathsf{Guards}_N(\mathcal{X}, M)$ denote the set of guards over $\mathcal{X}$ bounded by M and of granularity $1/N$, and let $\mathsf{Guards}^{\mathrm{nd}}_N(\mathcal{X}, M)$ denote the non-diagonal ones. Given a finite set of guards $G \subseteq \mathsf{Guards}^{\mathrm{nd}}(\mathcal{X}, M)$, we can find N such that $G \subseteq \mathsf{Guards}^{\mathrm{nd}}_N(\mathcal{X}, M)$, by denoting $\mathcal{S} \subseteq \mathbb{Q}$ the set of constants used in atomic constraints of G and using N the smallest integer such that $\mathcal{S}$ is of granularity $1/N$. For all $a \in \mathbb{R}_{\geqslant 0}$, $\lfloor a \rfloor \in \mathbb{N}$ denotes the integral part of a, and $\mathsf{fract}(a) \in [0, 1)$ its fractional part, such that $a = \lfloor a \rfloor + \mathsf{fract}(a)$.

Definition 2.7. With respect to the set $\mathcal{X}$ of clocks, a granularity $N \in \mathbb{N} \setminus \{0\}$ and an upper bound $M \in \mathbb{N} \setminus \{0\}$, we define $1/N$*-regions* as subsets of valuations r characterised by a valuation $\iota \in [0, M)^{\mathcal{X}}$ called the integral part of r such that $\iota(\mathsf{x}) \in \mathbb{Q}_N$ for every $\mathsf{x} \in \mathcal{X}$, and an ordered partition $R_0 \uplus R_1 \uplus \cdots \uplus R_m$ splitting $\mathcal{X}$ into $m+1$ subsets. The ordered partition is denoted $0 = R_0 < R_1 < \cdots < R_m$, where R_0 can be empty but $R_i \neq \emptyset$ for $1 \leqslant i \leqslant m$. We denote by $\mathsf{Reg}_N(\mathcal{X}, M)$ the set of $1/N$-regions bounded by M.

A valuation $\nu \in [0, M)^{\mathcal{X}}$ belongs to r if

- for all $\mathsf{x} \in \mathcal{X}$, $\iota(\mathsf{x})N = \lfloor \nu(\mathsf{x})N \rfloor$;
- for all $\mathsf{x} \in R_0$, $\mathsf{fract}(\nu(\mathsf{x})N) = 0$;
- for all $0 \leqslant i \leqslant m$, for all $\mathsf{x}, \mathsf{y} \in R_i$, $\mathsf{fract}(\nu(\mathsf{x})N) = \mathsf{fract}(\nu(\mathsf{y})N)$.
- for all i, j such that $0 \leqslant i < j \leqslant m$, for all $\mathsf{x} \in R_i$ and all $\mathsf{y} \in R_j$, $\mathsf{fract}(\nu(\mathsf{x})N) < \mathsf{fract}(\nu(\mathsf{y})N)$.

With granularity $N = 1$, we recover the classical notion of regions from [AD94]. The set of valuations contained in a $1/N$-region r characterised by ι and $0 = \{\mathsf{x}^0_1, \ldots, \mathsf{x}^0_{m_0}\} < \{\mathsf{x}^1_1, \ldots, \mathsf{x}^1_{m_1}\} < \cdots < \{\mathsf{x}^m_1, \ldots, \mathsf{x}^m_{m_m}\}$ can be described by a formula, constructed as a conjunction of inequalities over $\mathcal{X}$: If we let $f(\mathsf{x})$ denote the term $(\mathsf{x} - \iota(\mathsf{x}))N$, and E^i

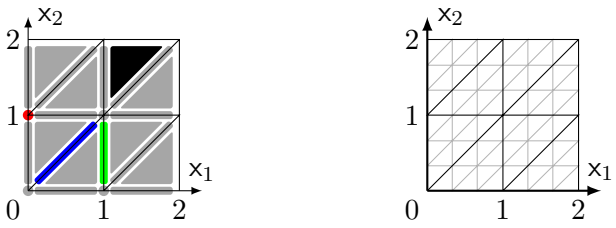

Figure 2.3 – All 1/1-regions in $\mathsf{Reg}_1(\{\mathsf{x}_1, \mathsf{x}_2\}, 2)$ on the left, their refinement of granularity 1/3 in $\mathsf{Reg}_3(\{\mathsf{x}_1, \mathsf{x}_2\}, 2)$ on the right.

denote the formula $f(\mathsf{x}_1^i) = \cdots = f(\mathsf{x}_{m_i}^i)$ for every $0 \leqslant i \leqslant m$, then $\nu \in r$ if and only if its coordinates satisfy

$$E^0 \wedge E^1 \wedge \cdots \wedge E^m \wedge 0 = f(\mathsf{x}_1^0) < f(\mathsf{x}_1^1) < \cdots < f(\mathsf{x}_1^m) < 1\,.$$

Therefore, every $1/N$-region is a zone of granularity $1/N$, associated to a guard in $\mathsf{Guards}_N(\mathcal{X}, M)$. Regions of $\mathsf{Reg}_N(\mathcal{X}, M)$ form a finite partition of $[0, M)^{\mathcal{X}}$, and the number $|\mathsf{Reg}_N(\mathcal{X}, M)|$ of $1/N$-regions is polynomial in MN and exponential in $|\mathcal{X}|$. If ν is a valuation in $[0, M)^{\mathcal{X}}$, $[\nu]$ denotes the unique region that contains ν. Valuations in the same $1/N$-region satisfy the same guards in $\mathsf{Guards}_N(\mathcal{X}, M)$. In fact, zones associated to guards in $\mathsf{Guards}_N(\mathcal{X}, M)$ can be described as a finite union of regions in $\mathsf{Reg}_N(\mathcal{X}, M)$. If r is a $1/N$-region in $\mathsf{Reg}_N(\mathcal{X}, M)$, then the time-successor valuations in $\mathsf{PostTime}(r) \cap [0, M)^{\mathcal{X}}$ form a finite union of regions in $\mathsf{Reg}_N(\mathcal{X}, M)$, and the reset $r[\mathcal{Y} := 0]$ of $\mathcal{Y} \subseteq \mathcal{X}$ is a region in $\mathsf{Reg}_N(\mathcal{X}, M)$. A $1/N$-region r' is said to be a time successor of the $1/N$-region r if there exists $\nu \in r$, $\nu' \in r'$, and $t > 0$ such that $\nu' = \nu + t$.

Example 2.8. Figure 2.3 represents the 24 regions of granularity $N = 1$ with upper bound $M = 2$ over two clocks. The green region is characterised by $\iota = (1, 0)$ and $0 = \{\mathsf{x}_1\} < \{\mathsf{x}_2\}$, corresponds to the formula $0 = \mathsf{x}_1 - 1 < \mathsf{x}_2 < 1$ and thus is equal to the zone $[\![\mathsf{x}_1 = 1 \wedge 0 < \mathsf{x}_2 < 1]\!]$. The red region is characterised by $\iota = (0, 1)$ and $0 = \{\mathsf{x}_1, \mathsf{x}_2\}$. The black region is characterised by $\iota = (1, 1)$ and $0 < \{\mathsf{x}_1\} < \{\mathsf{x}_2\}$. The blue region is characterised by $\iota = (0, 0)$ and $0 < \{\mathsf{x}_1, \mathsf{x}_2\}$.

The partition into regions can be maintained throughout the play in a weighted timed game. By forgetting about the precise valuation of clocks, we obtain a *region abstraction* (that is usually called the *region automaton* in the literature).

Definition 2.9. Given a weighted timed game $\mathcal{G} = \langle L_{\mathsf{Min}}, L_{\mathsf{Max}}, L_{\mathsf{t}}, L_{\mathsf{u}}, \mathcal{X}, \Delta, \mathsf{wt}, \mathsf{fwt}\rangle$ such that all clocks are bounded by M and all guards belong to $\mathsf{Guards}_N^{\mathrm{nd}}(\mathcal{X}, M)$ for some granularity $1/N$, we define the *region abstraction* of $\mathcal{G}$ as the finite labelled transition system $\langle L \times \mathsf{Reg}_N(\mathcal{X}, M), T\rangle$ labelled over $\mathsf{Reg}_N(\mathcal{X}, M) \times \Delta$, where T contains all transitions $(\ell, r) \xrightarrow{r'', \delta} (\ell', r')$ such that

— $\delta = \ell \xrightarrow{g, \mathcal{Y}} \ell'$ is a transition of $\mathcal{G}$;

— r'' is a time-successor of r such that $r'' = r$ in case $\ell \in L_{\mathrm{u}}$;

— $r'' \models g$;

— $r''[\mathcal{Y} := 0] = r'$.

The states of the region abstraction are called *region states*, and its paths are called *region paths*. As there are finitely many regions, the region abstraction of $\mathcal{G}$ is a finite transition system, where paths p represent a sequence of regions alternating between letting time elapse and taking edges, following some path π in $\mathcal{G}$. We say that π contains the region path p. From a play $\rho = (\ell_0, \nu_0) \xrightarrow{t_1,\delta_1} (\ell_1, \nu_1) \xrightarrow{t_2,\delta_2} \cdots$ in $\mathcal{G}$, we can construct a region path $\mathsf{p} = (\ell_0, [\nu_0]) \xrightarrow{[\nu_0+t_1],\delta_1} (\ell_1, [\nu_1]) \xrightarrow{[\nu_1+t_2],\delta_2} \cdots$, and say that ρ follows p.

From the region abstraction, we can construct a *region game* that can be seen as a product of the original weighted timed game with the region abstraction.

Definition 2.10. Given a weighted timed game $\mathcal{G} = \langle L_{\mathsf{Min}}, L_{\mathsf{Max}}, L_{\mathsf{t}}, \mathcal{X}, \Delta, \mathsf{wt}, \mathsf{fwt}\rangle$ such that all clocks are bounded by M and all guards belong to $\mathsf{Guards}^{\mathrm{nd}}_N(\mathcal{X}, M)$ for some granularity $1/N$, we define the *region game* of $\mathcal{G}$ as the weighted timed game $\mathcal{R}_N(\mathcal{G}) = \langle L_{\mathsf{Min}} \times \mathsf{Reg}_N(\mathcal{X}, M), L_{\mathsf{Max}} \times \mathsf{Reg}_N(\mathcal{X}, M), L_{\mathsf{t}} \times \mathsf{Reg}_N(\mathcal{X}, M), L_{\mathrm{u}} \times \mathsf{Reg}_N(\mathcal{X}, M), \mathcal{X}, \Delta', \mathsf{wt}', \mathsf{fwt}'\rangle$ whose locations are region states, wt' and fwt' are obtained trivially from wt and fwt by forgetting about regions, and Δ' is defined by transforming every transition $(\ell, r) \xrightarrow{r'',\delta} (\ell', r')$ in T, where δ is labelled by $(g, \mathcal{Y})$, into a transition $(\ell, r) \xrightarrow{g'',\mathcal{Y}} (\ell', r')$, with $[\![g'']\!] = r'' \subseteq [\![g]\!]$.

Every play in $\mathcal{G}$ exists in $\mathcal{R}_N(\mathcal{G})$ as a play following a region path p, and conversely every play in $\mathcal{R}_N(\mathcal{G})$ following some region path p is contained in the play π of $\mathcal{G}$ followed by p. By projecting away the region information of $\mathcal{R}_N(\mathcal{G})$, we obtain that for all $\ell \in L$, $1/N$-regions r, and $\nu \in r$, $\mathsf{Val}_{\mathcal{G}}(\ell, \nu) = \mathsf{Val}_{\mathcal{R}_N(\mathcal{G})}((\ell, r), \nu)$.

In particular, this enrichment with region information is enough to solve (unweighted) reachability timed games, where the objective of Min is to reach the target. We can model that in the setting of weighted timed games by letting all weights be 0. Then, Min can guarantee value 0 if and only if they have a strategy guaranteeing to reach the target set of locations. These games have been studied in [AM99; JT07] showing that the existence problem is EXPTIME-complete (with at least two clocks), by using the region game as a tool. Indeed, in this context, the winner is constant over each region, which enables the possibility to only consider an untimed version of the region game, resulting in the study of a untimed reachability game. Even better, we may let all weights of locations be 1, and weights of transitions be 0, in which case the cumulated weight of a play is the time spent before reaching the target. This gives rise to the so-called *reachability-time games* where Min wants to reach the target as soon as possible: the existence problem for those games is also EXPTIME-complete [AM99; JT07].

A timed game $\mathcal{G}$ can be scaled by $N \in \mathbb{N} \setminus \{0\}$ by multiplying every constant in the guards of $\mathcal{G}$ by N. The executions in the scaled game are scaled versions of the executions in $\mathcal{G}$, where delays and valuations are multiplied by N. This means that we can restrict the constants to integer values without loss of generality. Formally, let $\mathcal{G}$ be a weighted timed game, with guards in $\mathsf{Guards}^{\mathrm{nd}}_N(\mathcal{X}, M)$, such that $N \in \mathbb{N} \setminus \{0\}$ is as small as possible for the constants in $\mathcal{G}$. There exists a weighted timed game $\mathcal{G}'$ with guards in $\mathsf{Guards}^{\mathrm{nd}}_1(\mathcal{X}, MN)$ such that all problems on $\mathcal{G}$ can be solved by considering similar problems on $\mathcal{G}'$. Moreover, if constants are encoded in binary, the size of $\mathcal{G}'$ is at most quadratic in the size of $\mathcal{G}$ [AD94].

From now on, we assume that the guards in the weighted timed games we consider have granularity $1/N = 1$, such that every constant that appears in atomic constraints belong to $\mathbb{Z}$. In this case, we omit N from previous notations about regions, such that

$1/N$-regions are simply called regions, $\mathsf{Reg}_N(\mathcal{X}, M)$ is denoted $\mathsf{Reg}(\mathcal{X}, M)$, and $\mathcal{R}_N(\mathcal{G})$ becomes $\mathcal{R}(\mathcal{G})$.

2.5 Corner-point abstraction

Despite all the interest and success of regions to study timed systems, they are not sufficient to handle weighted timed games (even with a single player). Indeed, for a single-clock case, and in a location ℓ of weight 1, spending time in ℓ from region $(0,1)$ to region $\{2\}$ can cost any possible weight in the interval $(2,3)$. We therefore must also keep a more precise information of *where we are inside each region*. This is the goal of the *corner-point abstraction* used in [Beh+01; Lar+01] to study one-player weighted timed games with non-negative weights, and generalised in [Bou+07] to handle negative weights, and in [BBL08] for the multi-cost setting.

If r is an $1/N$-region, let $\overline{r}$ denote its topological closure, i.e. the smallest zone that contains r associated to a non-strict guard. The *corners* of r are valuations in $\overline{r}$ that belong to $\mathbb{Q}_N^{\mathcal{X}}$. If r is characterised by an integral part ι and a clock ordering $0 = R_0 < R_1 < \cdots < R_m$, then ι is a corner of r. If $m = 0$ then $r = \{\iota\}$, otherwise r does not include its corners but contains valuations arbitrarily close to them. The corners of r are the vertices of the polytope $\overline{r}$, such that $\overline{r}$ is their convex hull.

Example 2.11. Corners of the green region in Figure 2.3 are the valuations $(1,0)$ and $(1,1)$.

We call *corner state* a triple (ℓ, r, v) that contains information about a region state (ℓ, r) of $\mathcal{R}_N(\mathcal{G})$, and a corner v of the $1/N$-region r. Every region has at most $|\mathcal{X}| + 1$ corners.

Definition 2.12. The corner-point abstraction $\Gamma_N(\mathcal{G})$ of a weighted timed game $\mathcal{G}$ is the weighted timed game obtained as a refinement of $\mathcal{R}_N(\mathcal{G})$ where guards on transitions are enforced to stay on one of the corners of the current $1/N$-region: the locations of $\Gamma_N(\mathcal{G})$ are all corner states of $\mathcal{R}_N(\mathcal{G})$, associated to each player accordingly, and transitions are all $(\ell, r, \mathsf{v}) \xrightarrow{g'', \mathcal{Y}} (\ell', r', \mathsf{v}')$ such that there exists $\delta = (\ell, r) \xrightarrow{g, \mathcal{Y}} (\ell', r')$ a transition of $\mathcal{R}_N(\mathcal{G})$ such that the model of guard g'' is a corner v'' satisfying the guard $\overline{g}$ (recall that $\overline{g}$ is the closed version of g), $\mathsf{v}'' \in \mathsf{PostTime}(\mathsf{v})$, $\mathsf{v}' = \mathsf{v}''[\mathcal{Y} := 0]$, and there exist two valuations $\nu \in r$, $\nu' \in r'$ such that $((\ell, r), \nu) \xrightarrow{t', \delta} ((\ell', r'), \nu')$ for some $t' \in \mathbb{R}_{\geqslant 0}$ (the latter condition ensures that the edge between corners is not spurious, i.e. created by the closure of guards). Weights of locations and transitions are trivially recovered from $\Gamma_N(\mathcal{G})$. We define the final weight function of $\Gamma_N(\mathcal{G})$ over the only valuation v reachable in location (ℓ, r, v) (with $\ell \in L_{\mathsf{t}}$) by $\mathsf{fwt}((\ell, r, \mathsf{v}), \mathsf{v}) = \lim_{\nu \to \mathsf{v}, \nu \in r} \mathsf{fwt}(\ell, \nu)$ (the limit is well defined since fwt is piecewise affine with a finite number of pieces on region r).

The weighted timed game $\Gamma_N(\mathcal{G})$ can be seen as a finite shortest-path game by removing guards, resets and rates of locations, and replacing the weights of transitions by the actual weight of jumping from one corner to another: a transition $((\ell, r), \mathsf{v}) \xrightarrow{g'', \mathcal{Y}} ((\ell', r'), \mathsf{v}')$ becomes an edge from $((\ell, r), \mathsf{v})$ to $((\ell', r'), \mathsf{v}')$ with weight $t \cdot \mathsf{wt}(\ell) + \mathsf{wt}(t)$, with $t \in \mathbb{R}_{\geqslant 0}$ the only delay such that $[\![g'']\!] = \{\mathsf{v} + t\}$. Note that delay t is necessarily a rational of the form α/N with $\alpha \in \mathbb{N}$, since it must relate corners of $1/N$-regions. In particular, this

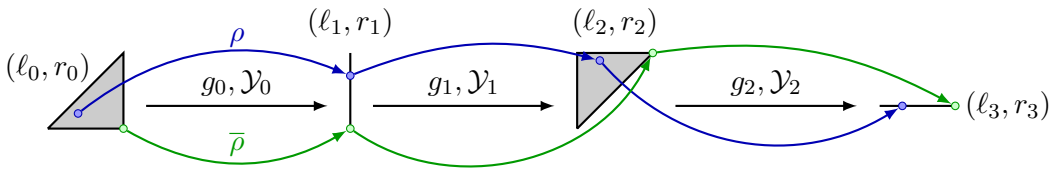

Figure 2.4 – A play ρ (in blue), its projected path p in the region game (in black), and one of its associated corner plays $\overline{\rho}$ (in green).

proves that the cumulated weight $\mathsf{wt}(\overline{\rho})$ of a finite play $\overline{\rho}$ in $\Gamma_N(\mathcal{G})$ is indeed a rational number with denominator N.

We will call *corner play* every play $\overline{\rho}$ in the corner-point abstraction $\Gamma_N(\mathcal{G})$: it can also be interpreted as an execution in $\mathcal{G}$ where all guards are closed (as explained in the definition above). It straightforwardly projects on a finite path p in the region game $\mathcal{R}_N(\mathcal{G})$: in this case, we say again that $\overline{\rho}$ follows p. Figure 2.4 depicts a play, its projected path in the region game and one of its associated corner plays.

Let $\overline{\rho}$ be a corner play following a region path p. The weight of $\overline{\rho}$ refers to its weight in $\Gamma_N(\mathcal{G})$. It is possible to find a play ρ following p close to $\overline{\rho}$, in the sense that we control the difference between their respective cumulated weights: for all $\varepsilon > 0$, there exists a play ρ in $\mathcal{G}$ following p such that $|\mathsf{wt}(\rho) - \mathsf{wt}(\overline{\rho})| \leqslant \varepsilon$. Thus, corner plays allow one to obtain faithful information on the plays that follow the same path.

Lemma 2.13. *If p is a finite region path in $\mathcal{R}_N(\mathcal{G})$, the set of cumulated weights $\{\mathsf{wt}(\rho) \mid \rho$ play of $\mathcal{G}$ following $\mathsf{p}\}$ is an interval bounded by the minimum and the maximum values of the set $\{\mathsf{wt}(\overline{\rho}) \mid \overline{\rho}$ corner play of $\Gamma_N(\mathcal{G})$ following $\mathsf{p}\}$.*

The corner-point abstraction has originally been used (under the name of *priced regions*) in [Beh+01] to solve one-player weighted timed games with only non-negative weights (called *priced timed automata*) using a branch-and-bound state-space exploration algorithm of the corner-point automaton. They notice that such algorithm is "guaranteed to be rather inefficient" since it relies on the data structure of regions, that we know to be prone to state-space explosion in practice. In [Lar+01], this drawback is solved by using *priced zones* instead, described as a usual zone (the semantics of a conjunction of atomic guards) and an affine function describing the weight over this zone: they choose to represent it with an offset, as well as derivatives of the affine function with respect to each clock-dimension. This work requires (as we do in this manuscript) all clocks to be bounded. This last restriction is removed in [BCM16] by considering abstraction techniques and effective inclusion checking of priced zones.

We will study in Chapter 3 a very restricted fragment of one-clock weighted timed game (still with two-players and both positive and negative weights) where the corner-point abstraction (with granularity $N = 1$) may faithfully be used, also advocating why this is hopeless to generalise the use of such techniques for a more general fragment.

In Chapter 4, speaking about the divergent fragment we already studied in Section 1.5 for the untimed setting, we will see that the corner-point abstraction, though with a granularity $N \neq 1$, may also be used to obtained decidability or approximation results. In this context, we will more precisely use some theoretical tools stemming from the corner-point abstraction. Notably, when focussing on a cyclic region path

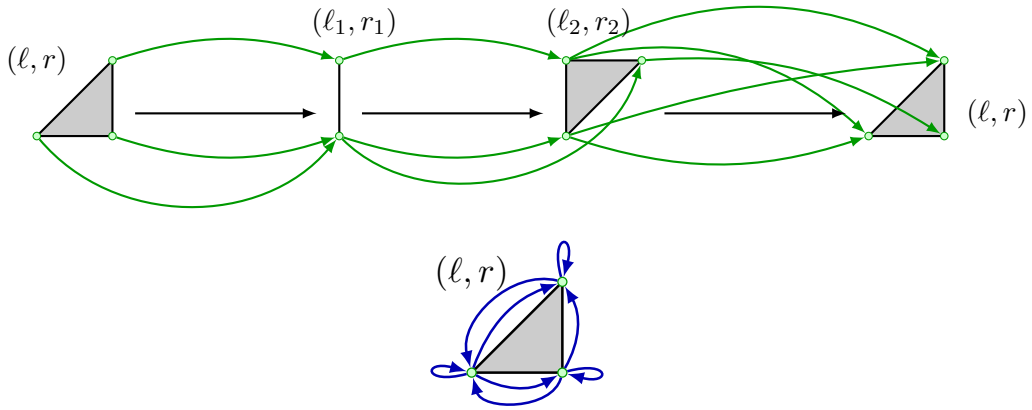

Figure 2.5 – A region cycle p in the region game (in black), its associated corner plays (in green), and its folded orbit graph (in blue). Note that there is no edge between the top right corner and the bottom right corner, as no corner play goes from the former to the latter.

$\mathsf{p} = (\ell_1, r = r_1) \xrightarrow{r'_1,\delta_1} (\ell_2, r_2) \xrightarrow{r'_2,\delta_2} \cdots \xrightarrow{r'_n,\delta_n} (\ell_1, r)$ (that we call a *region cycle* in the following), and to study some properties of the corner plays following this cycle, we only need to consider the aggregation of all the behaviours following it. Inspired by the *folded orbit graphs* (FOG) introduced in [Pur00], we define the folded orbit graph $\mathsf{FOG}(\mathsf{p})$ as a graph whose vertices are the corners states of region r, and that contains an edge from corner v to corner v' if there exists a corner play $\overline{\rho}$ from (ℓ_1, r, v) to (ℓ_1, r, v') following p. We fix $\overline{\rho}$ arbitrarily and label the edge between v and v' in $\mathsf{FOG}(\mathsf{p})$ by this corner play: it is then denoted by $\mathsf{v} \xrightarrow{\overline{\rho}} \mathsf{v}'$. An example is depicted in Figure 2.5. The folded orbit graph inherits interesting topological properties from the corner-point abstraction. For instance, an important property of the corner-point abstraction is that corner plays cannot get stuck as long as they follow a region path: if p is a region path starting from (ℓ, r) and ending in (ℓ', r'), for all corners v of r and v' of r', there exist a corner play following p that starts in (ℓ, r, v), and a corner play following p that ends in (ℓ', r', v'). With respect to the folded orbit graph, this implies that for all vertices v, there exists at least one outgoing edge $\mathsf{v} \xrightarrow{\overline{\rho}'} \mathsf{v}'$, and at least one incoming edge $\mathsf{v}'' \xrightarrow{\overline{\rho}''} \mathsf{v}$ in $\mathsf{FOG}(\mathsf{p})$.

3 Weighted Timed Games with One Clock

This chapter presents some results obtained with Thomas Brihaye, Gilles Geeraerts, Axel Haddad, Shankara Narayanan Krishna, Engel Lefaucheux, Lakshmi Manasa, and Ashutosh Trivedi, published in the conferences CONCUR [Bri+14], and FSTTCS[Bri+15a].

Table of contents

Undecidability results given in Theorem 2.4 force us to imagine restrictions to regain decidability. A first one that we study in this chapter is to restrict the number of clocks. We know that games with two (if negative weights are allowed) or three clocks are undecidable. We therefore focus here on weighted timed games with only one clock. These are very close to (untimed) shortest-path games, though the decidability status of the value or existence problem is not settled so far, in the presence of negative weights.

As a first attempt towards decidability, we study a fragment of one-clock weighted timed games by restricting the number of distinct weights allowed in locations. This permits us to encode the game into the corner-point abstraction introduced in Section 2.5, and to compute the value of a particular configuration. We will also demonstrate why this fragment is the largest one for which the corner-point abstraction technique is applicable.

To allow for more distinct weights in locations, crucial from a modelling point of view, we refer to the results of [Bou+06] showing the decidability of the value problem for one-clock weighted timed games with only non-negative weights. A 3-exponential time algorithm was first proposed to compute the value function, further refined in [Rut11; HIM13] into an exponential time algorithm. The key point of those algorithms is to reduce the problem to the computation of optimal values in a restricted family of weighted timed games called *simple weighted timed games*, where the underlying automata contain no guard, no reset, and the play is forced to stop after one time unit. More precisely, the weighted timed game $\mathcal{G}$ is decomposed into a sequence of simple weighted timed games whose value functions are computed and re-assembled to yield the value function of $\mathcal{G}$. A survey summarising results on weighted timed games can be found in [Bou15]. From a lower-bound perspective, the best known result has been obtained recently by Fearnley, Ibsen-Jensen and Ravani [FIS20], showing that the value problem of one-clock weighted timed games is PSPACE-hard, even with only non-negative weights.

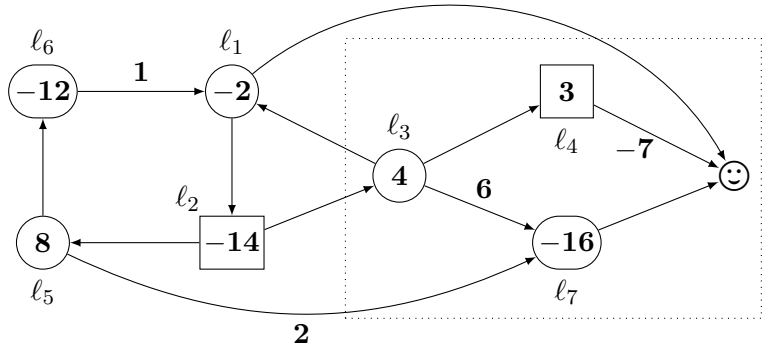

Figure 3.1 – A simple weighted timed game, where all guards are $0 \leqslant \mathsf{x} \leqslant 1$ and clock x is never reset. Transitions without labels have weight 0.

The second contribution of this chapter is thus the extension of this technique for simple weighted timed games with arbitrary weights. This is highly non trivial since several proof arguments of [Bou+06; Rut11; HIM13] heavily relies on the non-negativity of weights. What is still currently not working as good as in the non-negative weights is the decomposition of general one-clock weighted timed games into a sequence of simple weighted timed games. We still explain how to push our technique as much as possible, yet not obtaining the decidability of the whole class of one-clock weighted timed games with arbitrary weights so far.

In this chapter, all weighted timed games are restricted to only one clock, denoted by x. We will thus no longer recall this restriction. Moreover, in this chapter, we consider a valuation ν of the clock x to be a value in $\mathbb{R}_{\geqslant 0}$ (instead of a mapping $\{\mathsf{x}\} \to \mathbb{R}_{\geqslant 0}$).

3.1 Continuity of the value function

Allowing for only one clock drastically reduces the modelling power of weighted timed games, though we might benefit from urgent locations that allow us to model urgency without paying the need for a second clock. However, this severe restriction allows us to recover stronger properties, like the continuity of the value function over each region, which will appear to be very useful in the rest of the chapter.

An example of weighted timed game is given in Figure 3.1. This is a very special case of *simple* weighted timed game where all guards on transitions are $0 \leqslant \mathsf{x} \leqslant 1$ (hence this guard is not displayed and transitions are only labelled by their respective discrete weight) and the clock is never reset. It is easy to check that Min can force reaching the target location ☺ from all configurations of the game so that all values are different from $+\infty$. Let us comment on the optimal strategies for both players. From a configuration (ℓ_4, ν), with $\nu \in [0, 1]$, Max better waits until the clock takes value 1, before taking the transition to ☺. Hence, the value is $\mathsf{Val}(\ell_4, \nu) = 3(1-\nu) - 7 = -3\nu - 4$. Symmetrically, it is easy to check that Min better waits as long as possible in ℓ_7, hence their value is $\mathsf{Val}(\ell_7, \nu) = -16(1-\nu)$ for all $\nu \in [0, 1]$. However, optimal value functions are not always *that simple*, see for instance the value function of ℓ_1 displayed in Figure 3.2, which is a piecewise-affine function. To understand why value functions can be piecewise

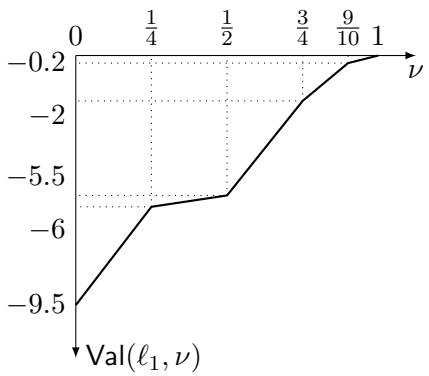

Figure 3.2 – Value function of location ℓ_1 in the weighted timed game of Figure 3.1

affine in this very simple setting, consider the sub-game enclosed in the dotted rectangle in Figure 3.1, and consider the value that Min can guarantee from a configuration of the form (ℓ_3, ν) in this sub-game. Clearly, Min must decide how long they will delay in ℓ_3 and whether they will go to ℓ_4 or ℓ_7. Their optimal value from all (ℓ_3, ν) is thus

$$\inf_{0 \leqslant t \leqslant 1-\nu} \min\big(4t + 3(1-(\nu+t)) - 7, 4t + 6 - 16(1-(\nu+t))\big) = \min(-3\nu - 4, 16\nu - 10)$$

Since $16\nu - 10 \geqslant -3\nu - 4$ if and only if $\nu \leqslant 6/19$, the best choice of Min is to move instantaneously to ℓ_7 if $\nu \in [0, 6/19]$ and to move instantaneously to ℓ_4 if $\nu \in (6/19, 1]$, hence the value function of ℓ_3 (in the sub-game) is a piecewise-affine function with two pieces.

This shape for value functions is not specific to this case, and we therefore will rely on the notion of *piecewise-affine value function* (we will update this notion in the next chapters, when generalising to more than one clock). Formally, for a set of guards $S \subseteq \mathsf{Guard}(\mathsf{x})$, we let $[\![S]\!] = \bigcup_{g \in S} [\![g]\!]$. In the context of games with only one clock, regions can be made more succinct by only considering zones $\{a\}$ and (a, b) with a and b constants appearing in the guards of the game [1]. Therefore, assuming $M_0 = 0 < M_1 < \cdots < M_k$ are all the integers appearing in the guards of S (to which we add 0 if needed), we let

$$\mathsf{Reg}_S = \{(M_i, M_{i+1}) \mid 0 \leqslant i \leqslant k-1\} \cup \{\{M_i\} \mid 0 \leqslant i \leqslant k\}$$

be the set of *(one-clock) regions* of S. Observe that Reg_S can also be seen as a set of guards.

Definition 3.1. A piecewise-affine value function over S is a function $f \colon [\![\mathsf{Reg}_S]\!] \to \mathbb{R}_\infty$ such that over each region $r \in \mathsf{Reg}_S$, f is either infinite or it is a continuous piecewise affine function with a finite set of cutpoints (points where the first derivative is not defined) $\{\kappa_1, \ldots, \kappa_p\} \subseteq \mathbb{Q}$, and satisfying $f(\kappa_i) \in \mathbb{Q}$ for all $1 \leqslant i \leqslant p$. We denote by PAF_S the set of all piecewise-affine value functions over S.

In particular, if $f(r) = \{f(\nu) \mid \nu \in r\}$ contains $+\infty$ (resp. $-\infty$) for some region r, then $f(r) = \{+\infty\}$ (resp. $f(r) = \{-\infty\}$).

1. This is inspired by a construction by Laroussinie, Markey, and Schnoebelen [LMS04].

We consider in this chapter weighted timed games where the final weight function fwt maps every location to an affine value function, i.e. for all $\ell \in L_{\mathsf{t}}$, there exists $(a,b) \in \mathbb{Q}^2$ such that for all $\nu \in [0,M]$, $\mathsf{fwt}(\ell,\nu) = a\nu + b$. We denote by $\mathsf{Reg}_{\mathcal{G}}$ the set $\mathsf{Reg}_{S_{\mathcal{G}}}$ of regions of $\mathcal{G}$ with $S_{\mathcal{G}}$ the set of all guards occurring on some transitions, and $\mathsf{PAF}_{\mathcal{G}}$ the set of all piecewise-affine value functions over $\mathsf{Reg}_{\mathcal{G}}$. We further denote by $w^{\mathrm{t}}_{\max} = \sup_{\nu\in[0,M]} \max_{\ell\in L_{\mathsf{t}}} |\mathsf{fwt}(\ell,\nu)| = \max_{\ell\in L} \max(|\mathsf{fwt}(\ell,0)|, |\mathsf{fwt}(\ell,M)|)$ (the last equality holds because we have assumed that $\mathsf{fwt}(\ell,\cdot)$ is affine).

We now discuss a first useful preliminary property of the value functions of such one-clock weighted timed games. We have already shown the determinacy of weighted timed games in a broader context (Theorem 2.3), ensuring the existence of the value function. We now state and sketch the proof of a stronger result, that is, for all ℓ, $\mathsf{Val}(\ell,\cdot)$ is a piecewise continuous function that might exhibit discontinuities *only on the corners of the regions* of $\mathsf{Reg}_{\mathcal{G}}$.

Theorem 3.2. *For all (one-clock)* WTG*s* $\mathcal{G}$*, for all* $r \in \mathsf{Reg}_{\mathcal{G}}$*, for all* $\ell \in L$*,* $\mathsf{Val}(\ell,\cdot)$ *is either equal to* $+\infty$*, or equal to* $-\infty$*, or is a continuous function over* r*.*

Proof. The proof of continuity of $\mathsf{Val}(\ell,\cdot)$ over a region r (for some fixed location ℓ) can be done by using the classical definition of continuity: for all $\nu \in r$, for all $\varepsilon > 0$, there exists $\delta > 0$ such that for all ν' with $|\nu - \nu'| \leqslant \delta$, we have $|\mathsf{Val}(\ell,\nu) - \mathsf{Val}(\ell,\nu')| \leqslant \varepsilon$. To this end, we even show a stronger property (Lipschitz-continuity):

$$\forall (\nu,\nu') \in r^2 \qquad |\mathsf{Val}(\ell,\nu) - \mathsf{Val}(\ell,\nu')| \leqslant w^L_{\max}|\nu - \nu'|$$

By symmetry, it requires to show for instance:

$$\forall (\nu,\nu') \in r^2 \qquad \mathsf{Val}(\ell,\nu') \leqslant \mathsf{Val}(\ell,\nu) + w^L_{\max}|\nu - \nu'|$$

Now comes a schema of proofs by *double simulation* that we have used many times with great success in the context of weighted timed games. By using the definition of the upper value, we need to show that for all strategies σ_{Min} of Min, there exists a strategy σ'_{Min} such that

$$\forall (\nu,\nu') \in r^2 \qquad \mathsf{Val}((\ell,\nu'),\sigma'_{\mathsf{Min}}) \leqslant \mathsf{Val}((\ell,\nu),\sigma_{\mathsf{Min}}) + w^L_{\max}|\nu - \nu'|$$

Showing such an inequality requires, for each strategy σ'_{Max} of Max, to build a strategy σ_{Max} of Max such that

$$\forall (\nu,\nu') \in r^2 \qquad \mathbf{SP}(\mathsf{play}((\ell,\nu'),\sigma'_{\mathsf{Min}},\sigma'_{\mathsf{Max}})) \leqslant \mathbf{SP}(\mathsf{play}((\ell,\nu),\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}})) + w^L_{\max}|\nu - \nu'|$$

Indeed, it is not necessary to build strategy σ_{Max} for all plays, but only for the ones that are consistent with σ_{Min}. This is equivalent to show the existence of a function g mapping plays ρ' from (ℓ,ν') consistent with σ'_{Min} (that will be chosen by Max) to plays from (ℓ,ν) consistent with σ_{Min}, and such that:

$$\mathbf{SP}(\rho') \leqslant \mathbf{SP}(g(\rho')) + w^L_{\max}|\nu - \nu'|$$

Thus, our proof strategy is to build, given a strategy σ_{Min} of Min such a strategy σ'_{Min} and a function g. We do it by induction on the length of the finite plays ρ' taken as

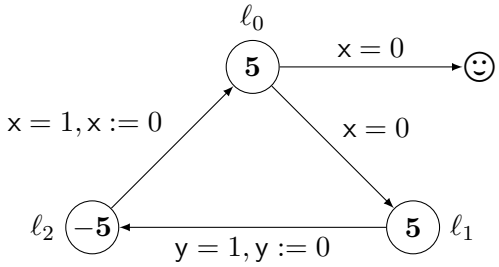

Figure 3.3 – A WTG with 2 clocks whose value function is not continuous inside a region

arguments in σ'_{Min} and g. The idea is to try to let the finite plays ρ' and $g(\rho')$ as close as possible, which is possible because of the one-clock restriction. We therefore define $\sigma'_{\mathsf{Min}}(\rho')$ and $g(\rho')$ for all plays ρ' from (ℓ, ν'), consistent with σ'_{Min} of length $k-1$ and k, respectively. This construction is performed by induction, keeping as an induction hypothesis: for all plays $\rho' = (\ell_1, \nu'_1) \xrightarrow{c'_1} \cdots \xrightarrow{c'_{k-1}} (\ell_k, \nu'_k)$ from (ℓ, ν'), consistent with σ'_{Min}, if we let $(\ell_1, \nu_1) \xrightarrow{c_1} \cdots \xrightarrow{c_{\ell-1}} (\ell_\ell, \nu_\ell) = g(\rho')$:

1. ρ' and $g(\rho')$ have the same length k;
2. for every $i \in \{1, \dots, k\}$, ν_i and ν'_i are in the same region;
3. valuations do not get further apart than at the beginning of plays: $|\nu_k - \nu'_k| \leqslant |\nu - \nu'|$;
4. cumulated weights stay close: $\mathsf{wt}(\rho') \leqslant \mathsf{wt}(g(\rho')) + w^L_{\max}(|\nu - \nu'| - |\nu_k - \nu'_k|)$.

Details of this induction proof can be found in [Bri+15a]. □

Let us consider the example in Figure 3.3, with two clocks x and y. One can easily check that, starting from a configuration $(\ell_0, (\mathsf{x} \mapsto 0, \mathsf{y} \mapsto 0.5))$, the following cycle can be taken: $(\ell_0, (\mathsf{x} \mapsto 0, \mathsf{y} \mapsto 0.5)) \xrightarrow{0,\delta_0} (\ell_1, (\mathsf{x} \mapsto 0, \mathsf{y} \mapsto 0.5)) \xrightarrow{0.5,\delta_1} (\ell_2, (\mathsf{x} \mapsto 0.5, \mathsf{y} \mapsto 0)) \xrightarrow{0.5,\delta_2} (\ell_0, (\mathsf{x} \mapsto 0, \mathsf{y} \mapsto 0.5))$, where δ_0, δ_1 and δ_2 denote respectively the transitions from ℓ_0 to ℓ_1; from ℓ_1 to ℓ_2; and from ℓ_2 to ℓ_0. Observe that the cumulated weight of this cycle is null, and that no other delays can be played, hence $\mathsf{Val}(\ell_0, (\mathsf{x} \mapsto 0, \mathsf{y} \mapsto 0.5)) = 0$. However, starting from a configuration $(\ell_0, (\mathsf{x} \mapsto 0, \mathsf{y} \mapsto 0.55))$, and following the same path, yields the cycle $(\ell_0, (\mathsf{x} \mapsto 0, \mathsf{y} \mapsto 0.55)) \xrightarrow{0,\delta_0} (\ell_1, (\mathsf{x} \mapsto 0, \mathsf{y} \mapsto 0.55)) \xrightarrow{0.45,\delta_1} (\ell_2, (\mathsf{x} \mapsto 0.45, \mathsf{y} \mapsto 0)) \xrightarrow{0.55,\delta_2} (\ell_0, (\mathsf{x} \mapsto 0, \mathsf{y} \mapsto 0.55))$ with cumulated weight -1. Hence, $\mathsf{Val}(\ell_0, (\mathsf{x} \mapsto 0, \mathsf{y} \mapsto 0.55)) = -\infty$, and the function is not continuous although both clocks values $(\mathsf{x} \mapsto 0, \mathsf{y} \mapsto 0.5)$ and $(\mathsf{x} \mapsto 0, \mathsf{y} \mapsto 0.55)$ are in the same region. Observe that this holds even for weighted timed *automata*, since our example requires only one player.

3.2 Bi-weighted timed games

As a first attempt to obtain a decidable subclass of weighted timed games, we rely on the corner-point abstraction by considering the maximal fragment where it can be used to solve the value problem. We need to restrict the set of distinct weights appearing in locations, so that no two different non-positive or non-negative weights co-exist.

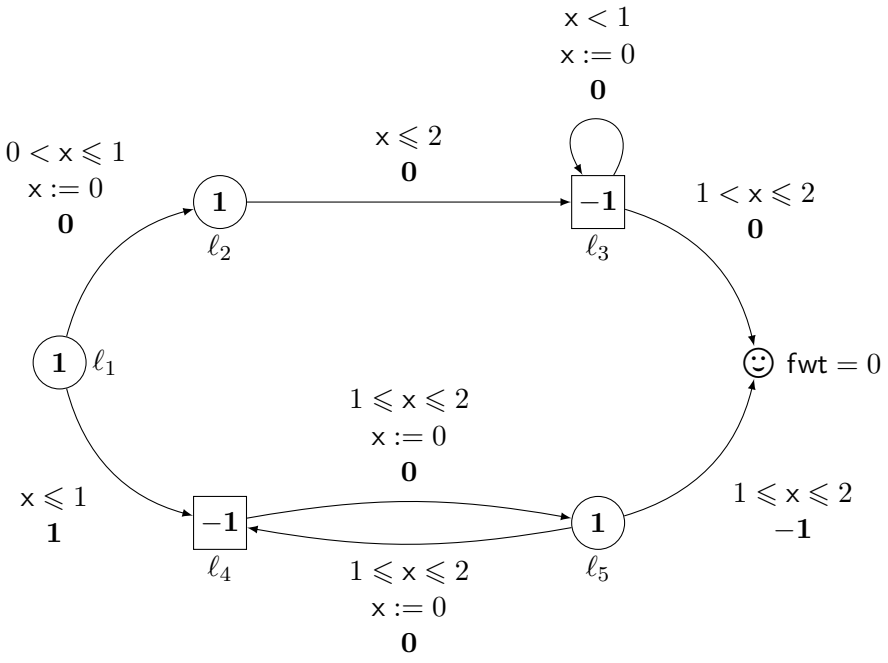

Figure 3.4 – A bi-weighted timed game

Definition 3.3. A *bi-weighted timed game* is a weighted timed game where weights of locations come from a doubleton set $\{p^-, p^+\}$ with $p^- < p^+$ two distinct elements of $\{-c, 0, d\}$, with $(c, d) \in (\mathbb{N} \setminus \{0\})^2$ (no condition is made on the weights of transitions). We let $\mathrm{1WTG}(p^-, p^+)$ be the set of such games.

Example 3.4. An example of bi-weighted timed game is given in Figure 3.4. More precisely, it belongs to the class $\mathrm{1WTG}(-1, 1)$. Another example, in the class $\mathrm{1WTG}(-2, 1)$, could be obtained by replacing all weights -1 in locations by -2. Figure 3.5 depicts two examples of weighted timed games with one clock that are not bi-weighted, because they contain three different weights $\{-1, 0, 1\}$ in locations.

Our contribution is to show that a particular value problem (for an initial valuation 0 of the clock x) is decidable for bi-weighted timed games, as well as the synthesis problem.

Theorem 3.5. *If $\mathcal{G}$ is a bi-weighted timed game, and ℓ one of its locations, the value $\mathsf{Val}(\ell, 0)$, as well as an ε-optimal strategy for Min when it exists, can be computed in pseudo-polynomial time (polynomial when weights of transitions and locations are encoded in unary). The complexity drops to polynomial time when all weights are non-negative.*

Remark. This result was originally obtained in [Bri+14] only for the case where $c = d = 1$, but it was noted by Engel Lefaucheux during an internship co-supervised by Gilles Geeraerts and myself [Lef15] that all proofs can be extended to the more general case of $c \neq d$.

The proof of Theorem 3.5 is based on the corner-point abstraction, that we redefine in the special context of games with only one clock: subdividing each zone (a, b) into three parts: two small parts around the corners of the region, and a big part in-between.

The crucial argument is to show that players can play almost-optimally by only jumping around the corners, and never into those big parts (Lemma 3.8). Then, Lemma 3.11 shows a stronger result that one can restrict attention to strategies that play closer and closer to the corners as the play goes. Finally, we combine these results to show that the corner-point abstraction of bi-weighted timed games is sufficient to compute the value as well as ε-optimal strategies (Lemma 3.14). This not only yields the desired result, but also provides us further insight into the shape of ε-optimal strategies for both players.

Reduction to η-region-uniform strategies

We say that two valuations $(\nu, \nu') \in (\mathbb{R}_{\geqslant 0})^2$ are region-equivalent, and we write $\nu \sim \nu'$, if they are in the same region of $\mathsf{Reg}_{\mathcal{G}}$. We extend the equivalence relation $\sim$ from valuations to configurations in a straightforward manner. We also generalize it to plays: for two (finite or infinite) plays $\rho = (\ell_0, \nu_0) \xrightarrow{t_0,\delta_0} \cdots$ and $\rho' = (\ell'_0, \nu'_0) \xrightarrow{t'_0,\delta'_0} \cdots$ we say that $\rho \sim \rho'$ if the lengths of ρ and ρ' are equal, and they define sequences of regional equivalent states (i.e. $\ell_i = \ell'_i$ and $\nu_i \sim \nu'_i$ for all i) and follow equivalent edges (i.e. $\delta_i = \delta'_i$ and $\nu_i + t_i \sim \nu'_i + t'_i$ for all i).

We instantiate the corner-point abstraction in this setting by isolating corners of the regions into closed balls around them, of a fixed radius η. To do so, we consider a refinement of the region equivalence relation that we call the *η-region equivalence relation*, and we write $\sim_\eta$, for a given $\eta \in (0, 1/3)$: $\nu \sim_\eta \nu'$ if both valuations are close or far from the corners, with respect to the distance η. Equivalence classes are called *η-regions*, that can be order by their lower bounds. For instance, letting $M_0 = 0 < M_1 < \cdots < M_k$ be all the integers appearing in the guards of $\mathcal{G}$, the set of η-regions is

$$\{0\}, (0, \eta], (\eta, M_1 - \eta), [M_1 - \eta, M_1), \{M_1\}, (M_1, M_1 + \eta], \ldots, [M_k - \eta, M_k), \{M_k\}, (M_k, +\infty)$$

We also extend the relation $\sim_\eta$ to configurations and plays.

We next introduce the strategies of a restricted shape with the properties that they depend only on the η-region abstraction of plays; their decision is uniform over each η-region; and they play η-close to the corners of the regions.

Definition 3.6. A strategy σ is said to be *η-region-uniform* if

- for all finite plays $\rho \sim_\eta \rho'$ ending respectively in (ℓ, ν) and (ℓ, ν'), we have $\sigma(\rho) = (t, \delta)$ and $\sigma(\rho') = (t', \delta')$ with $\delta = \delta'$ and $\nu + t \sim_\eta \nu' + t'$;
- for every finite play ρ ending in (ℓ, ν), if $\sigma(\rho) = (t, \delta)$ with $\nu + t \in (M_k, M_{k+1})$, then we have $\nu + t \in (M_k, M_k + \eta] \cup [M_{k+1} - \eta, M_{k+1})$.

We write $\mathsf{U}\Sigma^{\eta}_{\mathsf{Min}}$ and $\mathsf{U}\Sigma^{\eta}_{\mathsf{Max}}$ for the set of η-region-uniform strategies for Min and Max. We also define an upper value $\overline{\mathsf{UVal}}^{\eta}$ when both players are restricted to use only η-region-uniform strategies: for all configurations (ℓ, ν),

$$\overline{\mathsf{UVal}}^{\eta}(\ell, \nu) = \inf_{\sigma_{\mathsf{Min}} \in \mathsf{U}\Sigma^{\eta}_{\mathsf{Min}}} \sup_{\sigma_{\mathsf{Max}} \in \mathsf{U}\Sigma^{\eta}_{\mathsf{Max}}} \mathbf{SP}(\mathsf{play}((\ell, \nu), \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}}))$$

We do not need to prove that these restricted games with respect to the sets of strategies are indeed determined, which is the reason why we introduce only an upper value associated to Min.

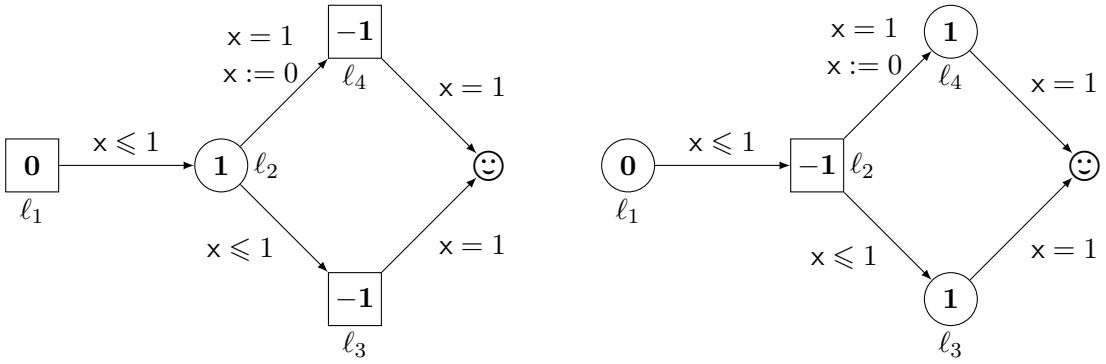

Figure 3.5 – Two weighted timed games $\mathcal{G}_1$ and $\mathcal{G}_2$ that are not bi-weighted, where all weights of transitions are 0, as well as final weights.

Example 3.7. Consider the weighted timed game $\mathcal{G}_1$ shown in Figure 3.5 (that is not bi-weighted). Notice that the only freedom when starting the play in $(\ell_1, 0)$ is the choice of a strategy of Max in $(\ell_1, 0)$. Indeed, no choice is allowed in locations ℓ_3 and ℓ_4. In configuration (ℓ_2, ν), Min may either go to ℓ_4 after a delay of $1-\nu$ resulting in a cumulated weight $1 \times (1-\nu) + (-1) \times 1 = -\nu$, or go to ℓ_3 after a delay t such that $\nu + t \leqslant 1$ resulting in a cumulated weight $t + (-1) \times (1 - \nu - t) = -1 + \nu + 2t$ that is minimal when $t = 0$. The best choice of Min in (ℓ_2, ν) therefore depends on the position of ν, relative to $1/2$: if $\nu \leqslant 1/2$, they choose the transition to ℓ_3; if $\nu > 1/2$, they choose the transition to ℓ_4. Therefore, the optimal choice for Max in ℓ_1 is to delay $1/2$ time units before jumping in location ℓ_2. We then see that $\mathsf{Val}(\ell, 0) = -1/2$. However, the optimal strategy of Max is not η-region-uniform, since they are obliged to jump *in the middle of a region*. Instead, an η-region-uniform strategy will delay t time units with $t \in [0, \eta] \cup [1 - \eta, 1]$. Hence, the upper value when players can only use η-region-uniform strategies is $\overline{\mathsf{UVal}}^{\eta}(\ell, 0) = -1$.

Contrary to this example, the next lemma shows that, in bi-weighted timed games, the upper value of the game increases when we restrict ourselves to η-region-uniform strategies. Intuitively, every cost that Max can secure with general strategies, they can also secure it with η-region-uniform strategies against η-region-uniform strategies of Min.

Lemma 3.8. *For all bi-weighted timed games $\mathcal{G}$ and $\eta \in (0, 1/3)$, $\mathsf{Val} \preccurlyeq \overline{\mathsf{UVal}}^{\eta}$.*

Reduction to η-convergent strategies

A similar result concerning the lower values of the games can be shown in case of η-region-uniform strategies. In subsequent proofs, we need a stronger result to avoid situations detailed in Example 3.10, where Max needs infinite precision to play incrementally closer to corners (as well as an infinite memory). For this reason, we restrict the shape of strategies to force them to play at distance $\eta/2^n$ of corners when playing the nth round of the play. The slight asymmetry in the definitions for the two players is exploited in proving subsequent results.

Definition 3.9. A strategy σ is said to be *η-convergent* if σ is η-region-uniform and for all finite plays ρ of length n ending in (ℓ, ν), letting $\sigma(\rho) = (t, \delta)$:

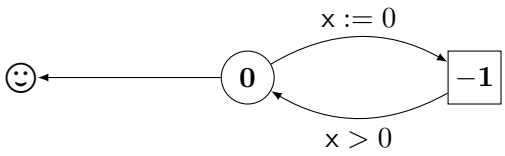

Figure 3.6 – A bi-weighted timed game where Max needs to play closer and closer to the corners

— if $\sigma \in \Sigma_{\mathsf{Min}}$, there exists k such that either $|\nu + t - M_k| \leqslant \eta/2^{n+1}$, or $t = 0$ and $\nu \in (M_k + \eta/2^{n+1}, M_k + \eta]$;
— if $\sigma \in \Sigma_{\mathsf{Max}}$, there exists k such that either $\nu + t \in \{M_k + \eta/2^{n+1}\} \cup [M_k - \eta/2^{n+1}, M_k)$, or $t = 0$ and $\nu \in (M_k + \eta/2^{n+1}, M_k + \eta]$.

We let $\mathsf{C}\Sigma^{\eta}_{\mathsf{Min}}$ and $\mathsf{C}\Sigma^{\eta}_{\mathsf{Max}}$ be respectively the set of η-convergent strategies for Min and Max, and we define, for every configuration (ℓ, ν),

$$\underline{\mathsf{CVal}}^{\eta}(\ell, \nu) = \sup_{\sigma_{\mathsf{Max}} \in \mathsf{C}\Sigma^{\eta}_{\mathsf{Max}}} \inf_{\sigma_{\mathsf{Min}} \in \mathsf{C}\Sigma^{\eta}_{\mathsf{Min}}} \mathbf{SP}(\mathsf{play}((\ell, \nu), \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}}))$$

Example 3.10. Consider the bi-weighted timed game of Figure 3.6. Max would like to exit as soon as possible from their location of weight $-1 < 0$, but because of the guard, they must spend some time before exiting. If they play according to a finite-memory strategy, there must be a bound ε such that they always stay in their state for a duration bounded from below by ε. Min can then exploit it by letting the play continue for an arbitrarily long time to achieve an arbitrarily small payoff. On the other hand, if Max plays an infinite-memory η-convergent strategy by staying in their location for a duration $\varepsilon/2^n$ in the n-th visit, they ensure a payoff $-\varepsilon$ for an arbitrarily small $\varepsilon > 0$, resulting in the value 0 of the game for both locations.

It is clear from the previous example that Max needs infinite-memory strategies to optimise their objective. The following lemma formalises our intuition that the lower value of the game decreases when we restrict ourselves to η-convergent strategies. Intuitively, every cost that Min can secure with general strategies, they can also secure it with η-convergent strategies against an η-convergent strategy of Max.

Lemma 3.11. *For all bi-weighted timed games $\mathcal{G}$ and $\eta \in (0, 1/3)$, $\underline{\mathsf{CVal}}^{\eta} \preccurlyeq \mathsf{Val}$.*

Observe that this lemma fails to hold when location weights can take more than two values as exemplified in the games $\mathcal{G}_2$ of Figure 3.5. It shows a game with three distinct weights with lower and upper value equal to $1/2$. However, when restricted to η-convergent strategies, the lower value equals 1.

So far we have shown that $\underline{\mathsf{CVal}}^{\eta} \preccurlyeq \mathsf{Val} \preccurlyeq \overline{\mathsf{UVal}}^{\eta}$. Our next goal is to find a common bound being both a lower bound on $\underline{\mathsf{CVal}}^{\eta}$ and an upper bound on $\overline{\mathsf{UVal}}^{\eta}$ by studying the value of a finite shortest-path game.

Finite abstraction of bi-weighted timed games

We now construct the finite shortest-path game $\tilde{\mathcal{G}}$, as a finite abstraction of the infinite shortest-path game $[\![\mathcal{G}]\!]$, based on η-regions. This is a refinement of the corner-point

abstraction $\Gamma(\mathcal{G})$ studied in Section 2.5. Since we have learned that η-region-uniform strategies suffice, we limit both players to remain at a distance at most η from the corners of regions. Only η-regions close to the corners are of interest, and moreover η-regions after the maximal constant M_K are not useful since $\mathcal{G}$ is bounded. Let $\mathcal{I}^\eta$ be the set of remaining useful η-regions. For example, if constant appearing are $M_1 = 2$ and $M_2 = 3$, we have $\mathcal{I}^\eta = \{\{0\}, (0,\eta], [2-\eta, 2), \{2\}, (2, 2+\eta], [3-\eta, 3), \{3\}\}$. We next define the *delay* $d(I, J)$ between two such η-regions I and J, the lower bound a of I being less than the lower bound b of J, as the closest integer to $b - a$. For example, $d((2, 2+\eta], [3-\eta, 3)) = 1$ and $d(\{0\}, [2-\eta, 2)) = 2$.

Definition 3.12. For every bi-weighted timed game $\mathcal{G}$, we let $\tilde{\mathcal{G}}$ be the finite shortest-path game $\tilde{\mathcal{G}} = \langle V_{\mathsf{Min}}, V_{\mathsf{Max}}, V_t, \Sigma, E, \mathsf{wt}\rangle$ where:

- vertices are of the form (ℓ, I) with $\ell \in L$ and $I \in \mathcal{I}^\eta$, partitioned into Min vertices, Max vertices and final vertices depending on the location only;
- $\Sigma = \mathcal{I}^\eta \times \Delta$;
- E is the set of edges $(\ell, I) \xrightarrow{J,\delta} (\ell', J')$ such that I is before J, $\delta = \ell \xrightarrow{g,\mathcal{Y}} \ell'$, $J \subseteq [\![g]\!]$ and $J' = J[\mathcal{Y} := 0]$. Action (J, δ) symbolises that the player wants to let time elapse until it reaches the η-region J, then playing transition δ of $\mathcal{G}$. It simulates any decision (t, δ) in $\mathcal{G}$, with t any delay reaching a point in J.
- $\mathsf{wt}((\ell, I) \xrightarrow{J,\delta} (\ell', J')) = \mathsf{wt}(\ell) \times d(I, J) + \mathsf{wt}(\delta)$.

Example 3.13. Consider the abstraction of the bi-weighted timed game of Figure 3.4 shown in Figure 3.7. Observe that we depict only a succinct representation of the real abstraction, since we only show the reachable part of the game from $(\ell_1, 0)$, and we have removed multiple edges (introduced due to label hiding) and kept only the most useful ones for the corresponding player. For example, consider the location $(\ell_5, \{0\})$. There are edges labelled by (J, δ) for every interval $J \in \mathcal{I}^\eta$, all directed to $(\ell_4, \{0\})$ due to a reset being performed there. We only show the best possible edge—the one with lowest weight—since location ℓ_5 belongs to Min. Each vertex contains the η-region it represents. The value of vertex $(\ell_1, 0)$ is 1, and an optimal strategy for Min is to jump in location ℓ_2, and then to delay 1 time unit, before jumping in ℓ_3.

A bi-weighted timed game $\mathcal{G}$ and its finite abstraction $\tilde{\mathcal{G}}$ relate as follows:

Lemma 3.14. *If* $\mathsf{Val}_{\tilde{\mathcal{G}}}(\ell, \{M_k\})$ *is finite for all* $0 \leqslant k \leqslant K$ *and* $\ell \in L$*, then, for all* $\varepsilon > 0$*, there is* $\eta \in (0, 1/3)$ *such that* $\overline{\mathsf{UVal}}^\eta_{\mathcal{G}}(\ell, M_k) - \varepsilon \leqslant \mathsf{Val}_{\tilde{\mathcal{G}}}(\ell, \{M_k\}) \leqslant \underline{\mathsf{CVal}}^\eta_{\mathcal{G}}(\ell, M_k) + \varepsilon$.

We now explain how to combine the previous lemmas to obtain Theorem 3.5. In case of infinite values $\mathsf{Val}_{\tilde{\mathcal{G}}}(\ell, \{M_k\})$, we can show directly that $\mathsf{Val}_{\mathcal{G}}(\ell, M_k) = \mathsf{Val}_{\tilde{\mathcal{G}}}(\ell, \{M_k\})$. Otherwise, let $\varepsilon > 0$. By Lemma 3.14, we know that there exists $\eta \in (0, 1/3)$ such that for every location $\ell \in L$ and $0 \leqslant k \leqslant K$:

$$\overline{\mathsf{UVal}}^\eta_{\mathcal{A}}(\ell, M_k) - \varepsilon \leqslant \mathsf{Val}_{\tilde{\mathcal{G}}}(\ell, \{M_k\}) \leqslant \underline{\mathsf{CVal}}^\eta_{\mathcal{A}}(\ell, M_k) + \varepsilon$$

Moreover Lemmas 3.8 and 3.11 show that:

$$\underline{\mathsf{CVal}}^\eta(\ell, M_k) \leqslant \mathsf{Val}_{\mathcal{G}}(\ell, M_k) \leqslant \overline{\mathsf{UVal}}^\eta(\ell, M_k)$$

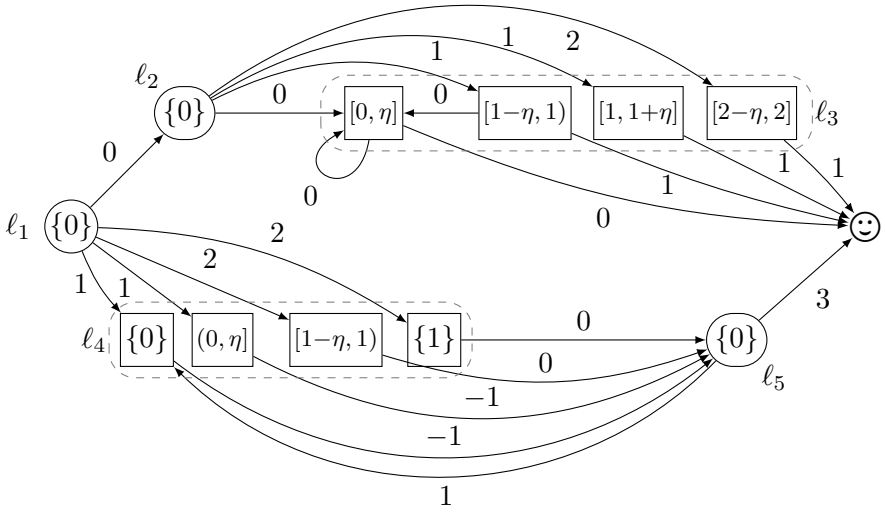

Figure 3.7 – Finite shortest-path game associated with the bi-weighted timed game of Figure 3.4.

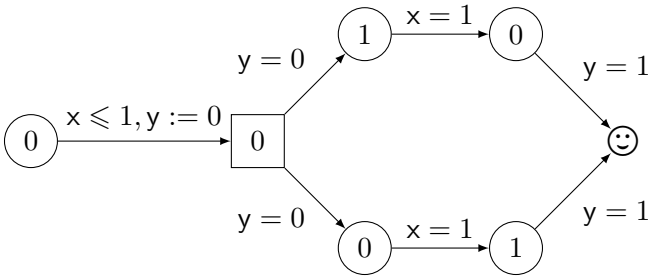

Figure 3.8 – A two-clock weighted timed game with weights of locations in $\{0, +1\}$ and value $1/2$

Both inequalities combined permit to obtain

$$\mathsf{Val}_{\tilde{\mathcal{G}}}(\ell, \{M_k\}) - \varepsilon \leqslant \mathsf{Val}_{\mathcal{G}}(\ell, M_k) \leqslant \mathsf{Val}_{\tilde{\mathcal{G}}}(\ell, \{M_k\}) + \varepsilon$$

Taking the limit when ε tends to 0, we obtain that $\mathsf{Val}(\ell, M_k) = \mathsf{Val}_{\tilde{\mathcal{G}}}(\ell, \{M_k\})$. Therefore, computing the values of the finite shortest-path game $\tilde{\mathcal{G}}$ with Theorem 1.4 allows one to compute the (exact) values of $\mathcal{G}$ in clock valuation 0 (or any M_k) in pseudo-polynomial time. Moreover, in case the values are finite, the previous reasoning allows us to compute ε-optimal strategies for both players, also in pseudo-polynomial time: Max may require infinite memory strategies, whereas finite memory is sufficient for Min.

This result implies that the value in valuations M_k of all bi-weighted timed games are either infinite or integers. This property fails if we allow more than one clock, as shown in Figure 3.8 even with only locations of weight in $\{0, 1\}$: indeed, we have $\mathsf{Val}(\ell_1, 0) = 1/2$ with ℓ_1 the leftmost location. It also fails if we allow more than two weights as was shown in Figure 3.5, where the value in $\mathcal{G}_1$ with weights in $\{-1, 0, 1\}$ is $-1/2$, while the value in $\mathcal{G}_2$ is $1/2$. This ensures that techniques of corner-point abstraction used for bi-weighted

timed games cannot be extended to a larger class of games.

Finally, notice that if all weights of $\mathcal{G}$ are non-negative, the exact computation in the finite abstraction $\tilde{\mathcal{G}}$ can be performed in polynomial time (see Theorem 1.4), resulting in a polynomial algorithm to solve bi-weighted timed games with non-negative weights.

3.3 Simple weighted timed games

As a second, more ambitious, attempt to get decidability for one-clock weighted timed games, we will solve the special case of *simple* games, where no guards and reset are allowed, and time stops after 1 time unit. Formally, an r-simple weighted timed game, with $r \in \mathbb{Q}^+ \cap [0,1]$, is a weighted timed game $\mathcal{G} = (L_{\mathsf{Min}}, L_{\mathsf{Max}}, L_f, L_{\mathrm{u}}, \Delta, \mathsf{wt}, \mathsf{fwt})$ such that for all transitions $\ell \xrightarrow{g,\mathcal{Y}} \ell'$, $[\![g]\!] = [0,r]$ (the clock is also bounded by r) and $\mathcal{Y} = \emptyset$. Hence, transitions are henceforth denoted by $\ell \to \ell'$, dropping the useless guard and reset. Then, a simple weighted timed game is a 1-simple weighted timed game. Remember that $\mathsf{Val}(\ell)$ denotes the function mapping a valuation ν to the value $\mathsf{Val}(\ell, \nu)$.

Theorem 3.15. *Let $\mathcal{G}$ be a simple weighted timed game. Then, for all locations ℓ, either $\mathsf{Val}(\ell) = +\infty$, or $\mathsf{Val}(\ell) = -\infty$, or $\mathsf{Val}(\ell)$ is a piecewise-affine value function with at most an exponential number of cutpoints (in the size of $\mathcal{G}$). The value functions $\mathsf{Val}(\ell)$ for all locations ℓ, as well as a pair of optimal strategies for both players (that always exist when no values are infinite) can be computed in exponential time.*

Let us now highlight the main steps to establish this theorem. The central argument consists in showing that all simple weighted timed games admit finitely-describable optimal strategies for both players. To this end, we rely on several new definitions.

We start by the case of Max: we will show that Max has always a memoryless optimal strategy. However, this is not sufficient to show that Max has an optimal strategy that can be *finitely* described: indeed, a memoryless strategy associates a decision to each *configuration* of the game, and there are uncountably many such configurations because of the possible values of the clock. Thus, we introduce the notion of *finite memoryless strategies* (FM-strategies for short). Such strategies partition the set $[0,1]$ of possible clock values into finitely many intervals, and ensure that the same move is played throughout each interval: this move can be either to *wait* until the clock reaches the end of the interval, or to *take immediately* a given transition.

Definition 3.16. A strategy σ is a *finite memoryless strategy* (FM-strategy for short) if it is a memoryless strategy and for all locations ℓ, there exists a finite sequence of rationals $0 \leqslant \nu_1^\ell < \nu_2^\ell < \cdots < \nu_k^\ell = 1$ and transitions $(\delta_1, \ldots, \delta_k) \in \Delta^k$ such that

1. for all $1 \leqslant i \leqslant k$, either for all $\nu \in (\nu_{i-1}^\ell, \nu_i^\ell]$, $\sigma(\ell, \nu) = (0, \delta_i)$, or for all $\nu \in (\nu_{i-1}^\ell, \nu_i^\ell]$ $\sigma(\ell, \nu) = (\nu_i^\ell - \nu, \delta_i)$ (assuming $\nu_0^\ell = \min(0, \nu_1^\ell)$);
2. if $\nu_1^\ell > 0$, then $\sigma(\ell, 0) = (\nu_1^\ell, \delta_1)$.

We let $\mathsf{pts}(\sigma)$ be the set of ν_i^ℓ for all ℓ and i, and $\mathcal{I}(\sigma)$ be the set of all successive intervals generated by $\mathsf{pts}(\sigma)$. Finally, we let $|\sigma| = |\mathcal{I}(\sigma)|$ be the size of σ.

The case of Min is more involved, since the need for memory already seen in finite shortest-path games accumulates with the difficulties due to time. We thus mimic the switching strategies used in the untimed setting: *first* play an FM-strategy σ^1_{Min} that

allows them to lower the weight of the play sufficiently low, by forcing negative cycles (if any); *second*, play another FM-strategy σ^2_{Min} that ensures that the target will eventually be reached. A threshold α describes after how many rounds to switch. Computing the latter of these two strategies is easy: σ^2_{Min} is an attractor strategy, which guarantees Min to reach the target (when possible). Thus, the main difficulty in identifying optimal switching strategies is to characterise σ^1_{Min}. It must be a negative-cycle strategy, as in the untimed setting. We update the definition in this timed setting though. Those strategies are FP-strategies which guarantee that all cycles taken have cost of -1 at most, without necessarily guaranteeing to eventually reach the target (as this will be taken care of by σ^2_{Min}).

Definition 3.17. A negative-cycle strategy (NC-strategy for short) σ_{Min} of Min is an FP-strategy such that for all plays $\rho = (\ell_1, \nu) \xrightarrow{c_1} \cdots \xrightarrow{c_{k-1}} (\ell_k, \nu')$ conforming to σ_{Min} with $\ell_1 = \ell_k$, and ν, ν' in the same interval of $\mathcal{I}(\sigma_{\mathsf{Min}})$, the sum of weights of *discrete transitions* is at most -1, i.e. $\mathsf{wt}(\ell_1, \ell_2) + \cdots + \mathsf{wt}(\ell_{k-1}, \ell_k) \leqslant -1$.

This definition allows one to find an upper bound on the weight of the plays following such NC-strategies, by iteratively removing sub-plays starting and ending in the same pair of locations and intervals:

Lemma 3.18. *Let ρ be a finite play conforming to an NC-strategy σ_{Min}. Then*

$$\mathsf{wt}(\rho) \leqslant w^{\mathrm{t}}_{\max} + w^{L}_{\max} + (2|\sigma_{\mathsf{Min}}| - 1) \times |L| w^{\Delta}_{\max} - (|\rho| - 3|\sigma_{\mathsf{Min}}|)/|L| \,.$$

Among those NC-strategies, we identify the ones that guarantee Min to obtain the optimal value (or better) when the target is reached, but do not necessarily guarantee to reach the target. For an NC-strategy σ_{Min}, and a configuration s, we let the *fake value* be defined by

$$\mathsf{fake}^{\sigma_{\mathsf{Min}}}_{\mathcal{G}}(s) = \sup\{\mathsf{wt}(\rho) \mid \rho \in \mathsf{Play}s, \sigma_{\mathsf{Min}}, \rho \text{ reaches a target}\}$$

Thus, $\mathsf{fake}^{\sigma_{\mathsf{Min}}}_{\mathcal{G}}(s) \leqslant \mathsf{Val}^{\sigma_{\mathsf{Min}}}(s)$. We then say that an NC-strategy σ_{Min} is *fake-optimal* if its fake value, in every configuration, is equal to the optimal value of the configuration in the game, i.e. $\mathsf{fake}^{\sigma_{\mathsf{Min}}}_{\mathcal{G}}(s) = \mathsf{Val}_{\mathcal{G}}(s)$ for all configurations s.

The final step is to combine such an NC-strategy σ^1_{Min} with the attractor strategy σ^2_{Min}, to obtain a switching strategy σ'_{Min} that satisfies $\mathsf{Val}^{\sigma'_{\mathsf{Min}}}_{\mathcal{G}}(s) \leqslant \mathsf{fake}^{\sigma_{\mathsf{Min}}}_{\mathcal{G}}(s)$ for all configurations s (only supposing that $\mathsf{Val}_{\mathcal{G}}(\ell, \nu) \neq +\infty$, for all ℓ and ν, i.e. that strategy σ^2_{Min} allows Min to reach the target from all configurations). In particular, if σ^1_{Min} is fake-optimal, then $\mathsf{Val}^{\sigma'_{\mathsf{Min}}}_{\mathcal{G}}(s) \leqslant \mathsf{fake}^{\sigma_{\mathsf{Min}}}_{\mathcal{G}}(s) = \mathsf{Val}_{\mathcal{G}}(s)$ which means that σ'_{Min} is optimal.

Thus, proving Theorem 3.15 reduces to showing that

1. Min has a fake-optimal NC-strategy;
2. Max has an optimal FM-strategy; and
3. $\mathsf{Val}_{\mathcal{G}}(\ell)$ is a piecewise-affine value function of $\mathsf{PAF}_{\mathcal{G}}$, for all locations ℓ.

Moreover, all strategies and value functions can be computed in exponential time. We gather these three properties in saying that $\mathcal{G}$ is *finitely optimal*.

The proof is by induction on the number of non-urgent locations of the game. In Section 3.4, we address the base case of games with urgent locations only (where no time

Algorithm 3: `solveInstant`$(\mathcal{G},\nu)$

Input: r-simple weighted timed game $\mathcal{G} = (L_{\mathsf{Min}}, L_{\mathsf{Max}}, L_f, L_{\mathrm{u}}, \boldsymbol{\varphi}, \Delta, \mathsf{wt})$, $\nu \in [0, r]$

foreach $\ell \in L$ **do**
  if $\ell \in L_f$ **then** $\mathsf{X}(\ell) := \varphi_\ell(\nu)$ **else** $\mathsf{X}(\ell) := +\infty$
repeat
  $\mathsf{X}_{pre} := \mathsf{X}$
  foreach $\ell \in L_{\mathsf{Max}}$ **do** $\mathsf{X}(\ell) := \max_{(\ell,\ell')\in\Delta} \big(\mathsf{wt}(\ell,\ell') + \mathsf{X}_{pre}(\ell')\big)$
  foreach $\ell \in L_{\mathsf{Min}}$ **do** $\mathsf{X}(\ell) := \min_{(\ell,\ell')\in\Delta} \big(\mathsf{wt}(\ell,\ell') + \mathsf{X}_{pre}(\ell')\big)$
  foreach $\ell \in L$ such that $\mathsf{X}(\ell) < -(|L|-1)w^{\Delta}_{\max} - w^{\mathrm{t}}_{\max}$ **do** $\mathsf{X}(\ell) := -\infty$
until $\mathsf{X} = \mathsf{X}_{pre}$
return X

can elapse). Since these are very close to the *untimed* shortest-payoff games studied in Chapter 1, we adapt the algorithm in this work and obtain the `solveInstant` function (Algorithm 3). This function can also compute $\mathsf{Val}(\ell, 1)$ for all ℓ and all games $\mathcal{G}$ (even with non-urgent locations) since time cannot elapse anymore when the clock has value 1. Next, using the continuity result of Theorem 3.2, we can detect locations ℓ where $\mathsf{Val}(\ell,\nu) \in \{+\infty, -\infty\}$, for all $\nu \in [0,1]$, and remove them from the game. Finally, in Section 3.5, we handle simple weighted timed games with non-urgent locations by refining the technique of [Bou+06; Rut11] (that originally worked only with non-negative weights).

3.4 Simple weighted timed games with only urgent locations

Throughout this section, we consider an r-simple weighted timed game $\mathcal{G}$ where all non-final locations are urgent, i.e. $L_{\mathrm{u}} \cup L_f = L_{\mathsf{Min}} \cup L_{\mathsf{Max}}$. Since all locations in $\mathcal{G}$ are urgent, we may extract from a play $\rho = (\ell_0,\nu) \xrightarrow{c_0} (\ell_1,\nu) \xrightarrow{c_1} \cdots$ the clock values, as well as weights $c_i = \mathsf{wt}(\ell_i, \ell_{i+1})$, hence denoting plays by their sequence of locations $\ell_0\ell_1\cdots$.

We first explain how we can compute the value function of the game for a *fixed* clock value $\nu \in [0,r]$: more precisely, we will compute the vector $(\mathsf{Val}(\ell,\nu))_{\ell\in L}$ of values for all locations. Since no time can elapse, it consists in an adaptation of the techniques developed in Chapter 1 to solve (untimed) shortest-path games. The main difference concerns the weights being rational (and not integers) and the presence of affine final value functions. Still, because of the particular structure of the game $\mathcal{G}$ (where a real price is paid only on the target location, all other weights being integers), for all plays ρ, $\mathsf{wt}(\rho)$ is a value from the set $\mathbb{Z}_{\nu,\boldsymbol{\varphi}} = \mathbb{Z} + \{\varphi_\ell(\nu) \mid \ell \in L_f\}$. We further define $\mathbb{Z}^{+\infty}_{\nu,\boldsymbol{\varphi}} = \mathbb{Z}_{\nu,\boldsymbol{\varphi}} \cup \{+\infty\}$. Clearly, $\mathbb{Z}_{\nu,\boldsymbol{\varphi}}$ contains at most $|L_f|$ values between two consecutive integers, i.e.

$$\forall i \in \mathbb{Z} \quad |[i, i+1] \cap \mathbb{Z}_{\nu,\boldsymbol{\varphi}}| \leqslant |L_f| \tag{3.1}$$

We can then summarise the adaptation of the untimed techniques in Algorithm 3. The crucial argument to show termination is to notice that the possible values taken by the components of vector X, apart from $\{-\infty, +\infty\}$, are among

$$\mathsf{PossVal}_\nu = [-(|L|-1)w^{\Delta}_{\max} - w^{\mathrm{t}}_{\max}, |L|w^{\Delta}_{\max} + w^{\mathrm{t}}_{\max}] \cap \mathbb{Z}_{\nu,\boldsymbol{\varphi}}$$

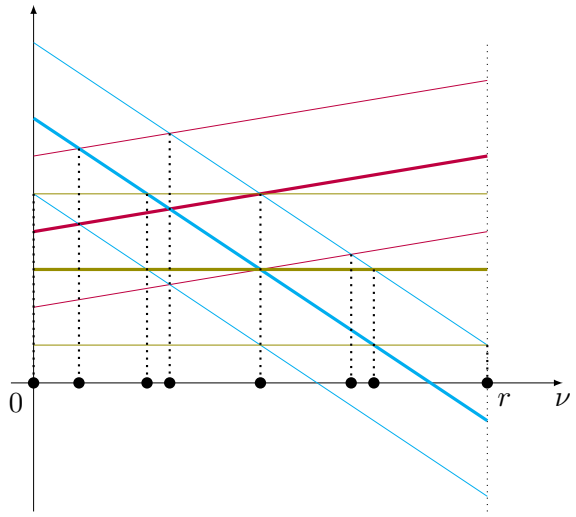

Figure 3.9 – Network of affine functions defined by $\mathsf{F}_{\mathcal{G}}$: functions in bold are final affine functions of $\mathcal{G}$, whereas non-bold ones are their translations with weights $k \in [-(|L|-1)w^{\Delta}_{\max}, |L|w^{\Delta}_{\max}] \cap \mathbb{Z}$. $\mathsf{PossCP}_{\mathcal{G}}$ is the set of abscissas of intersections points, represented by black disks.

that has a cardinality bounded by $|L_f| \times \big((2|L|-1)w^{\Delta}_{\max} + 2w^{\mathrm{t}}_{\max} + 1\big)$. Thus the loop in the algorithm terminates after a number of steps at most $|L_f| \times |L| \times \big((2|L|-1)w^{\Delta}_{\max} + 2w^{\mathrm{t}}_{\max} + 1\big) + |L|$, pseudo-polynomial with respect to the size of $\mathcal{G}$.

Finally, optimal FM-strategies for Max, and fake-optimal NC-strategies for Min can be computed as in Chapter 1.

Now let us explain how we can reduce the computation of $\mathsf{Val}(\ell)$ (for all ℓ) to a *finite number of calls* to Algorithm 3. We first study a precise characterisation of these functions, in particular showing that these are piecewise-affine value functions of $\mathsf{PAF}_{\{[0,r]\}}$.

We first define the set $\mathsf{F}_{\mathcal{G}}$ of affine functions over $[0,r]$ as follows:

$$\mathsf{F}_{\mathcal{G}} = \{k + \varphi_\ell \mid \ell \in L_f \wedge k \in [-(|L|-1)w^{\Delta}_{\max}, |L|w^{\Delta}_{\max}] \cap \mathbb{Z}\}$$

Observe that this set is finite and that its cardinality is $2|L|^2 w^{\Delta}_{\max}$, pseudo-polynomial in the size of $\mathcal{G}$. Moreover, this set contains enough information to compute the value of the game in each possible value of the clock: for all $\ell \in L$, for all $\nu \in [0,r]$: if $\mathsf{Val}(\ell,\nu)$ is finite, then there is $f \in \mathsf{F}_{\mathcal{G}}$ such that $\mathsf{Val}(\ell,\nu) = f(\nu)$.

Using the continuity of the value functions (Theorem 3.2), this shows that all the cutpoints of Val are intersections of functions from $\mathsf{F}_{\mathcal{G}}$, i.e. belong to the set of *possible cutpoints*

$$\mathsf{PossCP}_{\mathcal{G}} = \{\nu \in [0,r] \mid \exists f_1, f_2 \in \mathsf{F}_{\mathcal{G}} \quad f_1 \neq f_2 \wedge f_1(\nu) = f_2(\nu)\}$$

This set is depicted in Figure 3.9 on an example. Observe that $\mathsf{PossCP}_{\mathcal{G}}$ contains at most $|\mathsf{F}_{\mathcal{G}}|^2 = 4|L_f|^4(w^{\Delta}_{\max})^2$ points (also pseudo-polynomial in the size of $\mathcal{G}$) since all functions in $\mathsf{F}_{\mathcal{G}}$ are affine, and can thus intersect at most once with every other function. Moreover, $\mathsf{PossCP}_{\mathcal{G}} \subseteq \mathbb{Q}$, since all functions of $\mathsf{F}_{\mathcal{G}}$ take rational values in 0 and $r \in \mathbb{Q}$. Thus, for all ℓ, $\mathsf{Val}(\ell)$ is a piecewise-affine value function (with cutpoints in $\mathsf{PossCP}_{\mathcal{G}}$ and

pieces from $\mathsf{F}_{\mathcal{G}}$), and we can characterise it completely by computing only its value on its cutpoints. Hence, we can reconstruct $\mathsf{Val}(\ell)$ by calling Algorithm 3 on each rational clock value $\nu \in \mathsf{PossCP}_{\mathcal{G}}$. From the optimal strategies thus computed, we can also reconstruct a fake-optimal NC-strategy for Min and an optimal FM-strategy for Max, hence:

Proposition 3.19. *Every r-simple weighted timed $\mathcal{G}$ with only urgent locations is finitely optimal. Moreover, for all locations ℓ, the piecewise-affine value function $\mathsf{Val}(\ell)$ has cutpoints in $\mathsf{PossCP}_{\mathcal{G}}$ of cardinality $4|L_f|^4(w^{\Delta}_{\max})^2$, pseudo-polynomial in the size of $\mathcal{G}$.*

3.5 Finite optimality of general simple weighted timed games

In this section, we consider simple weighted timed games with non-urgent locations. We first prove that all such games are finitely optimal. Then, we introduce Algorithm 4 to compute optimal values and strategies. Throughout the section, we fix a simple weighted timed game $\mathcal{G} = (L_{\mathsf{Min}}, L_{\mathsf{Max}}, L_f, L_{\mathrm{u}}, \boldsymbol{\varphi}, \Delta, \mathsf{wt})$. Before presenting our core contributions, let us explain how we can detect locations with infinite values. As already argued, we can compute $\mathsf{Val}(\ell, 1)$ for all ℓ assuming all locations are urgent, since time cannot elapse anymore when the clock has value 1. This can be done with Algorithm 3. Then, by continuity, $\mathsf{Val}(\ell, 1) = +\infty$ (resp., $\mathsf{Val}(\ell, 1) = -\infty$) if and only if $\mathsf{Val}(\ell, \nu) = +\infty$ (resp., $\mathsf{Val}(\ell, \nu) = -\infty$) for all $\nu \in [0, 1]$. We remove from the game all locations with infinite value without changing the values of other locations. Thus, we henceforth assume that $\mathsf{Val}(\ell, \nu) \in \mathbb{R}$ for all (ℓ, ν).

To prove finite optimality and to establish correctness of our algorithm, we rely in both cases on a construction that consists in decomposing $\mathcal{G}$ into a sequence of simple weighted timed games with *more urgent locations*. Intuitively, such games are easier to solve since they are closer to an untimed game (in particular, when all locations are urgent, we can apply the techniques of Section 3.4). More precisely, given a set L' of non-urgent locations, and a clock value $r_0 \in [0, 1]$, we define a (possibly infinite) sequence of clock values $1 = r_0 > r_1 > \cdots$ and a sequence $\mathcal{G}_{L',r_0}, \mathcal{G}_{L',r_1}, \ldots$ of simple weighted timed games such that

1. all locations of $\mathcal{G}$ are also present in each $\mathcal{G}_{L',r_i}$, except that the locations of L' are now urgent; and
2. for all $i \geqslant 0$, the value function of $\mathcal{G}_{L',r_i}$ is equal to $\mathsf{Val}_{\mathcal{G}}$ on the interval $[r_{i+1}, r_i]$. Hence, we can re-construct $\mathsf{Val}_{\mathcal{G}}$ by assembling well-chosen parts of the value functions of the games $\mathcal{G}_{L',r_i}$ (assuming $\inf_i r_i = 0$).

To formalise these constructions, let $r \in [0, 1]$ and $\vec{x} = (x_\ell)_{\ell \in L}$ be a vector of rational values. Then, we let $\mathtt{wait}(\mathcal{G}, r, \vec{x})$ be the r-simple weighted timed game in which both players may now decide, in all non-urgent locations ℓ, to wait until the clock takes value r, and then to stop the game, adding the weight x_ℓ to the current price of the play. Formally, $\mathtt{wait}(\mathcal{G}, r, \vec{x}) = (L_{\mathsf{Min}}, L_{\mathsf{Max}}, L'_f, L_{\mathrm{u}}, \boldsymbol{\varphi}', T', \mathsf{wt}')$ is such that

— $L'_f = L_f \uplus \{\ell^f \mid \ell \in L \setminus L_{\mathrm{u}}\}$;

— for all $\ell' \in L_f$ and $\nu \in [0, r]$, $\varphi'_{\ell'}(\nu) = \varphi_{\ell'}(\nu)$, for all $\ell \in L \setminus L_{\mathrm{u}}$, $\varphi'_{\ell^f}(\nu) = (r - \nu) \cdot \mathsf{wt}(\ell) + x_\ell$;

— $T' = T \cup \{(\ell, [0, r], \bot, \ell^f) \mid \ell \in L \setminus L_{\mathrm{u}}\}$;

— for all $\delta \in T'$, $\mathsf{wt}'(\delta) = \mathsf{wt}(\delta)$ if $\delta \in T$, and $\mathsf{wt}'(\delta) = 0$ otherwise.

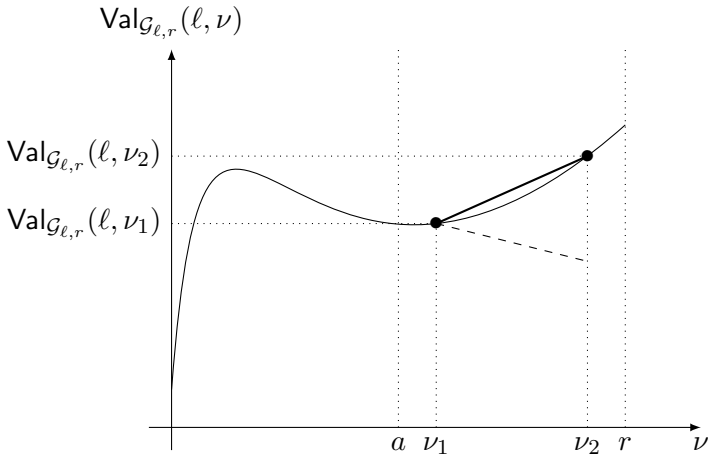

Figure 3.10 – The condition (3.2) (in the case $L' = \emptyset$ and $\ell \in L_{\mathsf{Min}}$): graphically, it means that the slope between every two points of the plot in $[a, r]$ (represented with a thick line) is greater than or equal to $-\mathsf{wt}(\ell)$ (represented with dashed line).

Then, we consider the game $\mathtt{wait}\big(\mathcal{G}, r, (\mathsf{Val}_{\mathcal{G}}(\ell, r))_{\ell \in L}\big)$ obtained thanks by letting $\vec{x}$ be the value of $\mathcal{G}$ in r. This first transformation does not alter the value of the game, for clock values before r:

Lemma 3.20. *For all $\nu \in [0, r]$ and locations ℓ, $\mathsf{Val}_{\mathcal{G}}(\ell, \nu) = \mathsf{Val}_{\mathtt{wait}(\mathcal{G}, r, (\mathsf{Val}_{\mathcal{G}}(\ell, r))_{\ell \in L})}(\ell, \nu)$.*

Next, we make locations urgent. For a set $L' \subseteq L \setminus L_{\mathrm{u}}$ of non-urgent locations, we let $\mathcal{G}_{L',r}$ be the simple weighted game obtained from $\mathtt{wait}\big(\mathcal{G}, r, (\mathsf{Val}_{\mathcal{G}}(\ell, r))_{\ell \in L}\big)$ by making urgent every location ℓ of L'. Observe that, although all locations $\ell \in L'$ are now urgent in $\mathcal{G}_{L',r}$, their clones ℓ^f allow the players to wait until r. When L' is a singleton $\{\ell\}$, we write $\mathcal{G}_{\ell,r}$ instead of $\mathcal{G}_{\{\ell\},r}$.

While the construction of $\mathcal{G}_r$ does not change the value of the game, turning locations urgent *does*. Yet, we can characterise an interval $[a, r]$ on which the value functions of $\mathcal{H} = \mathcal{G}_{L',r}$ and $\mathcal{H}^+ = \mathcal{G}_{L' \cup \{\ell\},r}$ coincide, as stated by the next proposition. The interval $[a, r]$ depends on the *slopes* of the pieces of $\mathsf{Val}_{\mathcal{H}^+}$ as depicted in Figure 3.10: for each location ℓ of Min, the slopes of the pieces of $\mathsf{Val}_{\mathcal{H}^+}$ contained in $[a, r]$ should be $\leqslant -\mathsf{wt}(\ell)$ (and $\geqslant -\mathsf{wt}(\ell)$ when ℓ belongs to Max). It is proved by lifting optimal strategies of $\mathcal{H}^+$ into $\mathcal{H}$, and strongly relies on the determinacy result of Theorem 2.3. Hereafter, we denote the slope of $\mathsf{Val}_{\mathcal{G}}(\ell)$ in-between ν and ν' by $\mathrm{slope}^{\ell}_{\mathcal{G}}(\nu, \nu') = \frac{\mathsf{Val}_{\mathcal{G}}(\ell,\nu') - \mathsf{Val}_{\mathcal{G}}(\ell,\nu)}{\nu' - \nu}$.

Lemma 3.21. *Let $0 \leqslant a < r \leqslant 1$, $L' \subseteq L \setminus L_{\mathrm{u}}$ and $\ell \notin L' \cup L_{\mathrm{u}}$ a non-urgent location of Min (respectively, Max). Assume that $\mathcal{G}_{L' \cup \{\ell\},r}$ is finitely optimal, and for all $a \leqslant \nu_1 < \nu_2 \leqslant r$*

$$\mathrm{slope}^{\ell}_{\mathcal{G}_{L' \cup \{\ell\},r}}(\nu_1, \nu_2) \geqslant -\mathsf{wt}(\ell) \quad (\textit{respectively, } \leqslant -\mathsf{wt}(\ell)) \tag{3.2}$$

Then, for all $\nu \in [a, r]$ and $\ell' \in L$, $\mathsf{Val}_{\mathcal{G}_{L' \cup \{\ell\},r}}(\ell', \nu) = \mathsf{Val}_{\mathcal{G}_{L',r}}(\ell', \nu)$. Furthermore, fake-optimal NC-strategies and optimal FM-strategies in $\mathcal{G}_{L' \cup \{\ell\},r}$ are also fake-optimal and

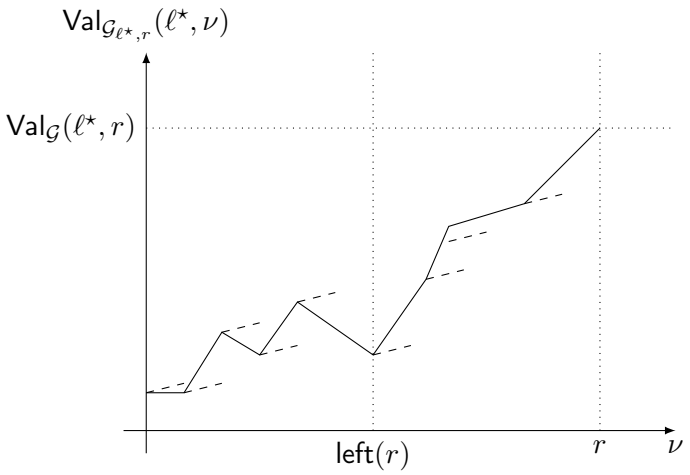

Figure 3.11 – In this example $L' = \{\ell^\star\}$ and $\ell^\star \in L_{\mathsf{Min}}$. $\mathsf{left}(r)$ is the leftmost point such that all slopes on its right are smaller than or equal to $-\mathsf{wt}(\ell^\star)$ in the graph of $\mathsf{Val}_{\mathcal{G}_{\ell^\star,r}}(\ell^\star,\nu)$. Dashed lines have slope $-\mathsf{wt}(\ell^\star)$.

optimal over $[a,r]$ *in* $\mathcal{G}_{L',r}$.

Given an SWTG $\mathcal{G}$ and some *finitely optimal* $\mathcal{G}_{L',r}$, we now characterise precisely the left endpoint a of the maximal interval ending in r where the value functions of $\mathcal{G}$ and $\mathcal{G}_{L',r}$ coincide, with the operator $\mathsf{left}_{L'}\colon (0,1] \to [0,1]$ (or simply left, if L' is clear) defined as

$$\mathsf{left}_{L'}(r) = \inf\{r' \leqslant r \mid \forall \ell \in L\ \forall \nu \in [r',r]\ \mathsf{Val}_{\mathcal{G}_{L',r}}(\ell,\nu) = \mathsf{Val}_{\mathcal{G}}(\ell,\nu)\}\,.$$

By continuity of the value (Theorem 3.2), this infimum exists and $\mathsf{Val}_{\mathcal{G}}(\ell,\mathsf{left}_{L'}(r)) = \mathsf{Val}_{\mathcal{G}_{L',r}}(\ell,\mathsf{left}_{L'}(r))$. Moreover, $\mathsf{Val}_{\mathcal{G}}(\ell)$ is a piecewise-affine value function on $[\mathsf{left}(r),r]$, since $\mathcal{G}_{L',r}$ is finitely optimal. However, this definition of $\mathsf{left}(r)$ is semantical. Yet, building on the ideas of Lemma 3.21, we can effectively compute $\mathsf{left}(r)$, given $\mathsf{Val}_{\mathcal{G}_{L',r}}$. We claim that $\mathsf{left}_{L'}(r)$ is the *minimal clock value* such that for all locations $\ell \in L' \cap L_{\mathsf{Min}}$ (respectively, $\ell \in L' \cap L_{\mathsf{Max}}$), the slopes of the affine sections of the piecewise-affine value function $\mathsf{Val}_{\mathcal{G}_{L',r}}(\ell)$ on $[\mathsf{left}(r),r]$ are at least (at most) $-\mathsf{wt}(\ell)$. Hence, $\mathsf{left}(r)$ can be obtained (see Figure 3.11) by inspecting iteratively, for all ℓ of Min (respectively, Max), the slopes of $\mathsf{Val}_{\mathcal{G}_{L',r}}(\ell)$ by decreasing clock values until we find a piece with a slope greater than $-\mathsf{wt}(\ell)$ (respectively, smaller than $-\mathsf{wt}(\ell)$). This enumeration of the slopes is effective as $\mathsf{Val}_{\mathcal{G}_{L',r}}$ has finitely many pieces, by hypothesis. Moreover, this guarantees that $\mathsf{left}(r) < r$.

It only remains to choose L'. With what we have obtained above, a natural choice is to let $L' = \{\ell^\star\}$ with $\ell^\star$ a non-urgent location of *minimum weight* (i.e. for all $\ell \in L$, $\mathsf{wt}(\ell^\star) \leqslant \mathsf{wt}(\ell)$). Given $r_0 \in [0,1]$, we let $r_0 > r_1 > \cdots$ be the decreasing sequence of clock values such that $r_i = \mathsf{left}_{\ell^\star}(r_{i-1})$ for all $i > 0$. Then, one can reconstruct $\mathsf{Val}_{\mathcal{G}}$ on $[\inf_i r_i, r_0]$ from the value functions of the (potentially infinite) sequence of games $\mathcal{G}_{L',r_0}$, $\mathcal{G}_{L',r_1},\ldots$ Assuming finite optimality of those games (by induction), this proves that $\mathcal{G}$ is finitely optimal *under the condition* that $r_0 > r_1 > \cdots$ eventually stops, i.e. $r_i = 0$ for some i. The crucial argument is to relate the optimal value functions with the final value functions:

Lemma 3.22. *Assume that $\mathcal{G}_{L',r}$ is finitely optimal. If $\mathsf{Val}_{\mathcal{G}_{L',r}}(\ell^\star)$ is affine on a non-singleton interval $I \subseteq [0,r]$ with a slope greater*[2] *than $-\mathsf{wt}(\ell^\star)$, then there exists $f \in \mathsf{F}_{\mathcal{G}}$ (see definition in page 60) such that for all $\nu \in I$, $\mathsf{Val}_{\mathcal{G}_{L',r}}(\ell^\star, \nu) = f(\nu)$.*

Then, the proof of termination of the sequence of r_i's is achieved by showing that, for all i, the owner of $\ell^\star$ has a strictly better strategy in configuration $(\ell^\star, r_{i+1})$ than waiting until r_i in location $\ell^\star$, and relying on the finiteness of $\mathsf{F}_{\mathcal{G}}$:

Lemma 3.23. *If $\mathcal{G}_{\ell^\star,r_i}$ is finitely optimal for all $i \geqslant 0$, then*

1. *if $\ell^\star \in L_{\mathsf{Min}}$ (respectively, L_{Max}), $\mathsf{Val}_{\mathcal{G}}(\ell^\star, r_{i+1}) < \mathsf{Val}_{\mathcal{G}}(\ell^\star, r_i) + (r_i - r_{i+1})\mathsf{wt}(\ell^\star)$ (respectively, $\mathsf{Val}_{\mathcal{G}}(\ell^\star, r_{i+1}) > \mathsf{Val}_{\mathcal{G}}(\ell^\star, r_i) + (r_i - r_{i+1})\mathsf{wt}(\ell^\star)$), for all i; and*
2. *there is $i \leqslant |\mathsf{F}_{\mathcal{G}}|^2 + 2$ such that $r_i = 0$.*

By iterating this construction, we make all locations urgent iteratively, and obtain the finite optimality of all simple weighted timed games. Moreover, for all locations ℓ, $\mathsf{Val}_{\mathcal{G}}(\ell)$ can be shown to have at most $\mathcal{O}\left((w^{\Delta}_{\max}|L|^2)^{2|L|+2}\right)$ cutpoints.

This construction suggests a *recursive* algorithm in the spirit of [Bou+06; Rut11] (for non-negative weights). From a simple weighted timed game $\mathcal{G}$ with minimal non-urgent location $\ell^\star$, solve recursively $\mathcal{G}_{\ell^\star,1}$, $\mathcal{G}_{\ell^\star,\mathsf{left}(1)}$, $\mathcal{G}_{\ell^\star,\mathsf{left}(\mathsf{left}(1))}, \ldots$, handling the base case where all locations are urgent with Algorithm 3. While our results above show that this is correct and terminates, we show that this recursion can be *avoided*. Instead of turning locations urgent one at a time, we make them all urgent at once, and compute directly the sequence $\mathcal{G}_{L\setminus L_{\mathrm{u}},1}$, $\mathcal{G}_{L\setminus L_{\mathrm{u}},\mathsf{left}(1)}, \ldots$ of simple weighted timed games with only urgent locations. Its proof of correctness relies on the finite optimality of simple weighted timed games and, again, on our basic result linking the value functions of $\mathcal{G}$ and games $\mathcal{G}_{L',r_i}$.

However, the key argument of Lemma 3.23 that ensures termination of the recursive algorithm cannot be applied in this case. Instead, we rely on the following result, stating, that there will be at least one cutpoint of $\mathsf{Val}_{\mathcal{G}}$ in each interval $[\mathsf{left}(r), r]$. Observe that this lemma relies on the fact that $\mathcal{G}$ is finitely optimal, hence the need to first prove this fact independently with the sequence $\mathcal{G}_{\ell^\star,1}, \mathcal{G}_{\ell^\star,\mathsf{left}(1)}, \mathcal{G}_{\ell^\star,\mathsf{left}(\mathsf{left}(1))}, \ldots$. Termination then follows from the fact that $\mathsf{Val}_{\mathcal{G}}$ has finitely many cutpoints by finite optimality.

Lemma 3.24. *Let $r_0 \in (0,1]$ such that $\mathcal{G}_{L',r_0}$ is finitely optimal. Suppose that $r_1 = \mathit{left}_{L'}(r_0) > 0$, and let $r_2 = \mathit{left}_{L'}(r_1)$. There exists $r' \in [r_2, r_1)$ and $\ell \in L'$ such that*

1. *$\mathsf{Val}_{\mathcal{G}}(\ell)$ is affine on $[r', r_1]$, of slope equal to $-\mathsf{wt}(\ell)$, and*
2. *$\mathsf{Val}_{\mathcal{G}}(\ell, r_1) \neq \mathsf{Val}_{\mathcal{G}}(\ell, r_0) + \mathsf{wt}(\ell)(r_0 - r_1)$.*

As a consequence, $\mathsf{Val}_{\mathcal{G}}(\ell)$ has a cutpoint in $[r_1, r_0)$.

Algorithm 4 implements this new non-recursive algorithm. In our algorithm, we combine value functions thanks to the $\rhd$ operator. Let $f \in \mathsf{PAF}_S$ and $f' \in \mathsf{PAF}_{S'}$ be two piecewise-affine value functions on sets of guards S, S', such that $[\![S]\!] \cap [\![S']\!]$ is a singleton. We let $f \rhd f'$ be the function in $\mathsf{PAF}_{S\cup S'}$ such that $(f \rhd f')(\nu) = f(\nu)$ for all $\nu \in [\![\mathsf{Reg}_S]\!]$, and $(f \rhd f')(\nu) = f'(\nu)$ for all $\nu \in [\![\mathsf{Reg}_{S'}]\!] \setminus [\![\mathsf{Reg}_S]\!]$.

Each iteration of the **while** loop computes a new game in the sequence $\mathcal{G}_{L\setminus L_{\mathrm{u}},1}$, $\mathcal{G}_{L\setminus L_{\mathrm{u}},\mathsf{left}(1)}, \ldots$, solves it thanks to Algorithm 3, and thus computes a new portion of

2. For this result, the order does not depend on the owner of the location, but on the fact that $\ell^\star$ has minimal weight amongst locations of $\mathcal{G}$.

Algorithm 4: $\texttt{solve}(\mathcal{G})$

Input: SWTG $\mathcal{G} = (L_{\mathsf{Min}}, L_{\mathsf{Max}}, L_f, L_{\mathrm{u}}, \boldsymbol{\varphi}, \Delta, \mathsf{wt})$

1 $\vec{f} = (f_\ell)_{\ell \in L} := \texttt{solveInstant}(\mathcal{G}, 1)$ /* $f_\ell \colon \{1\} \to \mathbb{R}_\infty$ */
2 $r := 1$
3 **while** $0 < r$ **do** /* Invariant: $f_\ell \colon [r,1] \to \mathbb{R}_\infty$ */
4 $\quad \mathcal{G}' := \texttt{wait}(\mathcal{G}, r, \vec{f}(r))$ /* r-SWTG $\mathcal{G}' = (L_{\mathsf{Min}}, L_{\mathsf{Max}}, L'_f, L'_{\mathrm{u}}, \boldsymbol{\varphi}', T', \mathsf{wt}')$ */
5 $\quad L'_{\mathrm{u}} := L'_{\mathrm{u}} \cup L$ /* every location is made urgent */
6 $\quad b := r$
7 $\quad$ **repeat** /* Invariant: $f_\ell \colon [b,1] \to \mathbb{R}_\infty$ */
8 $\quad\quad a := \max(\mathsf{PossCP}_{\mathcal{G}'} \cap [0,b))$
9 $\quad\quad \vec{x} = (x_\ell)_{\ell \in L} := \texttt{solveInstant}(\mathcal{G}', a)$ /* $x_\ell = \mathsf{Val}_{\mathcal{G}'}(\ell, a)$ */
10 $\quad\quad$ **if** $\forall \ell \in L_{\mathsf{Min}}\ \frac{f_\ell(b)-x_\ell}{b-a} \leqslant -\mathsf{wt}(\ell) \wedge \forall \ell \in L_{\mathsf{Max}}\ \frac{f_\ell(b)-x_\ell}{b-a} \geqslant -\mathsf{wt}(\ell)$ **then**
11 $\quad\quad\quad$ **foreach** $\ell \in L$ **do** $f_\ell := \left(\nu \in [a,b] \mapsto f_\ell(b) + (\nu - b)\frac{f_\ell(b)-x_\ell}{b-a}\right) \rhd f_\ell$
12 $\quad\quad\quad b := a\,;\ stop := false$
13 $\quad\quad$ **else** $stop := true$
14 $\quad$ **until** $b = 0$ *or* $stop$
15 $\quad r := b$
16 **return** $\vec{f}$

$\mathsf{Val}_{\mathcal{G}}$ on an interval on the left of the current point $r \in [0,1]$. More precisely, the vector $(\mathsf{Val}_{\mathcal{G}}(\ell,1))_{\ell \in L}$ is first computed in line 1. Then, the algorithm enters the **while** loop, and the game $\mathcal{G}'$ obtained when reaching line 6 is $\mathcal{G}_{L \setminus L_{\mathrm{u}},1}$. Then, the algorithm enters the **repeat** loop to analyse this game. Instead of building the whole value function of $\mathcal{G}'$, it builds only the parts of $\mathsf{Val}_{\mathcal{G}'}$ that coincide with $\mathsf{Val}_{\mathcal{G}}$. It proceeds by enumerating the possible cutpoints a of $\mathsf{Val}_{\mathcal{G}'}$, starting in r, by decreasing clock values (line 8), and computes the value of $\mathsf{Val}_{\mathcal{G}'}$ in each cutpoint thanks to Algorithm 3 (line 9), which yields a new piece of $\mathsf{Val}_{\mathcal{G}'}$. Then, the **if** in line 10 checks whether this new piece coincides with $\mathsf{Val}_{\mathcal{G}}$, using the condition given by Lemma 3.21. If it is the case, the piece of $\mathsf{Val}_{\mathcal{G}'}$ is added to f_ℓ (line 11); **repeat** is stopped otherwise. When exiting the **repeat** loop, variable b has value $\mathsf{left}(1)$. Hence, at the next iteration of the **while** loop, $\mathcal{G}' = \mathcal{G}_{L \setminus L_{\mathrm{u}}, \mathsf{left}(1)}$ when reaching line 6. By continuing this reasoning inductively, one concludes that the successive iterations of the **while** loop compute the sequence $\mathcal{G}_{L \setminus L_{\mathrm{u}},1}$, $\mathcal{G}_{L \setminus L_{\mathrm{u}},\mathsf{left}(1)}, \ldots$ as announced, and rebuilds $\mathsf{Val}_{\mathcal{G}}$ from them. Termination in exponential time is ensured by Lemma 3.24: each iteration of the **while** loop discovers at least one new cutpoint of $\mathsf{Val}_{\mathcal{G}}$, and there are at most exponentially many (note that a tighter bound on this number of cutpoints would entail a better complexity of our algorithm).

3.6 Towards non-simple weighted timed games

In [Bou+06; Rut11; HIM13], *general* weighted timed games with *non-negative weights* are solved by reducing them to a finite sequence of simple weighted timed games, by eliminating guards and resets. It is thus natural to try and adapt these techniques to the general case, in which case Algorithm 4 would allow us to solve general simple weighted

timed games with arbitrary weights. Let us explain where are the difficulties of such a generalisation.

The technique used to remove strict guards from the transitions of a weighted timed game $\mathcal{G}$, i.e. guards of the form $(a, b]$, $[b, a)$ or (a, b) with $a, b \in \mathbb{N}$, consists in enhancing the locations with regions while keeping an equivalent game $\mathcal{G}'$. This technique *can* be adapted to arbitrary weights, as we have already seen before: this is the purpose of the *corner-point abstraction* of Definition 2.12, that moreover allows one to consider a corner of a region, that is a point of its topological closure, as a point of the region itself. Interestingly, we can transform an ε-optimal strategy of $\mathcal{G}'$ into an ε'-optimal strategy of $\mathcal{G}$ with $\varepsilon' < 2\varepsilon$ and vice-versa.

The technique to handle resets, however, consists in *bounding* the number of clock resets that can occur in each play following an optimal strategy of Min or Max. Then, the weighted timed game can be *unfolded* into a *reset-acyclic* game with the same value. By reset-acyclic, we mean that no cycles in the configuration graph visit a transition with a reset. This reset-acyclic weighted timed game can be decomposed into a finite number of components that contain no reset and are linked by transitions with resets. These components can be solved iteratively, from the bottom to the top, turning them into simple weighted timed games. Thus, if we *assume* that the weighted timed games we are given as input *are* reset-acyclic, we can solve them in *exponential time*, and our techniques show that their value functions are piecewise-affine with at most exponentially many cutpoints.

In [Bou+06], the authors showed that with one-clock weighted timed games and non-negative weights only we could bound the sufficient number of resets by n, the number of locations, without changing the value functions. Unfortunately, the arguments to bound the number of resets do not hold for arbitrary weights, as shown by the weighted timed game in Figure 3.12. We claim that $\mathsf{Val}(\ell_0) = 0$; that Min has no optimal strategies, but a family of ε-optimal strategies $\sigma^{\varepsilon}_{\mathsf{Min}}$ with respective value ε; and that each $\sigma^{\varepsilon}_{\mathsf{Min}}$ requires *memory whose size depends on* ε and might *yield a play visiting at least* $1/\varepsilon$ *times the reset* between ℓ_0 and ℓ_1 (hence the number of resets cannot be bounded). For all $\varepsilon > 0$, $\sigma^{\varepsilon}_{\mathsf{Min}}$ consists in: waiting $1 - \varepsilon$ time units in ℓ_0, then going to ℓ_1 during the $\lceil 1/\varepsilon \rceil$ first visits to ℓ_0; and to go directly to ℓ_f afterwards. Against $\sigma^{\varepsilon}_{\mathsf{Min}}$, Max has two possible choices:

1. either wait 0 time unit in ℓ_1, wait ε time units in ℓ_2, then reach ℓ_f; or
2. wait ε time unit in ℓ_1 then force the cycle by going back to ℓ_0 and wait for Min's next move.

Thus, all plays according to $\sigma^{\varepsilon}_{\mathsf{Min}}$ will visit a sequence of locations which is either of the form $\ell_0(\ell_1\ell_0)^k\ell_1\ell_2\ell_f$, with $0 \leqslant k < \lceil 1/\varepsilon \rceil$; or of the form $\ell_0(\ell_1\ell_0)^{\lceil 1/\varepsilon \rceil}\ell_f$. In the former case, the weight of the play will be $-k\varepsilon + 0 + \varepsilon = -(k-1)\varepsilon \leqslant \varepsilon$; in the latter, $-\varepsilon(\lceil 1/\varepsilon \rceil) + 1 \leqslant 0$. This shows that $\mathsf{Val}(\ell_0) = 0$, but there is no optimal strategy as none of these strategies allow one to guarantee a value of 0 (neither does the strategy that waits 1 time unit in ℓ_0).

If bounding the number of resets is not possible in the general case, it could be done if one adds constraints on the cycles of the game. This kind of restriction was used in [BCR14] where they introduce the notion of robust games. Such games requires among other things that every play starting and ending in the same pair of location and region has either a positive weight or a weight smaller than -1. Here we require a less powerful

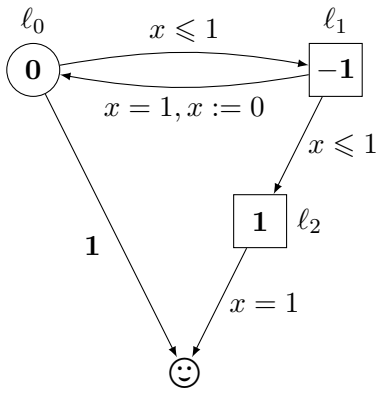

Figure 3.12 – A weighted timed game where the number of resets in optimal plays cannot be bounded a priori.

assumption as we put this restriction only on cycles containing a reset.

Definition 3.25. A negative-reset-acyclic weighted timed game is a weighted timed game where for every location $\ell \in L$ and every cyclic finite play ρ starting and ending in $(\ell, 0)$, either $\mathsf{wt}(\rho) \geqslant 0$ or $\mathsf{wt}(\rho) < -1$.

The weighted timed game of Figure 3.12 is not a negative-reset-acyclic weighted timed game as the play $(\ell_0, 0) \xrightarrow{0} (\ell_1, 1 - 1/2) \xrightarrow{-1/2} (\ell_0, 0)$ is a cycle containing a reset and with a negative weight strictly greater than -1. On the contrary, in Figure 3.13, we show a negative-reset-acyclic weighted timed game and its region game. Here, every cycle containing a reset is between ℓ_0 and ℓ_1 and such cycles have at most weight -1. The value $\mathsf{Val}(\ell_0, 0)$ is 0 in this game but no strategies for Max can achieve it because of the guard $x > 0$. As this guard is not strict anymore in the region game, both players have an optimal strategy (this is not always the case).

This allows us to give a bound on the value of a negative-reset-acyclic weighted timed game $\mathcal{G}$. Letting $k = |\mathsf{Reg}_{\mathcal{G}}|$ be the number of regions, for all configurations (ℓ, ν):

— either $\mathsf{Val}(\ell, \nu) \in \{-\infty, +\infty\}$,

— or $\underbrace{-|L|Mw^L_{\max} - |L|^2(|L|+2)w^\Delta_{\max}}_{\mathsf{Val}_{inf}} \leqslant \mathsf{Val}(\ell, \nu) \leqslant \underbrace{|L|Mw^L_{\max} + |L|kw^\Delta_{\max}}_{\mathsf{Val}^{sup}}$.

Using this, one can give a bound on the number of cycles that it is sufficient to allow. The idea is that if a reset is taken twice and the generated cycle (with a single clock, traversing twice the same transition with a reset creates a cycle) has positive weight, either Min can modify their strategy so that it does not take this cycle or the value of the game is $+\infty$ as Max can stop Min to reach an accepting state. On the contrary if the cycle has negative weight, then by definition, this weight is less than -1. Thus by allowing enough such cycles, as we have bounds on the values of the game, we know when we will have enough cycles to get under the lower bound of the value of the game. By solving the copies of the game, if we reach a value that is smaller than the lower bound of the value, then it means that the value is $-\infty$. More precisely, the value of a negative-reset-acyclic weighted timed game can be computed by solving $k = 2n(\mathsf{Val}^{sup} - \mathsf{Val}_{inf})$ WTGs without resets and using the same set of guards. Moreover, from ε-optimal strategies on those

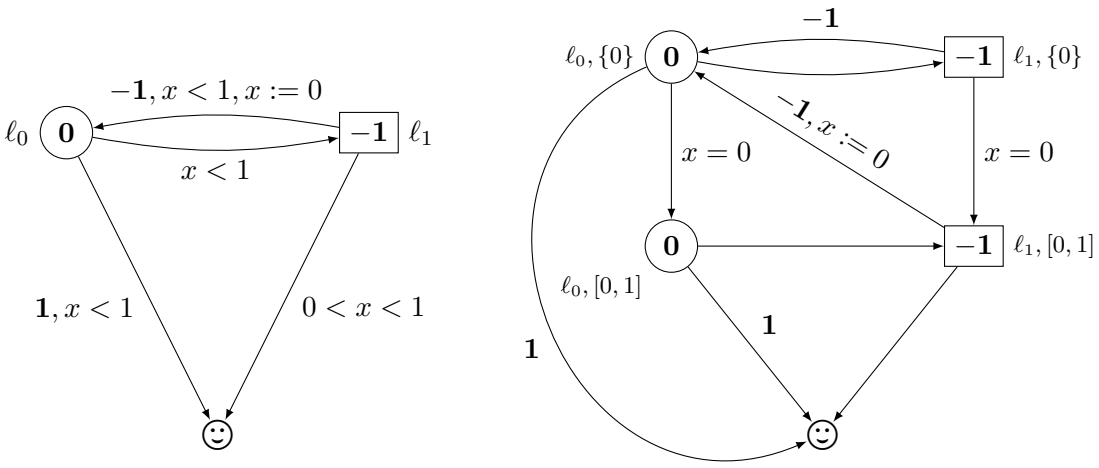

Figure 3.13 – A negative-reset-acyclic weighted timed game and its region game (some guards were removed for a better readability)

k games, we can build $k\varepsilon$-optimal strategies in the original game. With this, we can conclude:

Theorem 3.26. *Let $\mathcal{G}$ be a negative-reset-acyclic weighted timed game. Then for every location $\ell \in L$, the function $\mathsf{Val}(\ell)$ is computable in exponential time, and is piecewise-affine with at most an exponential number of cutpoints. Moreover, for every $\varepsilon > 0$, there exists (and we can effectively compute) ε-optimal strategies for both players.*

The robust games defined in [BCR14] restricted to one-clock are a subset of the negative-reset-acyclic weighted timed games, therefore their value is computable with the same complexity. While we cannot (yet) extend the computation of the value to all (one-clock) weighted timed games, we can still obtain information on the nature of the value function:

Theorem 3.27. *The value functions of all one-clock weighted timed games are piecewise-affine value functions with at most exponentially many cutpoints.*

Proof. Let $\mathcal{G}$ be a one-clock weighted timed game. Let us replace all transitions $(\ell, g, \{\mathsf{x}\}, \ell')$ resetting the clock by $(\ell, g, \emptyset, \ell'')$, where ℓ'' is a new final location with $\varphi_{\ell''} = \mathsf{Val}_{\mathcal{G}}(\ell, 0)$: observe that $\mathsf{Val}_{\mathcal{G}}(\ell, 0)$ exists even if we cannot compute it, so this transformation is well-defined. This yields a negative-reset-acyclic weighted timed game $\mathcal{G}'$ such that $\mathsf{Val}_{\mathcal{G}'} = \mathsf{Val}_{\mathcal{G}}$. □

Conclusion

In this chapter, we have pushed previously existing algorithms to solve one-clock weighted timed games, in order to be able to model arbitrary (positive and negative) weights. We obtained a positive result for the special case of simple games, where no guards and resets are allowed, that is easily extendable to the presence of guards. This complexity

result is comparable with previously obtained results in the case of non-negative weights only [HIM13; Rut11], even though we follow different paths to prove termination and correction (due to the presence of negative weights). In order to push our algorithm as far as we can, and allow for some resetting transitions, we have introduced the class of negative-reset-acyclic games for which we obtain the same result: as a particular case, we can solve all weighted timed games with one clock for which the clock is reset in every cycle of the underlying region automaton.

As future works, it is appealing to solve the full class of weighted timed games with arbitrary weights and one clock. We have shown why our technique breaks down in this more general setting, thus it could be interesting to study the difficult negative cycles without reset as their own, with different techniques. This is the object of current promising works performed with Julie Parreaux and Pierre-Alain Reynier.

Another question, wide open, is to solve weighted timed games with only non-negative weights and two clocks, since in the non-negative case, the undecidability barrier is at three clocks (see Theorem 2.4). Simpler fragments could be investigated to go in this direction, for instance where only one of the two clocks is allowed to be tested and reset, while the other clock follows the simple restriction investigated in this chapter.

4 Value Iteration Methods: (Almost-)Divergent Weighted Timed Games

This chapter presents results obtained with Damien Busatto-Gaston, Julie Parreaux, and Pierre-Alain Reynier, published in the conferences FoSSaCS [BMR17], FSTTCS [BMR18] and ICALP [MPR21], and follows the presentation of the submitted journal article [BMR21].

Table of contents

At this stage, it seems that we have almost a clear picture of the decidability frontier: a large fragment of one-clock weighted timed games is decidable (Chapter 3), while it becomes undecidable with two clocks (in the presence of negative weights). To push further the decidability frontier, we must thus consider other kinds of restrictions on the weighted timed games than the number of clocks.

In [Bou+04b], a completely different kind of restriction was studied, to obtain decidability: in the presence of non-negative weights only, a weighted timed game satisfies the strictly non-Zeno cost property when every finite play ρ in $\mathcal{G}$ following a cycle in the region game $\mathsf{Reg}_{\mathcal{G}}$ satisfies $\mathsf{wt}(\rho) \geqslant 1$. Then, as plays grow, their weight ultimately grows above any fixed bound, and diverges towards $+\infty$ for an infinite execution. Thus, it is only necessary to unfold the game graph to a finite depth in order to compute optimal values and (ε-)optimal strategies. This can be done thanks to the technique of [ABM04], solving tree-shaped weighted timed games with non-negative weights. They develop a heavy machinery to cut the clock space into small chunks (regions are not fitted for that purpose, since, as we have already seen, players may have to play in the middle of regions to play optimally) over which the value function is affine. Instead of unfolding and compute the value of a tree-shaped game, we may understand this technique as another appearance of the value iteration paradigm we have already exploited in previous chapters. We start this chapter by recalling this paradigm in the timed setting. We then exploit it in the setting of games with arbitrary weights and arbitrarily many clocks, where no decidable fragments were know before our contribution.

4.1 Value functions and value iteration algorithm

The value of a game has been defined as a mapping of each configuration (ℓ, ν) to a value in $\mathbb{R}_\infty$. We call *value functions* such mappings from $L \times \mathbb{R}_{\geqslant 0}^{\mathcal{X}}$ to $\mathbb{R}_\infty$. If $\mathbf{V}$ represents a value function, we denote by $\mathbf{V}_\ell$ the mapping $\nu \mapsto \mathbf{V}(\ell, \nu)$. As observed in [Bou+04b; ABM04], one step of the game is summarised in the following operator $\mathcal{F}$ mapping each value function $\mathbf{V}$ to a value function $\mathbf{V}' = \mathcal{F}(\mathbf{V})$ defined by $\mathbf{V}'_\ell(\nu) = \mathsf{fwt}(\ell, \nu)$ if $\ell \in L_{\mathsf{t}}$, and otherwise

$$\mathbf{V}'_\ell(\nu) = \begin{cases} \sup_{(\ell,\nu) \xrightarrow{t,e} (\ell',\nu')} \left[t \cdot \mathsf{wt}(\ell) + \mathsf{wt}(e) + \mathbf{V}_{\ell'}(\nu') \right] & \text{if } \ell \in L_{\mathsf{Max}} \\ \inf_{(\ell,\nu) \xrightarrow{t,e} (\ell',\nu')} \left[t \cdot \mathsf{wt}(\ell) + \mathsf{wt}(e) + \mathbf{V}_{\ell'}(\nu') \right] & \text{if } \ell \in L_{\mathsf{Min}} \end{cases} \tag{4.1}$$

where $(\ell, \nu) \xrightarrow{t,e} (\ell', \nu')$ ranges over valid transitions in $\mathcal{G}$. This generalises a similar operator used in Chapter 1 in the untimed setting.

Then, starting from $\mathbf{V}^0$ mapping every configuration (ℓ, ν) to $+\infty$, except for the targets mapped to $\mathsf{fwt}(\ell, \nu)$, we let $\mathbf{V}^i = \mathcal{F}(\mathbf{V}^{i-1})$ for all $i > 0$. The value function $\mathbf{V}^i$ contains the value $\mathsf{Val}^i_{\mathcal{G}}$, which is intuitively what Min can guarantee when forced to reach the target in at most i steps. More formally, we let $\mathsf{wt}^i(\rho)$ the weight of a maximal play ρ at horizon i, as $\mathsf{wt}(\rho)$ if ρ reaches a target in at most i steps, and $+\infty$ otherwise. Then, $\mathsf{Val}^i_{\mathcal{G}}(\ell, \nu) = \inf_{\sigma_{\mathsf{Min}}} \sup_{\sigma_{\mathsf{Max}}} \mathsf{wt}^i(\mathsf{play}((\ell, \nu), \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}}))$ refers to the value at horizon i.

We compare value functions componentwise: if $\mathbf{V}, \mathbf{V}'$ are two value function, we let $\mathbf{V} \leqslant \mathbf{V}'$ if $\mathbf{V}(\ell, \nu) \leqslant \mathbf{V}'(\ell, \nu)$ for all configurations (ℓ, ν). Notice that $\mathcal{F}$ is a monotonic operator, i.e. if $\mathbf{V} \leqslant \mathbf{V}'$, then $\mathcal{F}(\mathbf{V}) \leqslant \mathcal{F}(\mathbf{V}')$. Moreover, $\mathcal{F}(\mathbf{V}^0) \leqslant \mathbf{V}^0$ since $\mathbf{V}^0$ maps every non-target state to $+\infty$, and target state keep the same value. It follows that the sequence $(\mathbf{V}^i)_{i \in \mathbb{N}}$ is non-increasing, as $\mathbf{V}^i = \mathcal{F}^i(\mathbf{V}^0) \geqslant \mathcal{F}^i(\mathcal{F}(\mathbf{V}^0)) = \mathbf{V}^{i+1}$.

We have seen in Chapter 1 that, in the case of an (untimed) shortest-path game, this sequence converges in finite time towards the greatest fixed point of $\mathcal{F}$ that appears to be the value of the game. Such a computation of the greatest fixed point in finite time might not be possible in the timed setting, since the value problem is undecidable in general. We will come back later on this technique, also used by [Bou+04b] for special classes of weighted timed games.

We will heavily rely on the class of *piecewise-affine value functions*, and a way to efficiently encode them, as developed in [ABM04]. This class is closed by the application of operator $\mathcal{F}$. Let n denote the number of clocks, such that $\mathcal{X} = \{\mathsf{x}_1, \ldots, \mathsf{x}_n\}$, and $\ell \in L$ a location of $\mathcal{G}$. An *affine value function* is a mapping $\mathbf{V}_\ell \colon [0, M)^{\mathcal{X}} \to \mathbb{R}_\infty$ such that for all $\nu \in [0, M)^{\mathcal{X}}$,

$$\mathbf{V}_\ell(\nu) = a_1 \cdot \nu(\mathsf{x}_1) + \cdots + a_n \cdot \nu(\mathsf{x}_n) + b$$

with partial derivatives $a_i \in \mathbb{Q}$ for $1 \leqslant i \leqslant n$, and additive constant $b \in \mathbb{Q}$. In this case, we say that $\mathbf{V}_\ell$ is defined by the equation $\mathsf{y} = a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b$ where the variable $\mathsf{y} \notin \mathcal{X}$ refers to $\mathbf{V}_\ell(\mathsf{x}_1, \ldots, \mathsf{x}_n)$. We also consider infinite mappings $\nu \mapsto +\infty$ and $\nu \mapsto -\infty$ to be affine value functions, defined by $\mathsf{y} = +\infty$ and $\mathsf{y} = -\infty$, respectively.

Intuitively, we define a piecewise-affine value function as a partition of $[0, M)^{\mathcal{X}}$ into finitely many polyhedra, called cells, each equipped by an affine value function. Formally, an affine inequality is an equation I of the form

$$a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b \prec 0$$

where $b \in \mathbb{Q}$ is the additive constant of I, $\prec \in \{<, \leqslant\}$ is its comparison operator, and for every $1 \leqslant i \leqslant n$, $a_i \in \mathbb{Q}$ is the i-th partial derivative of I. Similarly, an affine equality is an equation E of the form

$$a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b = 0$$

We say that $\nu \in \mathbb{R}^{\mathcal{X}}_{\geqslant 0}$ satisfies I (resp. E), and write $\nu \models I$ (resp. $\nu \models E$), if $a_1 \cdot \nu(\mathsf{x}_1) + \cdots + a_n \cdot \nu(\mathsf{x}_n) + b \prec 0$ (resp. $= 0$) holds. In this case, $[\![I]\!]$ (resp. $[\![E]\!]$) refers to the set of valuations that satisfy I (resp. E). Equalities (resp. inequalities) are equivalent when they are satisfied by the same valuations. In particular, multiplying the additive constant b and all partial derivatives a_i by the same factor $N \in \mathbb{N} \setminus \{0\}$ gives an equivalent equality (resp. inequality), and we will therefore assume that they are always integers.

Definition 4.1. A *cell* is a set $c \subseteq \mathbb{R}^{\mathcal{X}}_{\geqslant 0}$, defined by a conjunction of affine inequalities $I_1 \wedge \cdots \wedge I_m$, such that $\nu \in c$ if and only if for all $1 \leqslant i \leqslant m$, $\nu \models I_i$. We write $c = [\![I_1 \wedge \cdots \wedge I_m]\!]$ in this case.

Cells are convex polyhedra, and the intersection of finitely many cells is a cell. From every affine inequality I, we can extract an affine equality $E(I)$, of identical partial derivatives and additive constant. Then, we call *borders* of a cell $c = [\![I_1 \wedge \cdots \wedge I_m]\!]$ the affine equalities $E(I_1), \ldots, E(I_m)$. The closure $\overline{c}$ of a cell c is obtained by replacing every comparison operator $<$ by $\leqslant$ in its affine inequalities. Note that regions and zones are particular cases of cells, where borders are of the form $\mathsf{x} + b = 0$ or $\mathsf{x} - \mathsf{x}' + b = 0$. An example of cell and its borders is given in Figure 4.1.

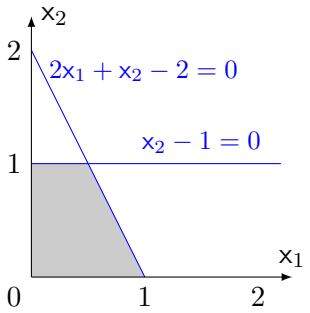

Figure 4.1 – The cell $2\mathsf{x}_1 + \mathsf{x}_2 - 2 < 0 \wedge \mathsf{x}_2 - 1 < 0$ in gray, and its borders in blue.

Let E be an affine equality of equation $a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b = 0$. We say that $\mathbb{R}^{\mathcal{X}}_{\geqslant 0}$ is partitioned by E into three cells:

— $c_<$, defined by $a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b < 0$;

— $c_>$, defined by $a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b > 0$, i.e. $-a_1\mathsf{x}_1 - \cdots - a_n\mathsf{x}_n - b < 0$;

— $c_=$, defined by $a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b = 0$, i.e. the conjunction of $a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b \leqslant 0$ and $-a_1\mathsf{x}_1 - \cdots - a_n\mathsf{x}_n - b \leqslant 0$.

Then, given a set $\mathcal{E} = \{E_1, \ldots, E_m\}$ of affine equalities, we denote $c^j_<$, $c^j_>$ and $c^j_=$ the three cells obtained from $E_j \in \mathcal{E}$. For every mapping $\phi : \mathcal{E} \to \{<, >, =\}$, we define c_ϕ as the cell $c^1_{\phi(E_1)} \cap \cdots \cap c^m_{\phi(E_m)}$. Every valuation of $\mathbb{R}^{\mathcal{X}}_{\geqslant 0}$ belongs to some c_ϕ, and if $\phi \neq \phi'$ then $c_\phi \cap c_{\phi'} = \emptyset$: hence, the set of mappings $\{<, >, =\}^{\mathcal{E}}$ provides a partition of $\mathbb{R}^{\mathcal{X}}_{\geqslant 0}$ into 3^m cells. We say that $\mathbb{R}^{\mathcal{X}}_{\geqslant 0}$ is partitioned by $\mathcal{E}$ into $m' \in \mathbb{N}$ cells if m' of those 3^m

cells are non-empty. In fact, m' is bounded by $\mathcal{O}((2m)^n)$ (see, e.g., [Mat02, Prop. 6.1.1]), and we denote $\mathsf{Splits}(m,n)$ this bound (polynomial in m and exponential in n) on the number of cells in the partition. Similarly, a cell $c \subseteq \mathbb{R}^{\mathcal{X}}_{\geqslant 0}$ is partitioned by $\mathcal{E}$ into at most $\mathsf{Splits}(m,n)$ sub-cells that have non-empty intersection with c. In particular, under the bounded clocks assumption we will partition $[0,M)^{\mathcal{X}}$ instead of $\mathbb{R}^{\mathcal{X}}_{\geqslant 0}$.

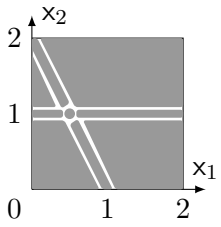

Figure 4.2 – A partition of $[0,2)^{\mathcal{X}}$ according to the two affine equalities of Figure 4.1.

Example 4.2. The 9 cells that partition $[0,2)^{\mathcal{X}}$ according to $\mathcal{E} = \{2\mathsf{x}_1 + \mathsf{x}_2 - 2 = 0, \mathsf{x}_2 - 1 = 0\}$ are represented in Figure 4.2.

Definition 4.3. A *partition* P is defined by a cell c_P and a set $\mathcal{E}_P$ of affine equalities, such that P encodes the set of cells that partition c_P according to $\mathcal{E}_P$, called *base cells*.

The cell c_P is called the domain of P. We denote $[\nu]_P$ the base cell that contains valuation $\nu \in c_P$.

Definition 4.4. A *partition value function* F defined over a partition P is a mapping from the base cells of P to affine value functions. It encodes a mapping from c_P to $\mathbb{R}_\infty$, denoted $[\![F]\!]$: if $\nu \in c_P$ and $F([\nu]_P)$ is defined by $\mathsf{y} = a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b$, then $[\![F]\!](\nu)$ equals $a_1 \cdot \nu(\mathsf{x}_1) + \cdots + a_n \cdot \nu(\mathsf{x}_n) + b$.

A partition value function F of domain c_P is *continuous* if for all $\nu \in c_P$, for every base cell c_b such that $\nu \in \overline{c_b}$, if $F(c_b)$ is defined by $\mathsf{y} = a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b$ then $[\![F]\!](\nu) = a_1 \cdot \nu(\mathsf{x}_1) + \cdots + a_n \cdot \nu(\mathsf{x}_n) + b$. In other words, the affine equations provided by F to neighbouring cells should match on the borders that separate them.

Finally, the *piecewise-affine value function* $\mathbf{V}_\ell : [0,M)^{\mathcal{X}} \to \mathbb{R}_\infty$ of a location ℓ is encoded as a pair (P,F) where P is a partition of domain $[0,M)^{\mathcal{X}}$, and F is a partition value function defined over P, such that $[\![F]\!] = \mathbf{V}_\ell$.

A piecewise-affine value function (P,F) is said *continuous on regions* if for every region $r \in \mathsf{Reg}(\mathcal{X},M)$, the restriction of F to domain r is continuous. There could be discontinuities in F, but only at borders separating different regions. In particular, if a partition value function is continuous over regions, and $[\![F]\!](\nu) = +\infty$ (resp. $-\infty$) for some ν, then for all ν' in the same region as ν, $[\![F]\!](\nu') = [\![F]\!](\nu)$.

Remark. In [ABM04], the domain of partitions is always a single region, and one value function is associated to each region. We define value functions over $[0,M)^{\mathcal{X}}$ instead, in order to obtain a symbolic algorithm, independent of regions. This induces slight differences in the way value functions are defined, because the mappings of [ABM04] are continuous everywhere while ours can have discontinuities at borders between regions. They define their partitions with overlaps over borders, such that $\mathbb{R}^{\mathcal{X}}_{\geqslant 0}$ is partitioned by an affine equality into two cells, $c_{\leqslant}$ and $c_{\geqslant}$, instead of the three $c_<$, $c_>$ and $c_=$. This changes the number of cells $\mathsf{Splits}(m,n)$, but not asymptotically.

Cells are bounded and convex polyhedra. Elementary operations over cells (emptiness, intersection and inclusion tests) can be seen as instances of linear programming, and can thus be performed in polynomial time. The main result of [ABM04], that we have formally reproved and extended to deal with negative weights, is to show that this data structure allows one to effectively compute the iterates $(\mathbf{V}^i)_{i\in\mathbb{N}}$. Contrary to the case without clocks, recalled before, this sequence does not stabilise in general. The algorithms we present in this chapter are thus based on reasons to either make the sequence stabilise (the divergent restriction we will first study) or stop its computation after a sufficient number i of turns to obtain a good approximation of the value (in the case of almost-divergent weighted timed games we will also introduce soon).

4.2 Divergent and almost-divergent weighted timed games

In this section, we introduce several classes of weighted timed games for which we state the results that this chapter shows. Let us recall first the class of weighted timed games studied in [Bou+04b], a large class with non-negative weights where the value problem is known to be decidable.

Definition 4.5. A weighted timed game $\mathcal{G}$ with non-negative weights satisfies the *strictly non-Zeno cost* property when every finite play ρ in $\mathcal{G}$ following a cycle in the region automaton $\mathcal{R}(\mathcal{G})$ satisfies $\mathsf{wt}(\rho) \geqslant 1$.

The intuition behind this class is that the weight of any long enough execution in $\mathcal{G}$ will ultimately grow above any fixed bound, and diverge towards $+\infty$ for an infinite execution. Therefore, the value of $\mathcal{G}$ is equal to the value $\mathsf{Val}^i_{\mathcal{G}}$ at some horizon i large enough, making the value problem decidable. It is shown in [Bou+04b] that i can be bounded exponentially in the size of $\mathcal{G}$.

We introduce divergent weighted timed games, as a natural generalisation of the strictly non-Zeno cost property to weights in $\mathbb{Z}$.

Definition 4.6. 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 $\mathsf{wt}(\rho) \notin (-1, 1)$.

If $\mathcal{G}$ has only non-negative weights on locations and edges, this definition matches with the strictly non-Zeno cost property, we will therefore refer to their class as the class of divergent weighted timed games with non-negative weights.

As in [Bou+04b], 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.

Our contributions on divergent weighted timed games summarise as follows:

Theorem 4.7. *The value problem over divergent weighted timed games is decidable in* 3-EXPTIME*, and is* EXPTIME*-hard. Moreover, deciding if a given weighted timed games is divergent is a* PSPACE*-complete problem.*

In [BJM15], the authors slightly extend the strictly non-Zeno cost property, to allow for cycles of weight exactly 0 while still preventing those of weight arbitrarily close to 0:

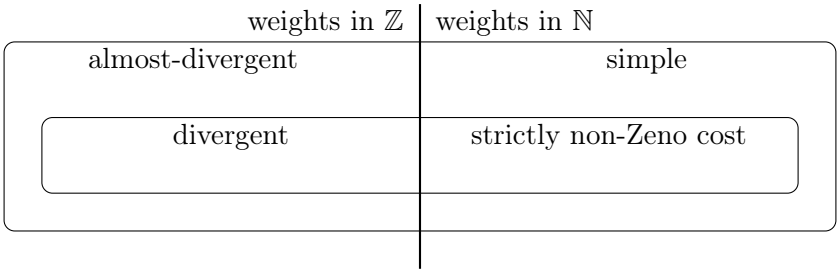

Figure 4.3 – Classes of weighted timed games, and their respective restrictions to non-negative weights.

Definition 4.8. A weighted timed games $\mathcal{G}$ with non-negative weights is called *simple* when every finite play ρ in $\mathcal{G}$ following a cycle in the region automaton $\mathcal{R}(\mathcal{G})$ satisfies $\mathsf{wt}(\rho) \in \{0\} \cup [1, +\infty)$.

Unfortunately, it is shown in [BJM15] that the value problem is undecidable, even for simple weighted timed games. They propose a solution to the value approximation problem, as a procedure computing an approximation of the value of every configuration. The intuition is that cycles of weight exactly 0 are only possible when every (non-negative) weight encountered along the cycle equals 0, allowing one to define a subgame where every cyclic execution has weight 0. They can then analyse this subgame separately, by applying a semi-unfolding procedure on $\mathcal{R}(\mathcal{G})$.

We introduce a class of weighted timed games that will extend this simplicity notion and allow negative weights, so that cyclic executions of weight exactly 0 are allowed, but not those close to 0. The first attempt would lead to the requirement that every finite play following a cycle in the region automaton $\mathcal{R}(\mathcal{G})$ has a weight in $(-\infty, -1] \cup \{0\} \cup [1, +\infty)$. However, we did not obtain positive results for the value approximation problem on this class of weighted timed games since cycles of weight exactly 0 do not have the good property presented above for simple weighted timed games. Instead, we require a stability by decomposition for cycles of weight 0.

If $\mathsf{p} = (\ell_0, r_0) \xrightarrow{r_1, e_0} (\ell_1, r_1) \xrightarrow{r_2, e_1} \cdots (\ell_{k-1}, r_{k-1}) \xrightarrow{r_0, e_{k-1}} (\ell_0, r_0)$ is a region cycle in $\mathcal{R}(\mathcal{G})$, it is either simple (i.e. for all i, j such that $0 \leqslant i < j < k$, $(\ell_i, r_i) \neq (\ell_j, r_j)$) or we can extract smaller cycles from it. Indeed, if p is not simple, there exists a pair (i, j) such that $0 \leqslant i < j < k$ and $(\ell_i, r_i) = (\ell_j, r_j)$. Then, for such a pair, we can write $\mathsf{p} = \mathsf{p}_1\mathsf{p}_2\mathsf{p}_3$ such that $|\mathsf{p}_1| = i$, $|\mathsf{p}_3| = k - j$. It follows that p_2 and $\mathsf{p}_3\mathsf{p}_1$ are both region cycles around (ℓ_i, r_i). This process is called a decomposition of p into smaller cycles $\mathsf{p}' = \mathsf{p}_3\mathsf{p}_1$ and $\mathsf{p}'' = \mathsf{p}_2$.

Definition 4.9. A weighted timed game $\mathcal{G}$ is *almost-divergent* if every play ρ following a cycle p of $\mathcal{R}(\mathcal{G})$ satisfies $\mathsf{wt}(\rho) \in (-\infty, -1] \cup \{0\} \cup [1, +\infty)$, and if $\mathsf{wt}(\rho) = 0$ then for every decomposition of p into smaller cycles p' and p'', and plays ρ' and ρ'' following p' and p'', respectively, it holds that $\mathsf{wt}(\rho') = \mathsf{wt}(\rho'') = 0$. [1]

Clearly, every divergent weighted timed game is almost-divergent. Moreover, as we will see in Proposition 4.19, when weights are non-negative, this class matches the simple

1. Once again, we could replace $-1, 1$ by $-\kappa, \kappa$ with $0 < \kappa < 1$ to define an equivalent notion of κ-almost-divergence.

weighted timed games of [BJM15] therefore inheriting their undecidability result. We will thus refer to simple weighted timed games as almost-divergent weighted timed games with non-negative weights. Figure 4.3 represents the hierarchy of the classes of games that we introduced.

Example 4.10. Consider the weighted timed game $\mathcal{G}$ in Figure 2.1 (page 37). The loop on ℓ_1 contains a cycle of $\mathcal{R}(\mathcal{G})$ around $(\ell_1, \mathbf{0})$ that jumps to the region $1 < \mathsf{x} < 2$. For every $d \in (1,2)$, there exists a play ρ following this cycle that uses delay d, so that $\mathsf{wt}(\rho) = 3 - 2d \in (-1,1)$. It follows that $\mathcal{G}$ is neither divergent nor almost-divergent. Changing the guard on this loop to $2 \leqslant \mathsf{x} < 3$ makes $\mathcal{G}$ divergent, as every region cycle left in $\mathcal{G}$ iterates the loops around $(\ell_1, \mathbf{0})$ and $(\ell_3, \mathbf{0})$ of cumulative weights in $(-3, -1]$ and $(-5, -1)$, respectively.

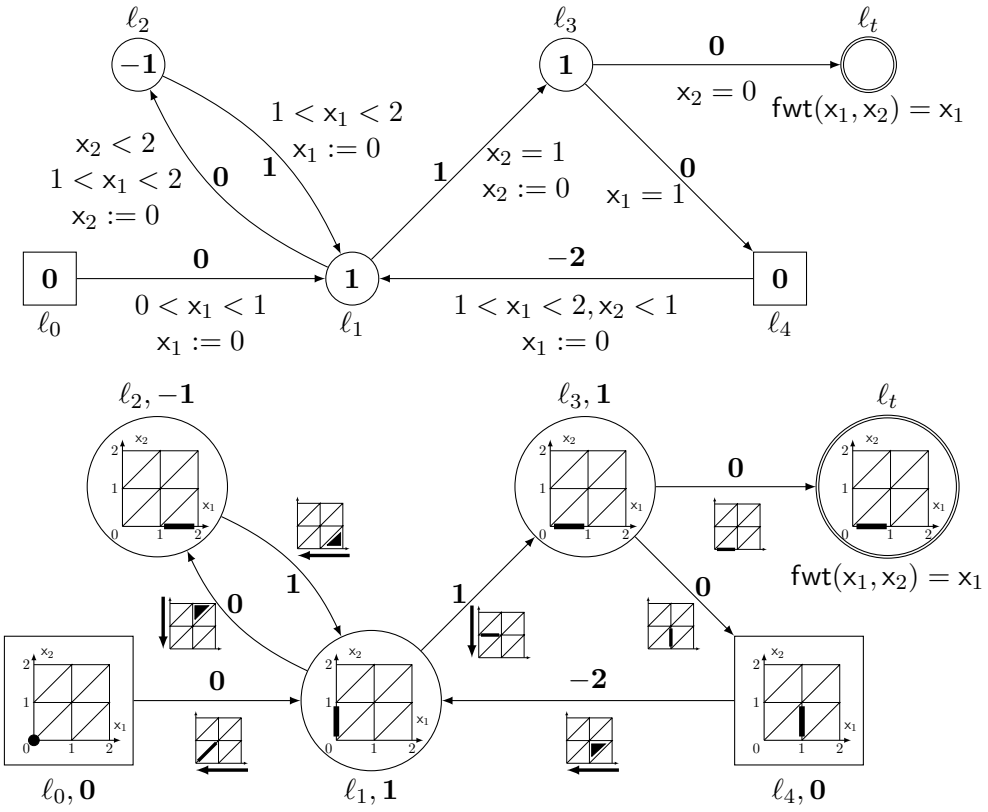

Figure 4.4 – A weighted timed game $\mathcal{G}$ with two clocks x_1 and x_2, and the portion of its region game $\mathcal{R}(\mathcal{G})$ accessible from configuration $(\ell_0, (0,0))$. The states of $\mathcal{R}(\mathcal{G})$ are labelled by their associated region, location and weight, and edges are labelled by a representation of their guards and resets. For example, the edge from (ℓ_0, r_0) to (ℓ_1, r_1) in $\mathcal{R}(\mathcal{G})$ highlights the time successors of the region r_0 that satisfy the guard $0 < \mathsf{x}_1 < 1$, and the arrow represents the direction in which this set of points is projected by the clock reset $\mathsf{x}_1 := 0$, so that we end up in the region r_1.

Example 4.11. Consider the weighted timed game $\mathcal{G}$ in Figure 4.4, and its region game $\mathcal{R}(\mathcal{G})$. We chose an example where $\mathcal{R}(\mathcal{G})$ is isomorphic to $\mathcal{G}$ for readability reasons. $\mathcal{R}(\mathcal{G})$

contains one SCC $\{\ell_1, \ell_2, \ell_3, \ell_4\}$, made of two simple cycles, $\mathsf{p}_1 = \ell_1 \to \ell_2 \to \ell_1$ and $\mathsf{p}_2 = \ell_1 \to \ell_3 \to \ell_4 \to \ell_1$, so that:

— all plays following p_1 have cumulative weight in the interval $(1, 3)$,

— and all plays following p_2 have cumulative weight 0.

As every cycle p of $\mathcal{R}(\mathcal{G})$ either iterates p_2 only, or contains p_1, it holds that a play ρ following p satisfies $\mathsf{wt}(\rho) \in \{0\} \cup [1, +\infty)$, and if $\mathsf{wt}(\rho) = 0$ then any decomposition of p into smaller cycles p' and p'' implies that they all follow iterates of p_2, so that plays along them must have cumulative weight 0. Therefore, $\mathcal{G}$ is almost-divergent. If one removes ℓ_4, $\mathcal{G}$ becomes divergent.

Our first result on almost-divergent weighted timed games is the following extension of the approximation procedure for non-negative weights:

Theorem 4.12. *Given an almost-divergent weighted timed game $\mathcal{G}$, a location ℓ and $\varepsilon \in \mathbb{Q}_{>0}$, we can compute an ε-approximation of $\mathsf{Val}_{\mathcal{G}}(\ell, \mathbf{0})$ in time triply-exponential in the size of $\mathcal{G}$ and polynomial in $1/\varepsilon$. Moreover, deciding if a weighted timed game is almost-divergent is a* PSPACE*-complete problem.*

To obtain these results on divergent and almost-divergent weighted timed games, we follow a computation schema that we now outline. First, we will always reason on the region game $\mathcal{R}(\mathcal{G})$. The goal is to compute or approximate $\mathsf{Val}_{\mathcal{R}(\mathcal{G})}((\ell_0, [\mathbf{0}]), \mathbf{0})$ for some initial location ℓ_0. Techniques of [ABM04], based on the partition value functions we have recalled previously, allow one to compute the (exact) values of a tree-shaped weighted timed game. The idea is thus to decompose as much as possible the game $\mathcal{R}(\mathcal{G})$ as a tree. First, we decompose the region game into strongly connected components (SCCs, left of Figure 4.5): we must think about the final weight functions as the previously computed approximations of the values of SCCs coming after the current one in the topological order. We will keep as an invariant that final weight functions are piecewise affine with a finite number of pieces, and are continuous on each region.

For an SCC of $\mathcal{R}(\mathcal{G})$ and an initial state $(\ell_0, [\mathbf{0}])$ of $\mathcal{R}(\mathcal{G})$ provided by the SCC decomposition, we show that the game on the SCC is equivalent to a game on a tree built from a semi-unfolding (see middle of Figure 4.5) of $\mathcal{R}(\mathcal{G})$ from $(\ell_0, [\mathbf{0}])$ of finite depth, with certain nodes of the tree being *kernels* (parts of $\mathcal{R}(\mathcal{G})$ that contain all cycles of weight 0). The semi-unfolding is stopped either when reaching a final location, or when some location (or kernel) has been visited for a certain fixed number of times. Notice that, for divergent weighted timed games, there are no kernels, which simplifies the computation.

Then, we compute or approximate $\mathsf{Val}_{\mathcal{G}}(\ell_0, \mathbf{0})$ with a bottom-up computation on the semi-unfolding. This computation is exact on nodes labelled by a single region state s, but approximated on kernel nodes K_s, if any. For the latter, we use the already studied corner-point abstraction (right of Figure 4.5) over $1/N$-regions to compute values, and prove that, with an appropriately chosen N, this provides an ε-approximation of values.

This resolution of the value problem for divergent weighted timed games and approximation problem for almost-divergent weighted timed games heavily relies on the region abstraction, and requires one to construct $\mathcal{R}(\mathcal{G})$ entirely and compute its SCCs, before unfolding it partially in a tree-shaped structure. Our second result is a more symbolic approximation schema based on the value iteration algorithm only: the computations are not performed on the region abstraction, but instead use the partition value functions introduced before, that can cover several regions.

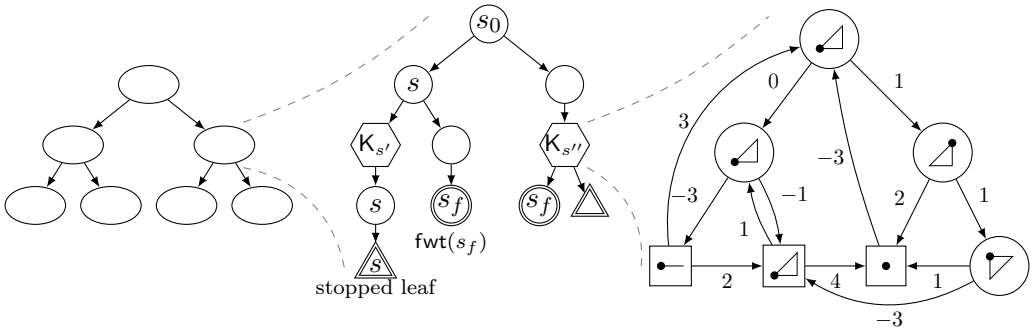

Figure 4.5 – Static approximation schema: SCC decomposition of $\mathcal{R}(\mathcal{G})$, semi-unfolding of an SCC, corner-point abstraction for the kernels

Theorem 4.13. *Let $\mathcal{G}$ be an almost-divergent weighted timed game such that $\mathsf{Val}_{\mathcal{G}}(\ell,\nu) > -\infty$ for every configuration (ℓ,ν). Then the sequence $(\mathsf{Val}^k_{\mathcal{G}})_{k\geqslant 0}$ converges towards $\mathsf{Val}_{\mathcal{G}}$ and for every $\varepsilon \in \mathbb{Q}_{>0}$, we can compute an integer P such that $\mathsf{Val}^P_{\mathcal{G}}$ is an ε-approximation of $\mathsf{Val}_{\mathcal{G}}$ for all configurations.*

Note that we have to control for configurations (ℓ,ν) of value $-\infty$, where the non-increasing sequence $(\mathsf{Val}^k_{\mathcal{G}}(\ell,\nu))_{k\in\mathbb{N}}$ (that starts at $+\infty$) will diverge towards $-\infty$, but has no hope of approximating it. However, we will show that the configurations with value $-\infty$ can be computed pre-emptively:

Proposition 4.14. *For almost-divergent weighted timed games, the $-\infty$-value problem is* EXPTIME*-complete.*

The exponential upper bound is obtained in Section 4.4. This contrasts with the general case (not necessarily almost-divergent), where the $-\infty$-value problem is undecidable:

Proposition 4.15. *Given a weighted timed game $\mathcal{G}$ and an initial location ℓ_0, the decision problem asking whether $\mathsf{Val}_{\mathcal{G}}(\ell_0,\mathbf{0}) = -\infty$ is undecidable.*

The rest of this chapter aims at giving hints of the proofs of these various results.

4.3 Deciding divergence and almost-divergence

In this section, we will study properties that region cycles must satisfy in divergent or almost-divergent weighted timed games. This will give us a better understanding of the modelling power these classes confer, as well as enable us to provide procedures of optimal complexity to decide if a weighted timed game fulfils the divergence or almost-divergence conditions.

Let us start with properties that hold for all almost-divergent weighted timed games $\mathcal{G}$. A region cycle p of $\mathcal{R}(\mathcal{G})$ is said to be a positive cycle (resp. a negative cycle, a 0-cycle) if every finite play ρ following p satisfies $\mathsf{wt}(\rho) \geqslant 1$ (resp. $\mathsf{wt}(\rho) \leqslant -1$, $\mathsf{wt}(\rho) = 0$).

We start by showing that, in an almost-divergent game, all cycles $\mathsf{p} = t_1 \cdots t_n$ of $\mathcal{R}(\mathcal{G})$ (with $t_1,\dots,t_n$ edges of $\mathcal{R}(\mathcal{G})$) are either 0-cycles, positive cycles or negative cycles [2], and we can classify a cycle by looking only at one of the corner plays following it:

2. In contrast, Definition 4.9 only requires that each play following a region cycle has weight in $(-\infty,-1]\cup\{0\}\cup[1,+\infty)$, without disallowing a region cycle to contain plays of different types.

Lemma 4.16. *Let $\mathcal{G}$ be an almost-divergent weighted timed game. A region cycle p is a positive cycle (resp. a negative cycle, a 0-cycle) if and only if there exists a corner play $\overline{\rho}$ following p with $\mathsf{wt}(\overline{\rho}) > 0$ (resp. $\mathsf{wt}(\overline{\rho}) < 0$, $\mathsf{wt}(\overline{\rho}) = 0$). Moreover, every region cycle in $\mathcal{G}$ is either positive, negative, or a 0-cycle.*

An important result is that the sign of cycles is stable by rotation. This is not trivial because plays following a region cycle can start and end in different valuations, therefore changing the starting region state of the cycle could *a priori* change the plays that follow it and the sign of their weights.

Lemma 4.17. *Let p and p' be region paths of an almost-divergent weighted timed game. If pp' is a positive cycle (resp. a negative cycle, a 0-cycle), then $\mathsf{p}'\mathsf{p}$ is a positive cycle (resp. a negative cycle, a 0-cycle).*

Therefore, region cycles in almost-divergent games behave well: we can compose and rotate them while preserving their sign in the expected way.

We now focus on strongly connected components (SCCs) of the region abstraction $\mathcal{R}(\mathcal{G})$. An SCC S of $\mathcal{R}(\mathcal{G})$ is said to be positive (resp. negative) if every cycle in S is positive (resp. negative), i.e. if every play ρ following a region cycle in S satisfies $\mathsf{wt}(\rho) \geqslant 1$ (resp. $\mathsf{wt}(\rho) \leqslant -1$).

Proposition 4.18. *A weighted timed game $\mathcal{G}$ is divergent if and only if, each SCC of $\mathcal{R}(\mathcal{G})$ is either positive or negative.*

Likewise, an SCC S of $\mathcal{R}(\mathcal{G})$ is said to be non-negative (resp. non-positive) if every region cycle in S is either a positive cycle or a 0-cycle (resp. either a negative cycle or a 0-cycle), i.e. every play ρ following a region cycle in S satisfies either $\mathsf{wt}(\rho) \geqslant 1$ or $\mathsf{wt}(\rho) = 0$ (resp. either $\mathsf{wt}(\rho) \leqslant -1$ or $\mathsf{wt}(\rho) = 0$). We obtain:

Proposition 4.19. *A weighted timed game $\mathcal{G}$ is almost-divergent if and only if each SCC of $\mathcal{R}(\mathcal{G})$ is either non-negative or non-positive.*

We now prove these two results, which makes it even clearer how the (almost-)divergence can be useful. First, note that if $\mathcal{G}$ is divergent it has no 0-cycle, and Proposition 4.19 implies that each SCC of $\mathcal{R}(\mathcal{G})$ is either positive or negative. Conversely, if each SCC of $\mathcal{R}(\mathcal{G})$ is either positive or negative, Proposition 4.19 implies that $\mathcal{G}$ is divergent. Therefore, Proposition 4.18 is a corollary of Proposition 4.19. The rest of this section now proves Proposition 4.19.

To prove the reciprocal implication of Proposition 4.19, we only need to show that non-negative (resp. non-positive) SCCs of $\mathcal{R}(\mathcal{G})$ satisfy the almost-divergent condition. By definition, they only contain plays ρ following region cycles p such that $\mathsf{wt}(\rho) \in \{0\} \cup [1, +\infty)$ (resp. $\mathsf{wt}(\rho) \in (-\infty, -1] \cup \{0\}$). Then, assume $\mathsf{wt}(\rho) = 0$ and that p can be decomposed into smaller cycles p' and p''. Definition 4.9 requires us to show that all plays ρ' and ρ'' following p' and p'', respectively, are such that $\mathsf{wt}(\rho') = \mathsf{wt}(\rho'') = 0$, i.e. p' and p'' are 0-cycles. Note that by Lemma 4.17, $\mathsf{p}'\mathsf{p}''$ is a 0-cycle. As p' and p'' are contained in the same SCC, they are either both non-negative cycles or both non-positive cycles. Let $\rho'\rho''$ be a play following $\mathsf{p}'\mathsf{p}''$, so that ρ' follows p' and ρ'' follows p''. Then, $\mathsf{wt}(\rho'\rho'') = \mathsf{wt}(\rho') + \mathsf{wt}(\rho'') = 0$, it follows that $\mathsf{wt}(\rho') = \mathsf{wt}(\rho'') = 0$, i.e. p' and p'' are 0-cycles.

For the direct implication, the situation is more complex: we need to be careful while composing cycles with each other. A very helpful tool in this context is the folded orbit graphs, introduced on page 45. Suppose that $\mathcal{G}$ is almost-divergent, and consider two cycles p and p' in the same SCC of $\mathcal{R}(\mathcal{G})$. We need to show that they are both either non-positive or non-negative.

Consider first the case where they share a region state (ℓ, r), and suppose by contradiction that p is negative and p' is positive. We assume that (ℓ, r) is the first region state of both p and p', possibly performing a rotation of the cycles if necessary (in particular this preserves their sign by Lemma 4.17). We construct a graph $\mathsf{FOG}(\mathsf{p}, \mathsf{p}')$ as the union of $\mathsf{FOG}(\mathsf{p})$ and $\mathsf{FOG}(\mathsf{p}')$ (that share the same set of vertices), colouring in blue the edges of $\mathsf{FOG}(\mathsf{p})$ and in red the edges of $\mathsf{FOG}(\mathsf{p}')$. A path in $\mathsf{FOG}(\mathsf{p}, \mathsf{p}')$ is said blue (resp. red) when all of its edges are blue (resp. red).

Since $\mathsf{FOG}(\mathsf{p})$ and $\mathsf{FOG}(\mathsf{p}')$ are finite graphs with no deadlocks, from every corner of $\mathsf{FOG}(\mathsf{p}, \mathsf{p}')$, 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 $\mathsf{FOG}(\mathsf{p}, \mathsf{p}')$, there exists a blue cycle C and a red cycle C' that can reach each other in $\mathsf{FOG}(\mathsf{p}, \mathsf{p}')$. Denote by v and v' the first corners of cycles C and C', respectively. Now, everything is finite, and weights of plays in-between corners of the folded orbit graphs are integer, so they can be recombined to get a 0-cycle.

To finish the proof of the direct implication of Proposition 4.19, we need to consider the case where the two cycles p and p' do not share any region states. By strong connectivity, in $\mathcal{R}(\mathcal{G})$, there exists a path p_1 from the first state of p to the first state of p', and a path p_2 from the first state of p' to the first state of p. Consider the cycle p'' of $\mathcal{R}(\mathcal{G})$ defined by $\mathsf{p}\mathsf{p}_1\mathsf{p}'\mathsf{p}_2$. By the almost-divergence of $\mathcal{G}$, p'' must be either positive, negative or a 0-cycle. Since it shares a state with both p and p', the previous case allows us to prove a contradiction in both positive and negative cases, and therefore p'' must be a 0-cycle. This contradicts the hypothesis as one of the decompositions of p'' into smaller cycles produces p and $\mathsf{p}_1\mathsf{p}'\mathsf{p}_2$, with p a non-0-cycle. This concludes the proof of Proposition 4.19.

Remark. These characterisations of divergent or almost-divergent WTGs in term of SCCs provide an intuitive understanding of the modelling power these classes hold. For divergence, the model should have a global structure (the SCC decomposition) linking modules in an acyclic fashion. For each module, we have to choose between a positive dynamic, where weights always eventually increase, and a negative dynamic, where weights always eventually decrease. For almost-divergence, the modules may also have portions that are (eventually) neutral with regard to weight accumulation. In both classes, arbitrarily small weights should not be allowed to accumulate.

The previous characterisation can be used to study the *membership problem* for divergent (resp. almost-divergent) weighted timed games, i.e. the decision problem that asks if a given weighted timed game is divergent (resp. almost-divergent). As mentioned in Theorems 4.7 and 4.12, we show that it is PSPACE-complete for both of these classes.

Relying on the previous characterisation of Propositions 4.18 and 4.19, the algorithms will consist in only considering region cycles of length bounded by the number of corners in the corner-point abstraction $\Gamma(\mathcal{G})$. For divergent weighted timed game, this will be correct by using the following result:

Lemma 4.20. *Let $\mathcal{G}$ be a weighted timed game. An SCC S of $\mathcal{R}(\mathcal{G})$ is positive (resp. negative) if and only if every region cycle in S, of length at most $|\Gamma(\mathcal{G})|$, is positive (resp. negative).*

Then, to decide if a game is *not divergent*, using Proposition 4.18, it suffices to search for an SCC of the region automaton containing a cycle such that there exists a corner play following it of non-negative weight, and a cycle such that there exists a corner play following it of non-positive weight, both of length bounded by $B = |\Gamma(\mathcal{G})| \leqslant |L| \times |\mathsf{Reg}(\mathcal{X}, M)| \times (|\mathcal{X}| + 1)$. We can test this condition in NPSPACE: we guess a starting region for each cycle, use standard reachability analysis [AD94] to check that they are in the same SCC of $\mathcal{R}(\mathcal{G})$ (in PSPACE), and use the following result with comparison $\geqslant 0$ and $\leqslant 0$, respectively, to check the sign of each cycle.

Lemma 4.21. *Consider a weighted timed game $\mathcal{G}$, a region state (ℓ, r) of $\mathcal{R}(\mathcal{G})$, a bound $B \in \mathbb{N}$, and a comparison operator $\bowtie \in \{<, >, \leqslant, \geqslant, =, \neq\}$. Deciding if there exists a corner play $\overline{\rho}$ following a cycle p of $\mathcal{R}(\mathcal{G})$ starting from (ℓ, r), such that $|\mathsf{p}| \leqslant B$ and $\mathsf{wt}(\overline{\rho}) \bowtie 0$, is in PSPACE (i.e. it can be done using space polynomial in $|\mathcal{G}|$ and $\log(B)$).*

Since the bound B is at most exponential in $|\mathcal{G}|$, this check can be performed in PSPACE. This shows that the membership problem for divergent weighted timed games is in coNPSPACE = coPSPACE = PSPACE by the theorems of Immerman-Szelepcsényi [Imm88; Sze88] and Savitch [Sav70]. The problem can also be shown PSPACE-hard (indeed the coPSPACE, which is identical) by a reduction from the reachability problem in a timed automaton.

For almost-divergent weighted timed games, the length of the required region cycles is bigger, because of the possible presence of 0-cycles.

Lemma 4.22. *Let $\mathcal{G}$ be a weighted timed game. An SCC S of $\mathcal{R}(\mathcal{G})$ is non-negative (resp. non-positive) if and only if every region cycle in S, of length at most $|\Gamma(\mathcal{G})|^2$, is either a positive cycle or a 0-cycle (resp. either a negative cycle or a 0-cycle).*

Then, to decide if a game is *not almost-divergent*, we distinguish two cases:

— There exists a region cycle, of length at most $B = |\Gamma(\mathcal{G})|^2$, and two corner plays $\overline{\rho}$ and $\overline{\rho}'$, both following p, such that $\mathsf{wt}(\overline{\rho}) = 0$ and $\mathsf{wt}(\overline{\rho}') \neq 0$.

— An SCC of the region automaton contains a cycle such that there exists a corner play following it of negative weight, and a cycle such that there exists a corner play following it of positive weight, both of length bounded by $B = |\Gamma(\mathcal{G})|^2$.

We can test both conditions in NPSPACE, by guessing the starting regions of these cycles and using respectively Lemma 4.21 (for the second condition) and the following result (for the first condition):

Lemma 4.23. *Consider a weighted timed game $\mathcal{G}$, a region state (ℓ, r) of $\mathcal{R}(\mathcal{G})$, a bound $B \in \mathbb{N}$, and comparison operators $\bowtie, \bowtie' \in \{<, >, \leqslant, \geqslant, =, \neq\}$. Deciding if there exists a cycle p of $\mathcal{R}(\mathcal{G})$ starting from (ℓ, r), and two corner plays $\overline{\rho}$ and $\overline{\rho}'$, both following p, such that $|\mathsf{p}| \leqslant B$, $\mathsf{wt}(\overline{\rho}) \bowtie 0$ and $\mathsf{wt}(\overline{\rho}') \bowtie' 0$, is in PSPACE.*

This shows that the membership problem for divergent and almost-divergent weighted timed games is in coNPSPACE = coPSPACE = PSPACE [Imm88; Sze88; Sav70].

4.4 Deciding infinite values

There are three reasons for an infinite value to appear in a weighted timed game:

— an infinite value ($+\infty$ or $-\infty$) could appear in a final weight function and be propagated along a play;

— a value $+\infty$ could be obtained if Min is not able to reach a target location;

— a value $-\infty$ could be obtained if Min is able to reach a negative cycle, loop arbitrarily many times inside, and then reach a target location.

This section explains how to detect the three situations in the region game $\mathcal{R}(\mathcal{G})$ of an almost-divergent weighted timed game $\mathcal{G}$. Before that, let us start by formalising a way to safely remove region states of $\mathcal{R}(\mathcal{G})$ for which the value of all their associated configurations is known to be infinite. Let $S^{+\infty}$ be a subset of $S = L \times \mathsf{Reg}(\mathcal{X}, M)$, such that $\mathsf{Val}_{\mathcal{R}(\mathcal{G})}((\ell, r), \nu) = +\infty$ for all $(\ell, r) \in S^{+\infty}$ and $\nu \in r$. If a configuration of $\mathcal{G}$ is in the attractor of Max towards $S^{+\infty}$, then Max has a strategy giving it value $+\infty$. Moreover, attractor being computable over regions all the configurations in the same region have a value $+\infty$: therefore we could add this region to $S^{+\infty}$. We will thus assume that $S^{+\infty}$ is closed by attractor for Max. We can define the same notion for a set $S^{-\infty}$ of states with value $-\infty$, that can be assumed closed by attractor of Min.

Lemma 4.24. *In $\mathcal{R}(\mathcal{G})$, let $S^{+\infty}$ be a set of region states of value $+\infty$ closed by attractor of* Max*, and let $S^{-\infty}$ be a set of region states of value $-\infty$ closed by attractor of* Min*. Removing the region states $S^{+\infty} \cup S^{-\infty}$ from $\mathcal{R}(\mathcal{G})$ will not change any other value.*

Infinite final values

As a first step, we explain how to compute and remove all region states with value $+\infty$, and region states with value $-\infty$ because of final weights. The only states with infinite value that will remain are some states that derive a value of $-\infty$ from arbitrary accumulation of negative weight, and we will deal with them later.

Recall that the final weight function fwt has been supposed piecewise affine with a finite number of pieces and is continuous on each region. In particular, final weights $+\infty$ or $-\infty$ are given to entire regions. Then, let $S_{\mathsf{t}}^{-\infty} \subseteq L_{\mathsf{t}} \times \mathsf{Reg}(\mathcal{X}, M)$ (resp. $S_{\mathsf{t}}^{+\infty}$) denote the set of target region states that fwt maps to $-\infty$ (resp. $+\infty$).

Proposition 4.25. *If a region state of $\mathcal{G}$ is in the attractor of* Min *towards $S_{\mathsf{t}}^{-\infty}$ (resp. in the attractor of* Max *towards $S_{\mathsf{t}}^{+\infty}$), then it has value $-\infty$ (resp. $+\infty$). Moreover, if we remove those states from $\mathcal{R}(\mathcal{G})$, the value of the other configurations does not change.*

Then, assuming that all final weights are finite, configurations with value $+\infty$ are those from which Min cannot reach the target region states: thus, they can also be computed and removed using the attractor algorithm.

Proposition 4.26. *If* fwt *maps all configurations to values in $\mathbb{R}$, then a configuration $((\ell, r), \nu)$ has value $+\infty$ if and only if (ℓ, r) is not in the attractor of* Min *towards region states $L_{\mathsf{t}} \times \mathsf{Reg}(\mathcal{X}, M)$. Moreover, if we remove those region states from $\mathcal{R}(\mathcal{G})$, the value of the other configuration does not change.*

We can now assume that all configurations have value in $\mathbb{R} \cup \{-\infty\}$ and that all target region states have final weight in $\mathbb{R}$: the precomputation needed so far consists only of attractor computations, that can be performed in complexity linear in $|\mathcal{R}(\mathcal{G})|$.

As previously explained, finding all states of value $-\infty$ is harder (in particular undecidable in general), but whenever we do manage to find them we can safely remove them by Lemma 4.24. The solution is quite simple for divergent weighted timed games, and slightly more involved in almost-divergent weighted timed games, because of the apparition of 0-cycles. In order to only present one uniform solution, we directly deal with the more general almost-divergent case. To do so, we introduce a new tool, the *kernels*, that will also be very useful in the rest of the study.

Kernel of an almost-divergent weighted timed game

Kernels are a way to group all 0-cycles of a weighted timed game. We study those kernels and give a characterisation allowing computability. In [BJM15], kernels have also been studied, and contain exactly all transitions and locations of weight 0. Contrary to their non-negative case, the situation is more complex in our case with arbitrary weights since 0-cycles could go through locations or transitions that have a weight different from 0. Moreover, it is not trivial (and may not be true in a non almost-divergent weighted timed game) to know whether it is sufficient to consider only simple 0-cycles, i.e. cycles without repetitions.

We will now construct the kernel K as the subgraph of $\mathcal{R}(\mathcal{G})$ containing all 0-cycles. Formally, let T_{K} be the set of edges of $\mathcal{R}(\mathcal{G})$ belonging to a *simple* 0-cycle, and S_{K} be the set of states covered by T_{K}. We define the kernel K of $\mathcal{R}(\mathcal{G})$ as the subgraph of $\mathcal{R}(\mathcal{G})$ defined by S_{K} and T_{K}. Edges in $T \backslash T_{\mathsf{K}}$ with starting state in S_{K} are called the output edges of K. We define it using only simple 0-cycles in order to ensure its computability. However, this is of no harm, since the kernel contains exactly all the 0-cycles, which will be crucial in our approximation schema.

Proposition 4.27. *A cycle of $\mathcal{R}(\mathcal{G})$ is entirely in K if and only if it is a 0-cycle.*

Values $-\infty$ coming from negative cycles

Equipped with the kernels, we are now ready to remove the only remaining configurations having a value $-\infty$ in $\mathcal{R}(\mathcal{G})$. Indeed, if the SCC is non-positive, let T_{t} be the set of edges of $\mathcal{R}(\mathcal{G})$ whose end state has location in L_{t}. We prove that a configuration has value $-\infty$ if and only if it belongs to a region state where player Min can ensure the LTL formula on edges $\phi = (\mathrm{G}\,\neg T_{\mathsf{t}}) \wedge \neg \mathrm{FG}\, T_{\mathsf{K}}$, thus giving us:

Proposition 4.28. *In an SCC of $\mathcal{R}(\mathcal{G})$, the set of configurations with value $-\infty$ is a union of regions computable in time linear in the size of $\mathcal{R}(\mathcal{G})$. Moreover, if we remove those states from $\mathcal{R}(\mathcal{G})$, the value of the other configurations does not change.*

From this point on, **we assume that no configurations of $\mathcal{R}(\mathcal{G})$ have value $+\infty$ or $-\infty$, and that the final weight function maps all configurations to $\mathbb{R}$.** Since fwt is piecewise affine with finitely many pieces, fwt is bounded. Let $w^{\mathsf{t}}_{\max}$ denote the supremum of $|\mathsf{fwt}|$, ranging over all target configurations.

4.5 Semi-unfolding of weighted timed games

Given an almost-divergent weighted timed game $\mathcal{G}$, we describe the construction of its *semi-unfolding* $\mathcal{T}(\mathcal{G})$ (as depicted in Figure 4.5). This crucially relies on the absence of

states with value $-\infty$ which has been explained in Section 4.4.

We only build the semi-unfolding $\mathcal{T}(\mathcal{G})$ of an SCC of $\mathcal{G}$ starting from some state $(\ell_0, r_0) \in S$ of the region game, since it is then easy to glue all the semi-unfoldings together to get the one of the full game. **We thus assume in this section that $\mathcal{R}(\mathcal{G})$ is an SCC.** Since every configuration has finite value, we will prove that values of the game are bounded by $|\mathcal{R}(\mathcal{G})|w_{\max} + w^{\mathsf{t}}_{\max}$. As a consequence, we can find a bound γ linear in $|\mathcal{R}(\mathcal{G})|$, $w_{\max}$ and $w^{\mathsf{t}}_{\max}$ such that a play that visits some state outside the kernel more than γ times has weight strictly above $|\mathcal{R}(\mathcal{G})|w_{\max} + w^{\mathsf{t}}_{\max}$, hence is useless for the value computation. This leads to considering the semi-unfolding $\mathcal{T}(\mathcal{G})$ of $\mathcal{R}(\mathcal{G})$ (nodes in the kernel are not unfolded, see Figure 4.5) such that each node not in the kernel is encountered at most γ times along a branch: the end of each branch is called a *stopped leaf* of the semi-unfolding. In particular, the depth of $\mathcal{T}(\mathcal{G})$ is bounded by $|\mathcal{R}(\mathcal{G})|\gamma$, and thus is polynomial in $|\mathcal{R}(\mathcal{G})|$, $w_{\max}$ and $w^{\mathsf{t}}_{\max}$. Leaves of the semi-unfolding are thus of two types: target leaves that are copies of target locations of $\mathcal{R}(\mathcal{G})$ for which we set the target weight as in $\mathcal{R}(\mathcal{G})$, and stop leaves for which we set their target weight as being constant to $+\infty$ if the SCC $\mathcal{R}(\mathcal{G})$ is non-negative, and $-\infty$ if the SCC is non-positive.

More formally, if (ℓ, r) is in K, we let $\mathsf{K}_{\ell,r}$ be the part of K accessible from (ℓ, r) (note that $\mathsf{K}_{\ell,r}$ is an SCC as K is a disjoint set of SCCs). We define the output edges of $\mathsf{K}_{\ell,r}$ as being the output edges of K accessible from (ℓ, r). If (ℓ, r) is not in K, the output edges of (ℓ, r) are the edges of $\mathcal{R}(\mathcal{G})$ starting in (ℓ, r).

We define a tree T whose nodes are either labelled by region states $(\ell, r) \in S \backslash S_{\mathsf{K}}$ or by kernels $\mathsf{K}_{\ell,r}$, and whose edges are labelled by output edges in $\mathcal{R}(\mathcal{G})$. The root of the tree T is labelled with an initial region state (ℓ_0, r_0), or K_{ℓ_0,r_0} (if (ℓ_0, r_0) belongs to the kernel), and the successors of a node of T are then recursively defined by its output edges. When a state (ℓ, r) is reached by an output edge, the child is labelled by $\mathsf{K}_{\ell,r}$ if $(\ell, r) \in \mathsf{K}$, otherwise it is labelled by (ℓ, r). Edges in T are labelled by the edges used to create them. Along every branch, we stop the construction when either a final state is reached (i.e. a state not inside the current SCC) or the branch contains $3|\mathcal{R}(\mathcal{G})|w_{\max} + 2w^{\mathsf{t}}_{\max} + 2$ nodes labelled by the same state ((ℓ, r) or $\mathsf{K}_{\ell,r}$). Leaves of T with a location belonging to L_{t} are called *target leaves*, others are called *stopped leaves*.

We now transform T into a weighted timed game $\mathcal{T}(\mathcal{G})$, by replacing every node labelled by a state (ℓ, r) by a different copy $(\tilde{\ell}, r)$ of (ℓ, r). Those states are said to inherit from (ℓ, r). Edges of T are replaced by the edges labelling them, and have a similar notion of inheritance. Every non-leaf node labelled by a kernel $\mathsf{K}_{\ell,r}$ is replaced by a copy of the weighted timed game $\mathsf{K}_{\ell,r}$, output edges being plugged in the expected way. We deal with stopped leaves labelled by a kernel $\mathsf{K}_{\ell,r}$ by replacing them with a single node copy of (ℓ, r), like we dealt with node labelled by a state (ℓ, r). State partition between players and weights are inherited from the copied states of $\mathcal{R}(\mathcal{G})$. The only initial state of $\mathcal{T}(\mathcal{G})$ is the state denoted by $(\tilde{\ell}_0, r_0)$ inherited from (ℓ_0, r_0) in the root of T (either (ℓ_0, r_0) or K_{ℓ_0,r_0}). The final states of $\mathcal{T}(\mathcal{G})$ are the states derived from leaves of T. If $\mathcal{R}(\mathcal{G})$ is a non-negative (resp. non-positive) SCC, the final weight function fwt is inherited from $\mathcal{R}(\mathcal{G})$ on target leaves and set to $+\infty$ (resp. $-\infty$) on stopped leaves.

Proposition 4.29. *Let $\mathcal{G}$ be an almost-divergent weighted timed game, and let (ℓ_0, r_0) be some region state of $\mathcal{R}(\mathcal{G})$. The semi-unfolding $\mathcal{T}(\mathcal{G})$ with initial state $(\tilde{\ell}_0, r_0)$ (a copy of a region state (ℓ_0, r_0)) is equivalent to $\mathcal{G}$, i.e. for all $\nu_0 \in r_0$, $\mathsf{Val}_{\mathcal{G}}(\ell_0, \nu_0) = \mathsf{Val}_{\mathcal{T}(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0)$.*

Note that the semi-unfolding procedure of an SCC depends on $w^{\mathsf{t}}_{\max}$, where fwt can be the value function of an SCCs under the current one. Assuming all configurations have finite value, we can bound all values in the full game by $|\mathcal{R}(\mathcal{G})|w_{\max} + w^{\mathsf{t}}_{\max}$, which lets us bound uniformly the unfolding depth of each SCC and gives us a bound on the depth of the complete semi-unfolding tree:

$$|\mathcal{R}(\mathcal{G})|(3|\mathcal{R}(\mathcal{G})|w_{\max} + 2w^{\mathsf{t}}_{\max} + 2) + 1 \tag{4.2}$$

4.6 Computing values

In this section we conclude the proofs of Theorems 4.7 and 4.12, with a triply-exponential upper bound on computing (resp. approximating) values in divergent (resp. almost-divergent) weighted timed games.

These upper bounds are obtained by computations performed on the semi-unfolding $\mathcal{T}(\mathcal{G})$ described in Section 4.5. Whereas the computation is direct for divergent weighted timed games, we then extend it to almost-divergent weighted timed games by first explaining how to compute value approximations in kernels.

Acyclic and divergent WTGs

We first focus on the class of acyclic weighted timed games where the value problem is decidable, as shown by [ABM04]. Notice that, with respect to the result obtained in [ABM04], our setting is more general, in the sense that we allow for negative weights and for final weights, where they do not do so explicitly. Moreover, their result is stated for concurrent games, a generalisation of the turn-based games we consider. This leads to simplifications in the proofs, and makes easier some parts of the complexity analysis (that we skip in this manuscript). We need, in Section 4.6, to bound the partial derivatives of the functions we compute, which cannot be deduced from their result directly. For reasons detailed in the PhD thesis of Damien Busatto-Gaston [Bus19], we are not able to replicate their (incomplete) complexity analysis. We therefore rely on a doubly-exponential upper bound instead of the exponential one originally claimed in [ABM04]. Last but not least, this detailed analysis allows us to solve the synthesis of ε-optimal strategies for both players, as will be detailed in Section 4.7.

Theorem 4.30. *Given $i \geqslant 0$, computing $\mathsf{Val}^i_{\mathcal{G}}$ can be done in time doubly-exponential in i and exponential in the size of $\mathcal{G}$.*

The intuition behind the result is the observation that the mappings $\mathbf{V}^0_\ell$ are piecewise affine for all ℓ, and a proof that $\mathcal{F}$ preserves piecewise affinity, so that all iterates $\mathbf{V}^i_\ell$ can be computed using piecewise-affine functions. In order to bound the size of $\mathbf{V}^i_\ell$ (in particular, its number of pieces), we need the fine encoding via cells and partition value functions previously introduced.

We are now able to compute the values of a divergent weighted timed game $\mathcal{G}$, thus proving Theorem 4.7. By definition, such a game contains no 0-cycles, and thus the kernel K is empty. In this case, the semi-unfolding $\mathcal{T}(\mathcal{G})$ is a tree of depth i exponential in $\mathcal{G}$, and thus of doubly-exponential size. Proposition 4.29 ensures that the values of $\mathcal{T}(\mathcal{G})$ coincide with the values of $\mathcal{G}$. By Theorem 4.30, we can compute the (exact) values in $\mathcal{T}(\mathcal{G})$ in time doubly-exponential in i and exponential in the size of $\mathcal{T}(\mathcal{G})$. We thus obtain

a triply-exponential algorithm computing the value of a divergent WTG. Equivalently, this shows that the value problem is in 3-EXPTIME for divergent WTGs.

Approximation of kernels

To generalise our computations to almost-divergent weighted timed games, we start by approximating a kernel K of a weighted timed game $\mathcal{G}$ by extending the region-based approximation schema of [BJM15]. More precisely, we thus consider here a weighted timed game $\mathcal{G}$ composed of a kernel (a subgraph of a region game containing only 0-cycles, as defined in Section 4.4) and some target locations equipped with final weights. In the setting of [BJM15], not only cycles but all finite plays in kernels have weight 0, allowing for a reduction to a finite game. In our setting, we have to approximate the timed dynamics of runs, and therefore resort to the corner-point abstraction (as shown on the right of Figure 4.5).

Our goal is to compute an ε-approximation of the value of the kernel (in a given initial configuration). Since final weight functions are piecewise affine with a finite number of pieces and continuous on regions, they are Λ-Lipschitz-continuous on regions, for a given constant $\Lambda \geqslant 0$. The technique to obtain the approximation is to consider regions of a refined enough granularity: we thus pick

$$N(\varepsilon, \Lambda) = \left\lceil \frac{w_{\max}^L \, |L| |\mathsf{Reg}(\mathcal{X}, M)| + \Lambda}{\varepsilon} \right\rceil \tag{4.3}$$

later denoted N when the parameters ε and Λ are clear from context.

Consider then the corner-point abstraction $\Gamma_N(\mathcal{G})$ described in Section 2.5, with locations of the form (ℓ, r, v) such that v is a corner of the $1/N$-region r. Two plays ρ of $\mathcal{G}$ and ρ' of $\Gamma_N(\mathcal{G})$ are said to be $1/N$-*close* if they follow the same path p in $\mathcal{R}_N(\mathcal{G})$. In particular, at each step the configurations (ℓ, ν) in ρ and (ℓ', r', v') in ρ' (with v' a corner of the $1/N$-region r') satisfy $\ell = \ell'$ and $\nu \in r'$, and the edges taken in both plays have the same discrete weights. Close plays have *close* weights, in the following sense:

Lemma 4.31. *For all $1/N$-close plays ρ of $\mathcal{G}$ and ρ' of $\Gamma_N(\mathcal{G})$, $|\mathsf{wt}_{\mathcal{G}}(\rho) - \mathsf{wt}_{\Gamma_N(\mathcal{G})}(\rho')| \leqslant \varepsilon$.*

In particular, if we start in some configurations (ℓ, ν) of $\mathcal{G}$, and $((\ell, r, \mathsf{v}), \mathsf{v})$ of $\Gamma_N(\mathcal{G})$, with $\nu \in r$, since both players have the ability to stay $1/N$-close all along the plays, a bisimulation argument allows us to obtain that the values of the two games are also close in (ℓ, ν) and $((\ell, r, \mathsf{v}), \mathsf{v})$, respectively:

Lemma 4.32. *For all locations $\ell \in L$, $1/N$-regions r, valuations $\nu \in r$ and corners v of r, $|\mathsf{Val}_{\mathcal{G}}(\ell, \nu) - \mathsf{Val}_{\Gamma_N(\mathcal{G})}((\ell, r, \mathsf{v}), \mathsf{v})| \leqslant \varepsilon$.*

We can thus obtain an ε-approximation of $\mathsf{Val}_{\mathcal{G}}(\ell, \nu)$ by computing $\mathsf{Val}_{\Gamma_N(\mathcal{G})}((\ell, r, \mathsf{v}), \mathsf{v})$ for any corner v of r. Recall that $\Gamma_N(\mathcal{G})$ can be considered as an (untimed) shortest-path game. Thus we can apply the result of Chapter 1, where it is shown that the optimal values of such games can be computed in pseudo-polynomial time (i.e. polynomial with respect to the number of locations, at most $(|\mathcal{X}|+1)|L||\mathsf{Reg}_N(\mathcal{X}, M)|$, and the weights of transitions $w_{\max}$ encoded in unary, instead of binary): more precisely, the value $\mathsf{Val}_{\Gamma_N(\mathcal{G})}$ is obtained as the i-th iterate of an operator, with $i = ((2(|\mathcal{X}|+1)|L||\mathsf{Reg}_N(\mathcal{X}, M)| - 1)w_{\max}+1)(|\mathcal{X}|+1)|L||\mathsf{Reg}_N(\mathcal{X}, M)|$, polynomial in $|L|$, $w_{\max}$, M and N, and exponential

in $|\mathcal{X}|$. We then define an ε-approximation of $\mathsf{Val}_{\mathcal{G}}$, named Val'_N, on each $1/N$-region by interpolating the values of its $1/N$-corners in $\Gamma_N(\mathcal{G})$ with a piecewise-affine function: if ν is a valuation that belongs to the $1/N$-region r, then ν can be expressed as a (unique) convex combination of the $1/N$-corners v of r, so that $\nu = \sum_{\mathsf{v}} \alpha_{\mathsf{v}} \mathsf{v}$ with $\alpha_{\mathsf{v}} \in [0,1]$ for all v, and we let $\mathsf{Val}'_N(\ell,\nu) = \sum_{\mathsf{v}} \alpha_{\mathsf{v}} \mathsf{Val}_{\Gamma_N(\mathcal{G})}((\ell,r,\mathsf{v}),\mathsf{v})$ for all locations ℓ of $\mathcal{G}$.

Moreover, we can control the growth of the Lipschitz constant of the approximated value for further use.

Lemma 4.33. *Val'_N is an ε-approximation of $\mathsf{Val}_{\mathcal{G}}$, i.e. $\|\mathsf{Val}'_N - \mathsf{Val}_{\mathcal{G}}\|_\infty \leqslant \varepsilon$. Moreover, Val'_N is piecewise affine with a finite number of pieces and $2(w^L_{\max}|L||\mathsf{Reg}(\mathcal{X},M)|+\Lambda)$-Lipschitz-continuous over regions.*

The time complexity needed to compute Val'_N is polynomial in $|L|$, $w_{\max}$, M and N, and exponential in $|\mathcal{X}|$.

Approximation of almost-divergent weighted timed games

We now explain how to approximate the value of an almost-divergent weighted timed game $\mathcal{G}$, thus proving Theorem 4.12. After having computed the semi-unfolding $\mathcal{T}(\mathcal{G})$ described in Section 4.5, we perform a bottom-up computation of the approximation. The addition of kernels (with respect to the case of divergent weighted timed games studied before) requires us to compute an approximation instead of the actual value. Notice that the techniques used in Theorem 4.30, applied for $i = 1$ step (and not for the whole tree as for divergent weighted timed games), allow us to compute the value of a non-kernel node of $\mathcal{T}(\mathcal{G})$, depending on the values of its children. There is no approximation needed here, so that if we have computed ε-approximations of all its children, we can compute an ε-approximation of the node. More formally, this is justified by the following lemma with $i = 1$:

Lemma 4.34. *Let $\mathcal{G}$ and $\mathcal{G}'$ be two weighted timed games that only differ on the final weight functions fwt and fwt'. Then, $\|\mathsf{Val}_{\mathcal{G}} - \mathsf{Val}_{\mathcal{G}'}\|_\infty \leqslant \|\mathsf{fwt} - \mathsf{fwt}'\|_\infty$. Moreover, for all $i \in \mathbb{N}$, $\|\mathsf{Val}^i_{\mathcal{G}} - \mathsf{Val}^i_{\mathcal{G}'}\|_\infty \leqslant \|\mathsf{fwt} - \mathsf{fwt}'\|_\infty$.*

We now explain in details the full process of approximation of the value $\mathsf{Val}_{\mathcal{G}}(\ell_0,\nu_0)$ of an almost-divergent WTG $\mathcal{G}$: it is a bottom-up computation on the tree T rooted in (ℓ_0,r_0) (with r_0 the region of ν_0) that we used to describe the semi-unfolding $\mathcal{T}(\mathcal{G})$. By Proposition 4.29, the value we want to approximate is equal to $\mathsf{Val}_{\mathcal{T}(\mathcal{G})}((\tilde{\ell}_0,r_0),\nu_0)$. For a node s in T, let $\mathbf{V}_s$ denote the exact value function of the corresponding node in $\mathcal{T}(\mathcal{G})$. In particular, the value function at the root of T is equal to $\nu \mapsto \mathsf{Val}_{\mathcal{T}(\mathcal{G})}((\tilde{\ell}_0,r_0),\nu)$. Our algorithm iteratively computes an approximated value function $\mathbf{V}'_s$ for all nodes s of T.

To obtain an adequate ε-approximation of $\mathsf{Val}_{\mathcal{G}}(\ell_0,\nu_0)$, we will thus need to guarantee a precision in kernels that depend on the number of kernels we visit in the semi-unfolding. Let α be the maximal number of kernels along a branch of the tree T. For a given node s in T, we also let $\alpha(s)$ be the maximal number of kernels along the branches of the subtree rooted in s. Finally, let us denote by $h(s)$ the maximal length of a branch in the subtree rooted in s.

We will maintain, along the bottom-up computation, that $\mathbf{V}'_s$ is a Λ_s-Lipschitz-continuous mapping on regions such that

$$\Lambda_s \leqslant \max(2,|\mathcal{X}|)^{h(s)}(2w^L_{\max}|L||\mathsf{Reg}(\mathcal{X},M)|+\Lambda) \quad \text{and} \quad \|\mathbf{V}_s - \mathbf{V}'_s\|_\infty \leqslant \alpha(s)\varepsilon/\alpha \tag{4.4}$$

where Λ is the maximal Lipschitz constant of the final weight functions of $\mathcal{G}$. In particular, at the root of T where $\alpha(s) = \alpha$, we indeed recover an ε-approximation of the value of the game in configuration (ℓ_0, ν_0).

For leaf nodes s of T, $\mathbf{V}_s$ and $\mathbf{V}'_s$ are equal to the final weight function fwt of $\mathcal{T}(\mathcal{G})$, and we thus get the invariant (4.4) for free (knowing that $\alpha(s) = h(s) = 0$).

For an internal node s of T (that either gives rise to a single state (ℓ, r) of $\mathcal{T}(\mathcal{G})$, or to a kernel $\mathsf{K}_{\ell,r}$), let us suppose that the value of all children s' of s in T have been computed and verify the invariant (4.4).

If s is a node of the form (ℓ, r) (that is not part of a kernel), we define two WTGs $\widetilde{\mathcal{G}}$ and $\widetilde{\mathcal{G}}'$ that contain the state (ℓ, r) as well as its children in T. The children s' are made target states with respective final weight functions given by $\mathbf{V}_{s'}$ and $\mathbf{V}'_{s'}$. Moreover, we know that

$$\alpha(s) = \max_{s'} \alpha(s') \text{ and } h(s) = \max_{s'} h(s') + 1$$

By definition, $\mathbf{V}_s$ is equal to $\nu \mapsto \mathsf{Val}_{\widetilde{\mathcal{G}}}(s, \nu)$ and thus to $\nu \mapsto \mathsf{Val}^1_{\widetilde{\mathcal{G}}}(s, \nu)$ since $\widetilde{\mathcal{G}}$ is acyclic of depth 1. Our approximation algorithm consists in letting $\mathbf{V}'_s = \nu \mapsto \mathsf{Val}^1_{\widetilde{\mathcal{G}}'}(s, \nu)$. By Lemma 4.34 (with $i = 1$ and ε being $\max_{s'} \alpha(s')\varepsilon/\alpha$), $\|\mathbf{V}_s - \mathbf{V}'_s\|_\infty \leqslant \alpha(s)\varepsilon/\alpha$. It is possible to show that $\mathbf{V}'_s$ is Λ_s-Lipschitz-continuous on regions with $\Lambda_s \leqslant \max(\max_{s'} \Lambda_{s'}, |\mathsf{wt}(\ell)| + (|\mathcal{X}| - 1) \max_{s'} \Lambda_{s'})$: with the help of the invariant (4.4) for all children s', and since $|\mathsf{wt}(\ell)| \leqslant w^L_{\max}$ and $h(s') \leqslant h(s) - 1$, we obtain $\Lambda_s \leqslant \max(2, |\mathcal{X}|)^{h(s)}(2w^L_{\max} |L||\mathsf{Reg}(\mathcal{X}, M)| + \Lambda)$.

If s is a node of the form $\mathsf{K}_{\ell,r}$, we define two WTGs $\widetilde{\mathcal{G}}$ and $\widetilde{\mathcal{G}}'$, that contain the locations of the kernel $\mathsf{K}_{\ell,r}$ of $\mathcal{T}(\mathcal{G})$, as well as the children of s in T (reached by output edges of $\mathsf{K}_{\ell,r}$). The children s' are made target states of respective final weight functions $\mathbf{V}_{s'}$ and $\mathbf{V}'_{s'}$. Moreover, we know that

$$\alpha(s) = \max_{s'} \alpha(s') + 1 \text{ and } h(s) = \max_{s'} h(s') + 1$$

Thus, games $\widetilde{\mathcal{G}}$ and $\widetilde{\mathcal{G}}'$ are identical, except for their final weight functions that are ε-close. Thus, we know by Lemma 4.34 that $\|\mathsf{Val}_{\widetilde{\mathcal{G}}} - \mathsf{Val}_{\widetilde{\mathcal{G}}'}\|_\infty \leqslant \max_{s'} \alpha(s')\varepsilon/\alpha$. Moreover, by definition, $\mathbf{V}_s$ is equal to $\nu \mapsto \mathsf{Val}_{\widetilde{\mathcal{G}}}(s, \nu)$. Our approximation algorithm consists in letting $\mathbf{V}'_s$ be equal to an ε/α-approximation of $\mathsf{Val}_{\widetilde{\mathcal{G}}'}$, obtained by Lemma 4.33 with a granularity $N(\varepsilon/\alpha, \max_{s'} \Lambda_{s'})$: we thus have $\|\mathsf{Val}_{\widetilde{\mathcal{G}}'}(s, \cdot) - \mathbf{V}'_s\|_\infty \leqslant \varepsilon/\alpha$, and $\mathbf{V}'_s$ is Λ_s-Lipschitz-continuous on regions with $\Lambda_s \leqslant 2(w^L_{\max} |L||\mathsf{Reg}(\mathcal{X}, M)| + \max_{s'} \Lambda_{s'})$. By triangular inequality, we deduce that $\|\mathbf{V}_s - \mathbf{V}'_s\|_\infty \leqslant \max \alpha(s')\varepsilon/\alpha + \varepsilon/\alpha = \alpha_s\varepsilon/\alpha$. Moreover, with the help of invariant (4.4) for $\Lambda_{s'}$, we once again obtain that $\Lambda_s \leqslant \max(2, |\mathcal{X}|)^{h(s)}(2w^L_{\max} |L||\mathsf{Reg}(\mathcal{X}, M)| + \Lambda)$.

We thus obtain an algorithm that faithfully computes an ε-approximation of the value of the game. Let us now discuss the complexity of the algorithm. Overall, the biggest Lipschitz constant for $\mathbf{V}'$ is $\max(2, |\mathcal{X}|)^h(2w^L_{\max} |L||\mathsf{Reg}(\mathcal{X}, M)| + \Lambda)$ with h the height of the semi-unfolding that is $|\mathcal{R}(\mathcal{G})|(3|\mathcal{R}(\mathcal{G})|w_{\max} + 2w^{\mathsf{t}}_{\max} + 2) + 1$ as noticed in (4.2) (page 85). This Lipschitz constant is thus at most doubly-exponential with respect to the size of $\mathcal{G}$. Therefore, the biggest granularity N used in kernel approximations (described in (4.3)) can be globally bounded as doubly-exponential in the size of $\mathcal{G}$ and polynomial in $1/\varepsilon$. This entails that each kernel approximation can be performed in time doubly-exponential in the size of $\mathcal{G}$, and polynomial in $1/\varepsilon$. As the height of the semi-unfolding is

at most exponential in the size of $\mathcal{G}$, the number of kernel approximations needed is at most doubly-exponential. The rest of the algorithm consist in applying Theorem 4.30, outside of the kernels. A careful complexity analysis allows one to obtain a triply-exponential algorithm computing an ε-approximation of the value of an almost-divergent WTG, which concludes the proof of Theorem 4.12. This complexity is polynomial in $1/\varepsilon$ as N is linear in $1/\varepsilon$.

A symbolic algorithm

The previous approximation result suffers from several drawbacks. It relies on the SCC decomposition of the region automaton. Each of these SCCs have to be analysed in a sequential way, and their analysis requires an a priori refinement of the granularity of regions. This approach is thus not easily amenable to implementation. We instead explain how to follow a symbolic approach as stated in Theorem 4.13, based on the value iteration paradigm, i.e. the computation of iterates of the operator $\mathcal{F}$, recalled in page 71.

Notice that configurations with value $+\infty$ are stable through value iteration, and do not affect its other computations. Since Theorem 4.13 assumes the absence of configurations of value $-\infty$, we will therefore consider in the following that all configurations have finite value in $\mathcal{G}$.

Consider first a game $\mathcal{G}$ that is a kernel, with final weight functions that are Λ-Lipschitz-continuous on regions. By Lemma 4.32, there exists an integer $N(\varepsilon, \Lambda)$ such that solving the untimed weighted game $\Gamma_{N(\varepsilon,\Lambda)}(\mathcal{G})$ computes an $\varepsilon/2$-approximation of the value of $1/N$ corners.

Using the results of Chapter 1, we know that those values are obtained after a finite number of steps of (the untimed version of) the value iteration operator, depending on the number $|L||\mathsf{Reg}_{N(\varepsilon,\Lambda)}(\mathcal{X}, M)|(|\mathcal{X}|+1)$ of locations of the region game. More precisely, if one considers a number of iterations

$$P(\varepsilon, \Lambda) = |L||\mathsf{Reg}_{N(\varepsilon,\Lambda)}(\mathcal{X}, M)|(|\mathcal{X}|+1)(2(|L||\mathsf{Reg}_{N(\varepsilon,\Lambda)}(\mathcal{X}, M)|(|\mathcal{X}|+1)-1)w_{\max}+1)\,,$$

then $\mathsf{Val}^{P(\varepsilon,\Lambda)}_{\Gamma_{N(\varepsilon,\Lambda)}(\mathcal{G})}((\ell, r, \mathsf{v}), \mathsf{v}) = \mathsf{Val}_{\Gamma_{N(\varepsilon,\Lambda)}(\mathcal{G})}((\ell, r, \mathsf{v}), \mathsf{v})$. From this observation, we deduce the following property of $P(\varepsilon, \Lambda)$:

Lemma 4.35. *If $\mathcal{G}$ is a kernel with no configurations of infinite value, then $|\mathsf{Val}_{\mathcal{G}}(\ell, \nu) - \mathsf{Val}^{P(\varepsilon,\Lambda)}_{\mathcal{G}}(\ell, \nu)| \leqslant \varepsilon$ for all configurations (ℓ, ν) of $\mathcal{G}$.*

Once we know that value iteration converges on kernels, we can use the semi-unfolding of Section 4.5 to prove that it also converges on non-negative SCCs when all values are finite.

Lemma 4.36. *If $\mathcal{G}$ is a non-negative SCC with no configurations of infinite value, we can compute a bound $P_+(\varepsilon, \Lambda)$ such that $|\mathsf{Val}_{\mathcal{G}}(\ell, \nu) - \mathsf{Val}^{P_+(\varepsilon,\Lambda)}_{\mathcal{G}}(\ell, \nu)| \leqslant \varepsilon$ for all configurations (ℓ, ν) of $\mathcal{G}$.*

Proving the same property on non-positive SCCs requires more work, because the semi-unfolding gives final weight $-\infty$ to stop leaves, which does not integrate well with value iteration (initialisation at $+\infty$ on non-target states). However, by unfolding those SCCs slightly more (at most $|\mathcal{R}(\mathcal{G})|$ more steps), we can obtain the desired property with a similar bound P_-.

Lemma 4.37. *If $\mathcal{G}$ is a non-positive SCC with no configurations of infinite value, we can compute $P_-(\varepsilon, \Lambda)$ such that $|\mathsf{Val}_{\mathcal{G}}(\ell, \nu) - \mathsf{Val}_{\mathcal{G}}^{P_-(\varepsilon,\Lambda)}(\ell, \nu)| \leqslant \varepsilon$ for all configurations (ℓ, ν) of $\mathcal{G}$.*

Now, if we are given an almost-divergent weighted timed game $\mathcal{G}$ and a precision ε, we can add the bounds for value iteration obtained from each SCC by Lemmas 4.36 and 4.37, and obtain a final bound P such that for all $k \geqslant P$, $\mathsf{Val}_{\mathcal{G}}^k$ is an ε-approximation of $\mathsf{Val}_{\mathcal{G}}$. This concludes the proof of Theorem 4.13.

Overall, this leads to an upper bound complexity that is of the order of a tower of α exponentials, with α exponential in the size of the input game. However, we argue that this symbolic procedure is more amenable to implementation than the previous approximation schema. First, it avoids the three already mentioned drawbacks (SCC decomposition, sequential analysis of the SCCs, and refinement of the granularity of regions) of the previous approximation schema. Then, it allows one to directly launch the value iteration algorithm on the game $\mathcal{G}$, and we can stop the computation whenever we are satisfied enough by the approximation computed: however, there are no guarantees whatsoever on the quality of the approximation before the number of steps P given above.

If $\mathcal{G}$ is not guaranteed to be free of configurations of value $-\infty$, then we must first perform the SCC decomposition of $\mathcal{R}(\mathcal{G})$, and, as $\mathcal{G}$ is almost-divergent, identify and remove regions whose value is $-\infty$, by Proposition 4.28. Then, we can apply the value iteration algorithm.

We also note that if $\mathcal{G}$ is divergent, the unfolding of kernels is not needed, so that Lemmas 4.36 and 4.37 construct bounds $P_+(\varepsilon, \Lambda)$ and $P_-(\varepsilon, \Lambda)$ allowing one to compute the exact values of $\mathcal{G}$. Moreover, these bounds become exponential in the size of $\mathcal{G}$ instead of being non-elementary, so that the overall complexity of the symbolic algorithm is 3-EXPTIME, matching the result of Section 4.6.

As a final remark, notice that our correctness proof strongly relies on Section 4.6, and thus would not hold with the approximation schema of [BJM15] (which does not preserve the continuity on regions of the computed value functions, in turn needed to define final weights on $\frac{1}{N(\varepsilon,\Lambda)}$-corners).

4.7 Strategy synthesis

We are also interested in the synthesis problem, that asks for an ε-optimal strategy of Min. In this section, we will prove the following result:

Theorem 4.38. *Let $\varepsilon > 0$ and let $\mathcal{G}$ be a divergent weighted timed game. We can compute in triple exponential time an ε-optimal strategy for* Min.

Recall that in the value iteration algorithm of Section 4.1, one step of the game is summarised by the operator $\mathcal{F}$ mapping each value function $\mathbf{V}$ to a value function $\mathbf{V}' = \mathcal{F}(\mathbf{V})$ (defined in (4.1)). Intuitively, ε-optimal strategies can be extracted from the value iteration operator, by mapping each play that ends in (ℓ, ν) with $\ell \in L_{\mathsf{Min}}$ to a choice (t, e) such that the transition $(\ell, \nu) \xrightarrow{t,e} (\ell', \nu')$ is ε-optimal. However, the choice depends on the step i in the value iteration computation. Formally, if A is a set, f is a mapping from A to $\mathbb{R}_\infty$ and $\varepsilon > 0$, let $\operatorname{arginf}^{\varepsilon}_{a \in A} f(a)$ denote the set of elements $a^* \in A$ such that $f(a^*) \leqslant \inf_{a \in A} f(a) + \varepsilon$. Then, let us name $\sigma^{i,\varepsilon}_{\mathsf{Min}}$ a strategy defined from the

application of (4.1) in $\mathbf{V}^i = \mathcal{F}(\mathbf{V}^{i-1})$, so that for all finite plays ρ ending in (ℓ, ν) and $\ell \in L_{\mathsf{Min}}$,

$$\sigma^{i,\varepsilon}_{\mathsf{Min}}(\rho) \in \operatorname{arginf}^{\varepsilon}_{(\ell,\nu) \xrightarrow{t,e} (\ell',\nu')} \left(t \cdot \mathsf{wt}(\ell) + \mathsf{wt}(e) + \mathbf{V}^{i-1}_{\ell'}(\nu')\right) \tag{4.5}$$

Now, let $\sigma^{\star,i,\varepsilon}_{\mathsf{Min}}$ denote a strategy that maps every finite play ρ ending in $L_{\mathsf{Min}} \setminus L_{\mathsf{t}}$ to $(t, e) = \sigma^{j,\varepsilon}_{\mathsf{Min}}(\rho)$, with $j = \max(0, i - |\rho|)$.

Proposition 4.39. *The strategy $\sigma^{\star,i,\varepsilon}_{\mathsf{Min}}$ is εi-optimal for* Min *during the first i steps:*

$$|\mathsf{Val}^i((\ell,\nu), \sigma^{\star,i,\varepsilon}_{\mathsf{Min}}) - \mathsf{Val}^i(\ell,\nu)| \leqslant \varepsilon i$$

We now explain how to extract some strategies $\sigma^{i,\varepsilon}_{\mathsf{Min}}$ from the partition value functions, in order to solve the synthesis problems on acyclic and divergent games (games where a known $i \in \mathbb{N}$ implies $\mathsf{Val} = \mathsf{Val}^i$).

We will use affine inequalities over $n+1$ variables to encode constraints on the choice of delays. Formally, an affine delay inequality is an equation I of the form $a_{\mathsf{d}}\mathsf{d} \prec a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b$, with all a_i, b and a_{d} integers of $\mathbb{Z}$, $a_{\mathsf{d}} \neq 0$ and $\prec \in \{<, \leqslant\}$. We say that I is a lower bound if $a_{\mathsf{d}} < 0$, and that it is an upper bound if $a_{\mathsf{d}} > 0$.

Definition 4.40. A *partition strategy function* S defined over a partition P is a mapping from the base cells of P to a tuple $(e, I_{\mathbf{l}}, I_{\mathbf{u}}, p)$, where e is an edge of $\mathcal{G}$, $I_{\mathbf{l}}$ and $I_{\mathbf{u}}$ are two affine delay inequalities that are respectively lower and upper bounds, and finally $p \in \{\mathbf{l}, \mathbf{u}\}$ selects one of the two.

Given a precision $\varepsilon > 0$, a partition strategy function encodes a mapping from c_P to pairs (e, d^ε) denoted $[\![S]\!]^\varepsilon$, with $e \in E$ and $d^\varepsilon \in \mathbb{R}_{\geqslant 0}$. Let $\nu \in c_P$ and $S([\nu]_P)$ be equal to $(e, I_{\mathbf{l}}, I_{\mathbf{u}}, p)$, with $I_{\mathbf{l}}$ defined by $a_{\mathsf{d}}\mathsf{d} \prec a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b$ and $I_{\mathbf{u}}$ defined by $a'_{\mathsf{d}}\mathsf{d} \prec' a'_1\mathsf{x}_1 + \cdots + a'_n\mathsf{x}_n + b'$. Let $D \subseteq \mathbb{R}_{\geqslant 0}$ denote the interval of delays d that satisfy both $a_{\mathsf{d}}\mathsf{d} \prec a_1\nu(\mathsf{x}_1) + \cdots + a_n\nu(\mathsf{x}_n) + b$ and $a'_{\mathsf{d}}\mathsf{d} \prec' a'_1\nu(\mathsf{x}_1) + \cdots + a'_n\nu(\mathsf{x}_n) + b'$. We denote by $d_{\mathbf{l}}$ and $d_{\mathbf{u}}$ the lower and upper bounds of D. Let D^ε denote $D \cap (d_{\mathbf{l}} - \varepsilon, d_{\mathbf{l}} + \varepsilon)$ if $p = \mathbf{l}$, and $D \cap (d_{\mathbf{u}} - \varepsilon, d_{\mathbf{u}} + \varepsilon)$ if $p = \mathbf{u}$. We denote by $d^\varepsilon_{\mathbf{l}}$ and $d^\varepsilon_{\mathbf{u}}$ the lower and upper bounds of D^ε. Then, $[\![S]\!]^\varepsilon(\nu) = (e, d^\varepsilon)$ with $d^\varepsilon = \frac{d^\varepsilon_{\mathbf{l}} + d^\varepsilon_{\mathbf{u}}}{2} \in D^\varepsilon$.

We argue that partition strategy functions can be used to compute a positional strategy $\sigma^{i,\varepsilon}_{\mathsf{Min}}$ given an encoding of $\mathsf{Val}^{i-1}_{\mathcal{G}}$ as partition value functions, so that (4.5) holds.

We assume that for all $\ell \in L$, $\mathbf{V}^i_\ell$ is piecewise affine with finitely many pieces and is Λ-Lipschitz-continuous over regions. By the previous study, it is possible to obtain that Λ is at most exponential in the size of $\mathcal{G}$ and i.

Proposition 4.41. *For all $\varepsilon > 0$ and $i \in \mathbb{N} \setminus \{0\}$, we can compute in time doubly-exponential in i and exponential in the size of $\mathcal{G}$ a partition P^i_ℓ and a partition strategy function S^i_ℓ for each location $\ell \in L_{\mathsf{Min}} \setminus L_{\mathsf{t}}$, so that $[\![S^i_\ell]\!]^{\frac{\varepsilon}{\Lambda}}(\nu) = \sigma^{i,\varepsilon}_{\mathsf{Min}}(\ell,\nu)$.*

By Proposition 4.39, we can therefore solve the synthesis problem in triple exponential time for all weighted time games $\mathcal{G}$ so that $\mathsf{Val}_{\mathcal{G}} = \mathsf{Val}^i_{\mathcal{G}}$ with i at most exponential in the size of $\mathcal{G}$. This holds for acyclic games, where i is bounded by $|\mathcal{R}(\mathcal{G})|$, and also for divergent games, where i can be bounded by Lemmas 4.36 and 4.37 (in the special case with no kernel), assuming that an exponential time pre-computation using Proposition 4.28 is used to remove the valuations of value $-\infty$. This concludes the proof of Theorem 4.38.

We can then refine this study in order to construct good strategies for Min that have a special form, the switching strategies, already studied before in Chapters 1 and 3. Recall that a switching strategy σ_{Min} is described by two deterministic memoryless strategies σ^1_{Min} and σ^2_{Min}, as well as a switching threshold α. The strategy σ_{Min} then consists in playing strategy σ^1_{Min} until either we reach a target location, or the finite play has length at least α, in which case we switch to strategy σ^2_{Min}.

Theorem 4.42. *In a divergent weighted timed game, for all $\varepsilon > 0$ and $N \in \mathbb{N}$, there exists a switching strategy σ_{Min} for Min such that for all configurations (ℓ, ν), $\mathsf{Val}^{\sigma_{\mathsf{Min}}}(\ell, \nu) \leqslant \max(-N, \mathsf{Val}(\ell, \nu)) + \varepsilon$.*

In particular, if all configurations have a finite value, there exists an ε-optimal switching strategy. In the presence of a configuration with value $-\infty$, we can build from Theorem 4.42 a family of switching strategies (indexed by the parameter N) whose value tends to $-\infty$.

The proof of Theorem 4.42 requires to build both strategies σ^1_{Min} and σ^2_{Min}, as well as a switching threshold α. As usual, the second strategy σ^2_{Min} only consists in reaching the target and is thus obtained as a memoryless strategy from a classical attractor computation in the region game $\mathcal{R}(\mathcal{G})$. In contrast, the first strategy σ^1_{Min} requires more care. We build it so that it fulfils two properties, that we summarise in saying that σ^1_{Min} is *fake-ε-optimal*:

1. each finite play conforming to σ^1_{Min} from (ℓ, ν) and reaching the target has a cumulated weight at most $\mathsf{Val}(\ell, \nu) + |\rho|\,\varepsilon$ (in particular, if $\mathsf{Val}(\ell, \nu) = -\infty$, no such plays should exist);
2. each finite play conforming to σ^1_{Min} following a *long enough* cycle in the region game $\mathcal{R}(\mathcal{G})$ has a cumulated weight at most -1.

Here, "fake" means that σ^1_{Min} is not obliged to guarantee reaching the target, but if it does so, it must do it with a cumulated weight close to $\mathsf{Val}_{\ell,\nu}$, the error factor depending linearly on the size of the play. The second property ensures that playing long enough σ^1_{Min} without reaching the target results in diminishing the cumulated weight. Then, if the switch happens at horizon α big enough, Min is sure that the cumulated weight so far is low enough so that the rest of the play to reach a target location (following σ^2_{Min} only) will not make the weight increase too much. In the absence of values $-\infty$ in Val, the first property allows one to obtain a $\alpha\varepsilon$-optimal strategy even in the case where the switch does not occur (because we reach the target prematurely). The construction of a fake-ε/α-optimal strategy σ^1_{Min} (the linear dependency on the length of the play in the first property of fake-optimality is thus taken care by a division by α here) relies on the fact that Val is a fixpoint of the operator $\mathcal{F}$, to play almost-optimally at horizon 1: $\sigma^1_{\mathsf{Min}}(\ell, \nu)$ is chosen as one of the edges $(\ell, \nu) \xrightarrow{t,e} (\ell', \nu')$ such that the value $t \times \mathsf{wt}(\ell) + \mathsf{wt}(e) + \mathsf{Val}_{\ell',\nu'}$ is close enough to $\mathcal{F}(\mathsf{Val})_{\ell,\nu}$.

Turning our study to the point of view of Max, looking for good strategies for this other player, we show that no switch is necessary to play ε-optimally: memoryless strategies are sufficient to guarantee a value as close as wanted to the deterministic value. For a configuration with a value equal to $-\infty$, all the deterministic strategies for Max are equivalent where they are all equally bad. Without loss of generality, we can therefore suppose that there are no configurations in $\mathcal{G}$ with a value equal to $-\infty$. Then, remaining values are bounded in absolute value by $w^e_{\max}|\mathcal{R}(\mathcal{G})|$, since *optimal plays* have no cycles.

We use that fact to build a memoryless deterministic strategy σ_{Max} analogous to strategy σ^1_{Min} before:

Theorem 4.43. *In a divergent weighted timed game, there exists a memoryless ε-optimal strategy for player* Max.

Conclusion

Our study of weighted timed games belongs to a series of works that explore the frontier of decidability. We introduced the first decidable class of weighted timed games with arbitrary weights and no restrictions on the number of clocks. We have given an approximation procedure for a larger class of weighted timed games, where the exact problem becomes undecidable. In addition, we have proved the correctness of a symbolic approximation schema, that does not start by splitting exponentially every region, but only does so when necessary. We argue that this paves the way towards an implementation of value approximation for weighted timed games. Such tool would likely struggle with instances of moderate size, but could help with the design and testing of alternative approaches that trade theoretical guarantees with performance.

Another perspective is to extend this work to the concurrent setting, where both players play simultaneously and the shortest delay is selected. It should be noted that several known results on weighted timed games with non-negative weights [Bou+04b; ABM04; BJM15] are stated in such a concurrent setting. We did not consider this setting so far because concurrent weighted timed games are not determined, and several of our proofs rely on this property for symmetrical arguments (mainly to lift results of non-negative strongly connected components to non-positive ones).

A long-standing open problem is the approximation of weighted timed games, i.e. whether one can compute an arbitrarily close approximation of the value of a given game. We successfully solved this problem on the class of almost-divergent games, but we were not able to extend further our techniques to more general games. As a first step, we could try to consider the slightly larger class of 0-isolated games, where we ask for every cycle of the region game to have a weight either $\geqslant 1$, or $\leqslant -1$, or exactly 0. We do not have approximation results on this 0-isolated class, and as such it forms a natural intermediate step between the best known decidable class and the general case. However we must prepare ourselves to possibly negative answers (the value of a weighted timed game could be *non approximable*). Therefore, pursuing better lower bounds in various settings could help in the future, in order to close the remaining complexity gaps.

5 Random Strategies in Weighted Timed Games

This chapter presents results obtained with Julie Parreaux and Pierre-Alain Reynier, published in the conferences CONCUR [MPR20] and ICALP [MPR21].

Table of contents

Strategies considered so far in this manuscript are deterministic. Though the game has no stochastic edges, it is possible to allow players to use stochastic strategies: a player may choose, depending on the current history, the probability distribution on the successors. For the controller (Min), this is a chance to describe less firmly their strategy, and let some place to the randomness: it could blur the lines sufficiently to let them secure a greater value a priori (even if we will see that this is not the case, in the following). For the environment (Max), allowing stochastic strategies (possibly with memory) strengthens the modelling power of weighted (timed) games since it forces a priori Min to optimise their payoff even in the presence of uncertainties.

In game theory, enabling randomisation in the strategies is often crucial. For instance, Nash equilibria are only ensured to exist in matrix games (like rock-paper-scissors) when players can play at random [Nas50]. In the context of games on graphs, the use of randomisation in strategies is only interesting in classes of games where memory is needed since otherwise memoryless (deterministic) strategies suffice. For shortest-path games and weighted timed games, the study is thus very much interesting in the presence of both positive and negative weights, where we have observed in previous chapters the need for memory, finite or even infinite, in strategies.

The tradeoff between memory and randomisation has already been investigated in many classes of games where memory is required to win or play optimally. This is for instance the case for qualitative games like Street or Müller games thoroughly studied (with and without randomness in the arena) in [CAH04]. The study has been extended to timed games [CHP08a] where the goal is to use as little information as possible about the precise values of real-time clocks. Memory or randomness is also crucial in multi-dimensional objectives [CRR14]: for instance, in mean-payoff parity games, if there exists a deterministic finite-memory winning strategy, then there exists a randomised memoryless almost-sure winning strategy.

In the context of shortest-path games, here is the intuition leading to the tradeoff we study between memory and randomness. Consider the shortest-path game depicted on the left of Figure 5.1, that we already studied thoroughly in Chapter 1 (see Figure 1.2). Recall that, from v_2, player Min could reach directly ☺, thus leading to a payoff of 0, or

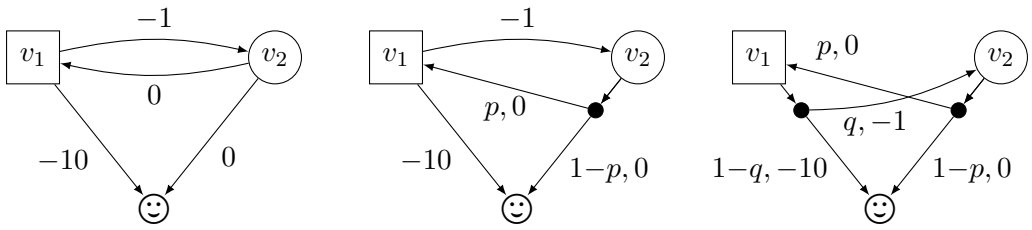

Figure 5.1 – On the left, a shortest-path game, where Min requires memory to play optimally. In the middle, the Markov decision process obtained when letting Min play at random, with a parametric probability $p \in (0, 1)$. On the right, the Markov chain obtained when Max plays along a memoryless randomised strategy, with a parametric probability $q \in [0, 1]$.

choose to go to v_1, in which case Max either jumps directly in ☺ (leading to a beneficial payoff -10), or comes back to v_2, but having already capitalised a total payoff -1. We can continue this way *ad libitum* until Min is satisfied (at least 10 times) and jumps to ☺. This guarantees a value at most -10 for Min when starting in v_2. Reciprocally, Max can guarantee a payoff at least -10 by directly jumping into ☺ when they must play for the first time. Thus, the optimal value is -10 when starting from v_1 or v_2. However, Min cannot achieve this optimal value by playing without memory. We have introduced the notion of switching strategy that allows a concise representation of an optimal strategy for Min: turning in a negative cycle long enough, before reaching the target. In general, such strategies use pseudo-polynomial memory with respect to the weights of the game graph.

In this example, such a switching strategy can be mimicked using randomisation only (and no memory), Min deciding to go to v_1 with high probability $p < 1$ and to go to the target vertex with the remaining low probability $1 - p > 0$ (we enforce this probability to be positive, in order to reach the target with probability 1, no matter how the opponent is playing). The resulting Markov decision process (MDP) is depicted in the middle of Figure 5.1. The shortest-path problem in such MDPs has been thoroughly studied in [BT91], where it is proved that Max does not require memory to respond optimally. Denoting by q the probability that Max jumps in v_2 in its memoryless strategy, we obtain the Markov chain (MC) on the right of Figure 5.1. We can compute the expected value in this MC, as well as the best strategies for both players: in the overall, the optimal value remains -10, even if Min no longer has an optimal strategy. They rather have an ε-optimal strategy, consisting in choosing $p = 1 - \varepsilon/10$ that ensures a value at most $-10 + \varepsilon$. The first contribution of this chapter is to show that deterministic memory and memoryless randomisation always provide the same power to Min in shortest-path games.

We then lift this result to the weighted timed games. A first important challenge is to analyse how to play stochastically in such timed games. Starting from a notion of stochastic behaviours in a timed automaton considered in [Ber+14] (for the one-player setting), we propose a new class of stochastic strategies. Compared with [Ber+14], our class is larger in the sense that we allow Dirac distributions for delays, which subsumes the setting of deterministic strategies. However, in order to ensure that strategies yield a well-defined probability distribution on sets of executions, we need measurability properties stronger than the one considered in [Ber+14] (we actually provide an example showing

that their hypothesis was not strong enough).

Then, we turn our attention towards the expected cumulated weight of the set of plays conforming to a pair of stochastic strategies. We first prove that under the previous measurability hypotheses, this expectation is well-defined when restricting to the set of plays following a finite sequence of transitions. In order to have the convergence of the global expectation, we identify another property of strategies of Min, which intuitively ensures that the set of target locations is reached quickly enough. This allows us to define a notion of stochastic value (resp. memoryless stochastic value) of the game, i.e. the best value Min can achieve using stochastic strategies (resp. memoryless stochastic strategies), when Max uses stochastic strategies (resp. memoryless stochastic strategies) too.

To lift the results from finite to timed games, we strongly rely (for now) on the presence of switching strategies for Min obtained at the end of the previous chapter for divergent weighted timed games. We will thus focus on this class of games, and show that the two versions of stochastic values are equal to the deterministic value. In other terms, we show that Min can emulate memory using randomisation, and vice versa. Moreover, combining memory and randomisation does not increase Min's capabilities.

5.1 Playing stochastically in shortest-path games

We extend the notion of strategies of both players to be able to incorporate randomisation. For a set V, we denote by $\mathsf{Dist}(V)$ the set of *distributions* over V (equipped with an underlying σ-algebra). When V is finite, these are all mappings $f\colon V \to [0,1]$ such that $\sum_{v\in V} f(v) = 1$. The support of a distribution f is the set $\{v \in V \mid f(v) > 0\}$, denoted by $\mathrm{supp}(f)$. A Dirac distribution is a distribution with a singleton support: the Dirac distribution of support $\{v\}$ is denoted by Dirac_v.

In this chapter, we thus now call a strategy for Min over a shortest-path game $\mathcal{G}$ any mapping $\sigma_{\mathsf{Min}}\colon \mathsf{FPlays}^{\mathsf{Min}} \to \mathsf{Dist}(V)$ such that for all finite play $\rho \in \mathsf{FPlays}^{\mathsf{Min}}$ ending in vertex $v_k \in V_{\mathsf{Min}}$, the support of the distribution $\sigma_{\mathsf{Min}}(\rho)$ is included in $E(v_k)$ (the possible successors of v_k). A play or finite play [1] $\rho = v_0 \to v_1 \to \cdots \to v_k \cdots$ conforms to the strategy σ_{Min} if for all k such that $v_k \in V_{\mathsf{Min}}$, we have that $\sigma_{\mathsf{Min}}(\rho[k])(v_{k+1}) > 0$. A similar definition allows one to define strategies $\sigma_{\mathsf{Max}}\colon \mathsf{FPlays}^{\mathsf{Max}} \to \mathsf{Dist}(V)$ for Max, and plays conforming to them. We let $\mathsf{r}\Sigma_{\mathsf{Min}}$ and $\mathsf{r}\Sigma_{\mathsf{Max}}$ be the (randomised) strategies of players Min and Max, respectively.

The previous notion of strategies, used so far in this manuscript, can be recovered: a strategy σ is *deterministic* if for all finite plays ρ, $\sigma(\rho)$ is a Dirac distribution, and in this case, we let (as before) $\sigma(\rho)$ denote the unique label in the support of this Dirac distribution. To clarify the distinction, we change the notations of the sets of deterministic strategies of players Min and Max, respectively, to let them be $\mathsf{d}\Sigma_{\mathsf{Min}}$ and $\mathsf{d}\Sigma_{\mathsf{Max}}$.

A strategy σ is *memoryless* if for all finite plays ρ, ρ' ending in the same vertex, we have $\sigma(\rho) = \sigma(\rho')$. We let $\mathsf{m}\Sigma_{\mathsf{Min}}$ and $\mathsf{m}\Sigma_{\mathsf{Max}}$ be the memoryless (possibly randomised) strategies of players Min and Max, respectively.

When starting in a vertex v_0, we would like to measure the *expected* shortest-path payoff of the plays conforming to some strategies $\sigma_{\mathsf{Min}} \in \mathsf{r}\Sigma_{\mathsf{Min}}$ and $\sigma_{\mathsf{Max}} \in \mathsf{r}\Sigma_{\mathsf{Max}}$. We start by defining a probability distribution on the sets of plays. Indeed, sets of plays can

1. Once again, in this untimed setting, we do not need the labels on edges, and thus remove them from all the notations.

be equipped with a σ-algebra, generated by cylinders that are all plays starting with a given finite prefix. By Caratheodory's theorem, defining a probability measure on this σ-algebra only requires to associate a probability with each finite play. Thus, letting ρ be a finite play from v_0 conforming to $\sigma_{\mathsf{Min}} \in \mathsf{r}\Sigma_{\mathsf{Min}}$ and $\sigma_{\mathsf{Max}} \in \mathsf{r}\Sigma_{\mathsf{Max}}$, we let $\mathbb{P}_{v_0}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}(\rho)$ be defined as 1 if $\rho = v_0$, and

$$\mathbb{P}_{v_0}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}(\rho \to v) = \begin{cases} \mathbb{P}_{v_0}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}(\rho) \times \sigma_{\mathsf{Min}}(\rho)(v) & \text{if } \rho \in \mathsf{FPlays}^{\mathsf{Min}} \\ \mathbb{P}_{v_0}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}(\rho) \times \sigma_{\mathsf{Max}}(\rho)(v) & \text{if } \rho \in \mathsf{FPlays}^{\mathsf{Max}} \end{cases}$$

We then also write $\mathbb{P}_{v_0}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}$ the unique probability measure on the sets of plays. In particular, letting TPaths_{v_0} be the set of plays from v_0 that reach a target, this is a measurable set of plays since it is a countable union of finite plays (the ones reaching the target). We then write $\mathbb{P}_{v_0}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}(\mathsf{TPaths}_{v_0})$ the probability to reach the target from v_0, while following strategies σ_{Min} and σ_{Max}.

The expected shortest-path payoff starting from vertex v_0, conforming to strategies $\sigma_{\mathsf{Min}} \in \mathsf{r}\Sigma_{\mathsf{Min}}$ and $\sigma_{\mathsf{Max}} \in \mathsf{r}\Sigma_{\mathsf{Max}}$, is then defined as

$$\mathbb{E}_{v_0}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}} = \sum_{\rho \in \mathsf{TPaths}_{v_0}} \mathbb{P}_{v_0}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}(\rho)\mathbf{TP}(\rho)$$

We use the total-payoff function here, instead of the shortest-path payoff, since the reachability of the target is ensured by the fact that ρ is taken in TPaths_{v_0}.

Under the condition $\mathbb{P}_{v_0}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}(\mathsf{TPaths}_{v_0}) = 1$ to reach the target with probability 1, we can show that when ρ grows, the probability $\mathbb{P}_{v_0}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}(\rho)$ diminishes exponentially, while $\mathbf{TP}(\rho)$ only grows linearly, which ensures the convergence of the above series:

Lemma 5.1. *Under the condition that $\mathbb{P}_{v_0}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}(\mathsf{TPaths}_{v_0}) = 1$, the sum in the definition of the expected payoff $\mathbb{E}_{v_0}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}$ converges to a real number.*

To prove that, we strongly rely on the fact that at any time of the computation the number of possible successors of a vertex is finite, since counter-examples of convergence can easily be found when infinite branching occurs: this will be one of the main challenges of the timed generalisation in the rest of this chapter.

We leave to Min the responsibility for guaranteeing the condition of reaching the target with probability 1. This asymmetric choice with respect to the players is grounded in our controller synthesis background, Min being the controller desiring to reach a target location with minimum expected payoff, while Max is an uncontrollable environment.

Definition 5.2. A strategy $\sigma_{\mathsf{Min}} \in \mathsf{r}\Sigma_{\mathsf{Min}}$ is said to be proper if for all vertices v_0 and strategies $\sigma_{\mathsf{Max}} \in \mathsf{r}\Sigma_{\mathsf{Max}}$, $\mathbb{P}_{v_0}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}(\mathsf{TPaths}_{v_0}) = 1$. We let $\mathsf{r}\Sigma_{\mathsf{Min}}^{\mathsf{p}}$ be the set of proper strategies, as well as $\mathsf{m}\Sigma_{\mathsf{Min}}^{\mathsf{p}}$ the subset of memoryless proper strategies.

We are then ready to define three possible values for shortest-path games. In this randomised context, showing determinacy, i.e. the equality of upper and lower values, is more challenging (though feasible by using Blackwell determinacy results [Mar98]), and we will indeed not rely on such results in the following, only focusing on the value of the *controller*, i.e. the upper value that Min can guarantee.

— The most general definition is the *randomised value*: for a strategy σ_{Min}, we let

$$\mathsf{pVal}(v_0, \sigma_{\mathsf{Min}}) = \sup_{\sigma_{\mathsf{Max}} \in \mathsf{r}\Sigma_{\mathsf{Max}}} \mathbb{E}_{v_0}^{\sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}}}$$

so that the upper randomised value is then given by

$$\overline{\mathsf{pVal}}(v_0) = \inf_{\sigma_{\mathsf{Min}} \in \mathsf{r}\Sigma^{\mathsf{p}}_{\mathsf{Min}}} \mathsf{pVal}(v_0, \sigma_{\mathsf{Min}})$$

— the usual *deterministic value* that we have studied so far in this manuscript is obtained by limiting the previous formulas to deterministic strategies, in which case the expected payoff is limited to the payoff of the unique generated play: we now denote by dVal this values (instead of just Val);

— and finally the *memoryless value*, when restricting players to play with memoryless (randomised) strategies:

$$\mathsf{mVal}(v_0, \sigma_{\mathsf{Min}}) = \sup_{\sigma_{\mathsf{Max}} \in \mathsf{m}\Sigma_{\mathsf{Max}}} \mathbb{E}_{v_0}^{\sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}}}$$
$$\overline{\mathsf{mVal}}(v_0) = \inf_{\sigma_{\mathsf{Min}} \in \mathsf{m}\Sigma^{\mathsf{p}}_{\mathsf{Min}}} \mathsf{mVal}(v_0, \sigma_{\mathsf{Min}})$$

Then, our contribution is as follows:

Theorem 5.3. *For all finite shortest-path games, and all vertices v_0,*

$$\overline{\mathsf{pVal}}(v_0) = \mathsf{dVal}(v_0) = \overline{\mathsf{mVal}}(v_0)$$

Therefore, Min does not gain power when they are allowed to play at random, and randomisation without memory is able to mimic the memory required to play optimally in such games.

Let us briefly consider the point of view of Max. Notice that when Min has chosen their proper strategy σ_{Min}, Max must choose theirs as if they were playing in a Markov decision process (MDP) whose states are in bijection with the memory used in σ_{Min}. This MDP has thus possibly infinitely many states but still a finite branching (since each state of the MDP has a number of outgoing edges at most the maximal outdegree of vertices in $\mathcal{G}$). Under this context, it is possible to generalise results shown in [BK08; BT91] for finite MDPs, to show that Max has a *best response* that is well-behaving:

Lemma 5.4. *In an MDP with finite branching,* Max *has an optimal strategy that is deterministic and memoryless.*

Proof. For all strategies σ_{Max} of Max, generating a vector of expected payoff (x_s) for all states s of the MDP, we construct a deterministic and memoryless strategy σ'_{Max} generating a greater expected payoff. For all states s of the MDP, we choose $\sigma'_{\mathsf{Max}}(s)$ in the support of $\sigma_{\mathsf{Max}}(s)$, that maximises the expected payoff at horizon 1 obtained by adding the weight of the next edge (s, s') to the expected payoff $x_{s'}$. Thanks to Bellman equations in Markov chains, and by using Knaster-Tarski theorem, we can compare the fixpoints of two operators in the two Markov chains obtained by playing with σ_{Max} and σ'_{Max}, to show that they are equal. □

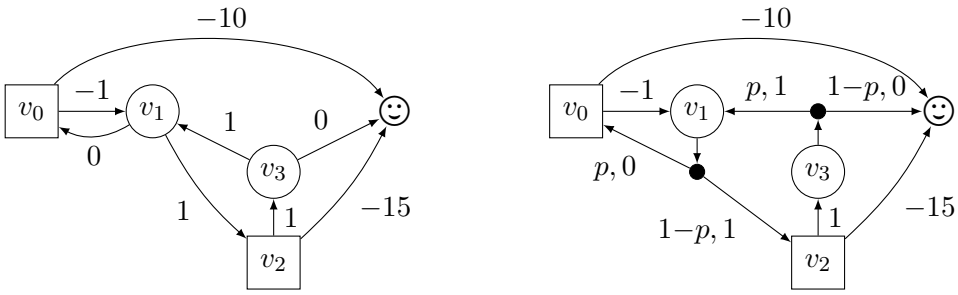

Figure 5.2 – On the left, a more complex example of shortest-path game. On the right, the MDP associated with a randomised strategy of Min with a parametric probability $p \in (0, 1)$.

Example 5.5. In Figure 5.1, a shortest-path game is presented on the left, with the MDP in the middle obtained by picking as a memoryless strategy for Min the one choosing to go to v_1 with probability $p \in (0, 1)$ and to the target vertex with probability $1 - p$. Another more complex example is given in Figure 5.2 where the memoryless strategy for Min consists, in vertex v_1, to choose successor v_0 with probability $p \in (0, 1)$ and successor v_2 with probability $1 - p$, and in vertex v_3, to choose successor v_1 with the same probability p and the target vertex with probability $1 - p$.

We now sketch the proof of Theorem 5.3. First, by the inclusion of deterministic strategies into stochastic ones and relying on the previous lemma showing that Max can always respond with a deterministic strategy, we can easily obtain that

Lemma 5.6. *For all finite shortest-path games, and all vertices v_0,*

$$\overline{\mathsf{pVal}}(v_0) \leqslant \mathsf{Val}(v_0)$$

We show the other inequalities by a simulation of deterministic strategies with randomised (memoryless) ones, and vice versa. We start here by ruling out the case of values $+\infty$. Indeed, $\mathsf{dVal}(v) = +\infty$ signifies that Min is not able to reach a target vertex from v with deterministic strategies. This also implies that Min has no memoryless randomised strategies to ensure reaching the target with probability 1, and thus $\overline{\mathsf{mVal}}(v) = +\infty$. Reciprocally, if $\overline{\mathsf{mVal}}(v) = +\infty$, then Min has no memoryless strategies to reach the target with probability 1 (since this is the only reason for having a value $+\infty$). Since reachability is a purely qualitative objective, and the game graph does not contain probabilities, Min cannot use memory in order to guarantee reaching the target: therefore, this also means that $\mathsf{dVal}(v) = \overline{\mathsf{pVal}} = +\infty$. In the end, we have shown that $\mathsf{dVal}(v) = +\infty$ if and only if $\overline{\mathsf{mVal}}(v) = +\infty$ if and only if $\overline{\mathsf{pVal}}(v) = +\infty$. We thus remove every such vertex from now on, which does not change the values of other vertices in the game.

Simulating deterministic strategies with memoryless strategies

We first show that $\overline{\mathsf{mVal}}(v_0) \leqslant \mathsf{dVal}(v_0)$. The trick of Lemma 5.6 cannot be applied since memoryless strategies and deterministic strategies are orthogonal subsets of strategies. We proceed by a simulation of a deterministic strategy of Min with a memoryless strategy. To ease the process (and this is indeed crucial, as we will also see in the timed setting,

since we do not know how to proceed without it), we start with a switching strategy for Min, which we know by Chapter 1 to be sufficient for Min to play optimally. We thus let σ^1_{Min} and σ^2_{Min} be two memoryless strategies composing the switching strategy, and α be the switching parameter α. The strategy of Min then consists in playing along σ^1_{Min}, until eventually switching to σ^2_{Min} when the length of the current finite play is greater than α. Moreover, strategy σ^1_{Min} is chosen so that every cyclic finite play conforming to σ^1_{Min} has a negative total weight. For all $N \in \mathbb{N}$, the parameter α can vary so that the switching strategy σ_{Min} has a deterministic value $\mathsf{dVal}(v_0, \sigma_{\mathsf{Min}}) \leqslant \max(-N, \mathsf{dVal}(v_0))$. We simulate it with a memoryless (randomised) strategy, denoted σ^p_{Min}, with a parametrised probability $p \in (0,1)$. This new strategy is a probabilistic superposition of the two memoryless deterministic strategies σ^1_{Min} and σ^2_{Min}.

Formally, we define σ^p_{Min} on each strongly connected components (SCC) of the graph according to the presence of a negative cycle. In an SCC that does not contain negative cycles, for each vertex $v \in V_{\mathsf{Min}}$ of the SCC, we let $\sigma^p_{\mathsf{Min}}(v) = \mathsf{Dirac}_{\sigma^1_{\mathsf{Min}}(v)}$: player Min chooses to play the first strategy σ^1_{Min} of the switching strategy, thus looking for a negative cycle in the next SCCs (in topological order) if any. In an SCC that contains a negative cycle, for each vertex $v \in V_{\mathsf{Min}}$ of the SCC, we let $\sigma^p_{\mathsf{Min}}(v)$ be the distribution of support $\{\sigma^1_{\mathsf{Min}}(v), \sigma^2_{\mathsf{Min}}(v)\}$ that chooses $\sigma^1_{\mathsf{Min}}(v)$ with probability p and $\sigma^2_{\mathsf{Min}}(v)$ with probability $1-p$, except if $\sigma^1_{\mathsf{Min}}(v) = \sigma^2_{\mathsf{Min}}(v)$ in which case we choose it with probability 1. Note that the MDPs in Figures 5.1 and 5.2 are obtained by applying this strategy σ^p_{Min}.

Notice that σ^p_{Min} is a proper strategy, i.e. for all strategies $\sigma_{\mathsf{Max}} \in \mathsf{m}\Sigma_{\mathsf{Max}}$,

$$\mathbb{P}_{v_0}^{\sigma^p_{\mathsf{Min}}, \sigma_{\mathsf{Max}}}(\mathsf{TPaths}_{v_0}) = 1$$

Thanks to the characterisation of [BK08, Lemma 10.111], this can be shown by proving that for all $\sigma_{\mathsf{Max}} \in \mathsf{m}\Sigma_{\mathsf{Max}}$, all bottom SCCs (the ones we cannot exit) of the Markov chain obtained by playing along σ^p_{Min} and σ_{Max} consist in a unique target vertex, which holds because of our definition of σ^p_{Min}.

To conclude, we then only need to show that for $\varepsilon \in \mathbb{R}_{\geqslant 0}$ small enough, there exists $\tilde{p}$ close enough to 1 such that for all $p \in [\tilde{p}, 1)$,

$$\mathsf{mVal}(v_0, \sigma^p_{\mathsf{Min}}) \leqslant \mathsf{dVal}(v_0, \sigma_{\mathsf{Min}}) + \varepsilon$$

This entails the expected result. Indeed, if $\mathsf{dVal}(v_0) \in \mathbb{Z}$, we get (with $N = |\mathsf{dVal}(v_0)|$) that $\mathsf{mVal}(v_0, \sigma^p_{\mathsf{Min}}) \leqslant \mathsf{dVal}(v_0) + \varepsilon$, and thus $\overline{\mathsf{mVal}}(v_0) \leqslant \mathsf{dVal}(v_0)$ since this holds for all $\varepsilon > 0$. Otherwise, $\mathsf{dVal}(v_0) = -\infty$, and letting N tend towards $+\infty$, we also get $\overline{\mathsf{mVal}}(v_0) = -\infty$.

Thanks to Lemma 5.4, against σ^p_{Min}, Max has a best response that is deterministic, therefore, we can fix a deterministic and memoryless strategy σ_{Max} and show that

$$\mathbb{E}_{v_0}^{\sigma^p_{\mathsf{Min}}, \sigma_{\mathsf{Max}}} \leqslant \mathsf{dVal}(v, \sigma_{\mathsf{Min}}) + \varepsilon \tag{5.1}$$

whenever $p < 1$ is close enough to 1. By gathering the finite number of lower bounds about p, for all deterministic memoryless strategies of Max (there are a finite number of such), we obtain the desired lower bound $\tilde{p}$ for p.

The case where the whole game graph does not contain any negative cycle is easy. In this case, σ^p_{Min} chooses the strategy σ^1_{Min} with probability 1, by definition since no SCC

contain a negative cycle (this is the only reason why we defined σ^p_{Min} as it is, for such SCCs): a play from initial vertex v_0 conforming to σ^p_{Min} is thus conforming to σ^1_{Min}. Since the graph contains no negative cycles and all cycles conforming to σ^1_{Min} must be negative, all plays from v_0 conforming to σ^1_{Min} reach the target set of vertices, with a total payoff at most $\mathsf{dVal}(v_0, \sigma_{\mathsf{Min}})$. This single play has probability 1, thus $\mathbb{E}^{\sigma^p_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{v_0} \leqslant \mathsf{dVal}(v_0, \sigma_{\mathsf{Min}})$, which proves that $\mathsf{mVal}(v, \sigma^p_{\mathsf{Min}}) \leqslant \mathsf{dVal}(v_0, \sigma_{\mathsf{Min}})$ as expected.

Now, suppose that the graph game contains negative cycles. We let $c > 0$ be the maximal size of an elementary cycle (that visits a vertex at most once), $w^- > 0$ be the opposite of the maximal weight of an elementary negative cycle, and $w^+ \geqslant 0$ be the maximal weight of an elementary non-negative cycle (or 0 if such a cycle does not exist).

Example 5.7. In the graph of Figure 5.1, we have $c = 2$, $w^- = 1$, and $w^+ = 0$ (since there is no non-negative cycles). In the game graph of Figure 5.2, we have $c = 3$, $w^- = 1$, and $w^+ = 3$.

The difficulty comes from the possible presence of non-negative cycles too. Indeed, when applying the switching strategy σ_{Min}, all cycles conforming to σ^1_{Min} have a negative weight. This is no longer true with the probabilistic superposition σ^p_{Min}, as can be seen in the example of Figure 5.2. Finding an adequate lower-bound for p requires to estimate $\mathbb{E}^{\sigma^p_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{v_0}$, by controlling the weight and probability of non-negative cycles, balancing them with the ones of negative cycles. The crucial argument comes from the definition of the superposition σ^p_{Min}, which implies that all cycles conforming to σ^p_{Min} and σ_{Max} of non-negative total weight contain at least one edge of probability $1-p$.

Showing (5.1) is then done by partitioning the set Π of plays starting in v_0, conforming to σ^p_{Min} and σ_{Max}, and reaching the target set of vertices, into subsets $\Pi_{i,\ell}$ according to the number i of edges of probability $1-p$ they go through, and their length ℓ (we always have $i \leqslant \ell$). The partition is depicted in Figure 5.3:

— $\Pi_{0,\mathbb{N}}$, depicted in yellow, contains all plays with no edges of probability $1-p$;

— $\Pi_{>0,\geqslant L}$, depicted in blue, contains all plays with $i \geqslant 1$ edges of probability $1-p$, and a length of at least $L(i) = ia + b$ with

$$a = \left\lceil c\left(1 + \frac{w^+}{w^-}\right)\right\rceil \quad \text{and} \quad b = \frac{|\mathsf{dVal}(v_0, \sigma_{\mathsf{Min}})| + |V|W + w^-}{w^-}c + |V|$$

— $\Pi_{>0,<L}$, depicted in red, is the rest of the plays, i.e. plays with $i \geqslant 1$ edges of probability $1-p$ and a length less than $L(i)$. We also let $\Pi_{i,<L(i)}$ be the set of plays with $i \geqslant 1$ edges of probability $1-p$, and a length of at most $L(i)$, so that $\Pi_{>0,<L}$ is the union of all such sets.

Partitioning the plays allows us to carefully control non-negative cycles: plays with a large enough length can compensate for the presence of non-negative cycles and thus obtain a favorable weight ($< \mathsf{dVal}(v_0, \sigma_{\mathsf{Min}})$). We let $\gamma_{0,\mathbb{N}}$ (resp. $\gamma_{>0,\geqslant L}$ and $\gamma_{>0,<L}$) be the expectation $\mathbb{E}^{\sigma^p_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{v_0}$ restricted to plays in $\Pi_{0,\mathbb{N}}$ (resp. $\Pi_{>0,\geqslant L}$ and $\Pi_{>0,<L}$). By linearity of expectation,

$$\mathbb{E}^{\sigma^p_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{v_0} = \gamma_{0,\mathbb{N}} + \gamma_{>0,\geqslant L} + \gamma_{>0,<L}$$

The end of the proof, very technical and that can be found in the report version of the article available at https://arxiv.org/abs/2005.04985, consists in controlling separately the

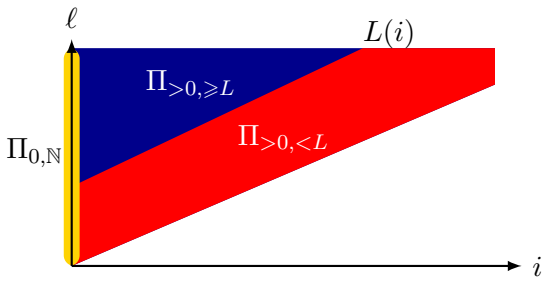

Figure 5.3 – Partition of plays Π.

three terms to obtain the desired inequality (5.1).

Simulating memoryless strategies with deterministic strategies

To finish the proof of Theorem 5.3, we show that $\mathsf{dVal}(v_0) \leqslant \overline{\mathsf{mVal}}(v_0)$ and $\mathsf{dVal}(v_0) \leqslant \overline{\mathsf{pVal}}(v_0)$: we here focus on the second inequality, but both are shown exactly the same way. For a given strategy σ_{Min} ensuring that Min reaches the target set V_t with probability 1, and that can be memoryless or not, we build a deterministic strategy σ'_{Min} which guarantees a value $\mathsf{dVal}(v_0, \sigma'_{\mathsf{Min}})$ at least as good as $\mathsf{pVal}(v_0, \sigma_{\mathsf{Min}}) + \varepsilon$, with $\varepsilon > 0$ as small as we want.

The first attempt to build a deterministic strategy σ'_{Min} would be to use classical techniques of finite-memory strategies, for instance in Street or Müller games: for instance, to ensure the visit of two vertices v_1 and v_2 infinitely often during an infinite play (to win a Müller game with winning objective $\{v_1, v_2\}$), we would try to reach v_1 with a first memoryless strategy, and then reach v_2 with another memoryless strategy, before switching again to reach v_1 again, etc. The main reason why this naive approach fails is that the plays are essentially finite in shortest-path games. We thus cannot delay the choices and must carefully play as soon as the play starts.

Instead, our solution is, once again to rely on the concept of switching strategy, and define σ'_{Min} as the switching strategy described by two memoryless deterministic strategies σ^1_{Min} (to be defined later) and σ^2_{Min} (an attractor strategy), and the switching parameter $\alpha = \max(0, |V|W - \mathsf{pVal}(v_0, \sigma_{\mathsf{Min}})) \times |V| + 1$.

Example 5.8. In the game of Figure 5.1, the attractor strategy is $\sigma^2_{\mathsf{Min}}(v_2) = ☺$. We then choose $\sigma^1_{\mathsf{Min}}(v_2)$ so as to minimise the immediate reward obtained by playing one turn and then getting the value ensured by σ_{Min}:

$$\sigma^1_{\mathsf{Min}}(v_2) = \operatorname*{argmin}_{v' \in \{v_1, ☺\}} \left[\mathsf{wt}(v, v') + \mathsf{pVal}(v', \sigma_{\mathsf{Min}})\right] = v_1$$

For an appropriate choice of α, we thus recover the optimal switching strategy for this game.

The construction of σ^1_{Min} in general is split in two parts. First, we restrict the possibilities for $\sigma^1_{\mathsf{Min}}(v)$ to the set

$$\widetilde{E}(v) = \operatorname*{argmin}_{v' \in E(v)} \left[\mathsf{wt}(v, v') + \mathsf{pVal}(v', \sigma_{\mathsf{Min}})\right]$$

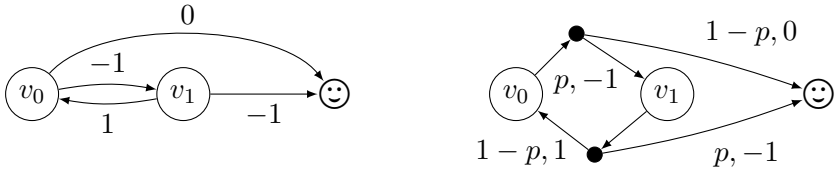

Figure 5.4 – On the left, a game with no negative cycles where σ^p_{Min} is optimal. On the right, the MC obtained when playing along σ^p_{Min}.

of successor vertices of v that minimise the expected value at horizon 1. With respect to the previous example, this forbids the use of edge $(v_2, ☺)$ in particular. We let $\widetilde{\mathcal{G}}$ be the game obtained from $\mathcal{G}$ by removing all edges $v \to v'$ from a vertex $v \in V_{\mathsf{Min}}$ such that $v' \notin \widetilde{E}(v)$. Then, we can show that each finite play of $\widetilde{\mathcal{G}}$ from a vertex v has a total payoff at most $\mathsf{pVal}(v, \sigma_{\mathsf{Min}})$, and each cycle in the game $\widetilde{\mathcal{G}}$ has a non-positive total weight.

Example 5.9. Consider the game graph on the left of Figure 5.4, and the memoryless strategy σ^p_{Min} giving rise to the MDP/MC on the right of Figure 5.4. We can compute $\mathsf{pVal}(v_0, \sigma^p_{\mathsf{Min}}) = -2p^2/(1-p(1-p))$ and $\mathsf{pVal}(v_1, \sigma^p_{\mathsf{Min}}) = (p^2-3p+1)/(1-p(1-p))$. Consider p close enough to 1 so that $\mathsf{pVal}(v_0, \sigma^p_{\mathsf{Min}}) \leqslant -3/2$ and $\mathsf{pVal}(v_1, \sigma^p_{\mathsf{Min}}) \leqslant -1/2$. Then, we have $\widetilde{E}(v_0) = \{v_1\}$ and $\widetilde{E}(v_1) = \{☺\}$. The corresponding game graph $\widetilde{\mathcal{G}}$ contains only edges (v_0, v_1) and $(v_1, ☺)$, and thus no cycles. The unique finite play from vertex v_0 has total-payoff $-2 \leqslant \mathsf{pVal}(v_0, \sigma^p_{\mathsf{Min}})$. In particular, the only possible memoryless deterministic strategy σ^1_{Min} in $\widetilde{\mathcal{G}}$ is optimal in $\mathcal{G}$.

For each vertex v in the game, we also let $d(v)$ be the distance (number of steps) of v to the target given by an attractor computation to the target in $\widetilde{\mathcal{G}}$. We then let, for all vertices $v \in V_{\mathsf{Min}}$, $\sigma^1_{\mathsf{Min}}(v)$ be a vertex $v' \in \widetilde{E}(v)$ that minimises $d(v')$.

Example 5.10. Consider again the game of Figure 5.4, but with a new memoryless strategy σ^p_{Min} defined by $\sigma^p_{\mathsf{Min}}(v_0) = \mathsf{Dirac}_{v_1}$ and $\sigma^p_{\mathsf{Min}}(v_1) = \delta$ such that $\delta(v_0) = 1-p$ and $\delta(☺) = p$, where $p \in (0,1)$. Then, we can check that $\mathsf{pVal}(v_0, \sigma^p_{\mathsf{Min}}) = -2$ and $\mathsf{pVal}(v_1, \sigma^p_{\mathsf{Min}}) = -1$. Thus, $\widetilde{E}(v_0) = \{v_1\}$ and $\widetilde{E}(v_1) = \{v_0, ☺\}$. Not all memoryless deterministic strategies taken in $\widetilde{\mathcal{G}}$ are NC-strategies, since it contains the cycle $v_0 \to v_1 \to v_0$ of cumulated weight 0. We thus apply the construction before, using the fact that $d(☺) = 0$, $d(v_1) = 1$ and $d(v_0) = 2$ (since the edge $(v_0, ☺)$ is not present in $\widetilde{\mathcal{G}}$). Thus, σ^1_{Min} is defined by $\sigma^1_{\mathsf{Min}}(v_0) = v_1$ and $\sigma^1_{\mathsf{Min}}(v_1) = ☺$, and is indeed an NC-strategy.

The final step is to show that σ^1_{Min} is always an NC-strategy (i.e. all cycles of $\widetilde{\mathcal{G}}$ conforming with σ^1_{Min} have a negative total weight). Combined with our choice of switching parameter α, we can show that the total-payoff of every play conforming to σ'_{Min}, that must necessarily reach the target, is at most the expected value obtained by playing along σ_{Min}.

Characterisation of optimality

All shortest-path games with values different from $-\infty$ admit an optimal deterministic strategy for both players: however, Min may require memory to play optimally. In this case, we also have seen that Min does not have an optimal memoryless (randomised)

strategy: they only have ε-optimal ones, for all $\varepsilon > 0$. But some shortest-path games indeed admit optimal memoryless strategies for Min: the strategy σ^p_{Min} described above is indeed optimal in games not containing negative cycles, for instance. We characterise the shortest-path games in which Min admits an optimal memoryless strategy.

Remember from Chapter 1 that, in order to compute values $\mathsf{dVal}(v)$, we may use a value iteration paradigm consisting in computing iterates $(X^{(i)})_{i\geqslant 0}$ in $\mathbb{Z}^V_\infty$ of an operator $\mathcal{F}\colon \mathbb{Z}^V_\infty \to \mathbb{Z}^V_\infty$ defined in (1.1) (page 20). We introduce a new notion, being the most permissive strategy of Min at each step $i \geqslant 0$ of the computation. It maps each vertex $v \in V_{\mathsf{Min}}$ to the set

$$\widetilde{E}^{(i)}(v) = \{v' \in E(v) \mid \mathsf{wt}(v,v') + X^{(i-1)}_{v'} = X^{(i)}_v\}$$

of vertices that Min can choose. For each such most permissive strategy $\widetilde{E}^{(i)}$, we let $\widetilde{\mathcal{G}}^{(i)}$ be the game graph where we remove all edges (v,v') with $v \in V_{\mathsf{Min}}$ and $v' \notin \widetilde{E}^{(i)}(v)$. This allows us to state the following result:

Theorem 5.11. *Let $\mathcal{G}$ be a shortest-path game such that $\mathsf{dVal}(v) \neq -\infty$ for all vertices v. The following assertions are equivalent:*

1. Min *has an optimal memoryless deterministic strategy in $\mathcal{G}$ (for dVal);*
2. Min *has an optimal memoryless (randomised) strategy in $\mathcal{G}$ (for $\overline{\mathsf{mVal}}$);*
3. $X^{(|V|-1)}_v = X^{(|V|)}_v = \mathsf{dVal}(v)$ *for all vertices v (this means that the sequence $(X^{(i)})_i$ is stationary as soon as step $|V|-1$), and* Min *can guarantee to reach V_t from all vertices in the game graph $\widetilde{\mathcal{G}}^{(|V|-1)}$.*

This characterisation of the existence of optimal memoryless strategy is testable in polynomial time since it is enough to compute vectors $X^{(|V|-1)}$ and $X^{(|V|)}$, check their equality, compute the sets $\widetilde{E}^{(|V|-1)}(v)$ (this can be done while computing $X^{(|V|)}$) and check whether Min can guarantee reaching the target in $\widetilde{\mathcal{G}}^{(|V|-1)}$ by an attractor computation. The proof of implication $3 \Rightarrow 1$ is constructive and actually allows one to build an optimal memoryless deterministic strategy when it exists, that respects the most permissive strategy.

5.2 Playing stochastically in weighted timed games

We aim at extending the previous study in the context of weighted timed games. Before doing so, we must introduce stochastic strategies in the context of weighted timed games, that we investigated for the first time a year ago. We will however strongly rely on a line of works aiming at studying stochastic timed automata [Ber+14; Bou+16; Ber+18; Bou+20], thus extending the results in the context of two-player games (instead of model-checking) and with weights, which indeed represents the main challenge in order to give a meaning to the expected payoff.

Naturally, deterministic strategies for Min are extended to more general stochastic strategies as mappings $\sigma_{\mathsf{Min}}\colon \mathsf{FPlays}^{\mathsf{Min}} \to \mathsf{Dist}(\Delta \times \mathbb{R}_{\geqslant 0})$ where each finite play is associated to a probability distribution over the set of pairs of transition and delay. Since Δ is a finite set, this is equivalent to letting first Min choose a transition via $\sigma^{\Delta}_{\mathsf{Min}}\colon \mathsf{FPlays}^{\mathsf{Min}} \to \mathsf{Dist}(\Delta)$, and then, knowing the chosen transition, choose a delay via $\sigma^{\mathbb{R}_{\geqslant 0}}_{\mathsf{Min}}\colon \mathsf{FPlays}^{\mathsf{Min}} \times \Delta \to \mathsf{Dist}(\mathbb{R}_{\geqslant 0})$,

the support of the distribution $\sigma_{\mathsf{Min}}^{\mathbb{R}_{\geqslant 0}}(\rho,\delta)$ being included in the interval $I(\rho,\delta)$ of valid delays. We can then recombine $\sigma_{\mathsf{Min}}^{\Delta}$ and $\sigma_{\mathsf{Min}}^{\mathbb{R}_{\geqslant 0}}$ to obtain the distribution $\sigma_{\mathsf{Min}}(\rho)$. Similar definitions hold for Max whose general strategies are denoted by σ_{Max}.

Notice that deterministic strategies are a special case of such strategies, where the distributions are chosen to be Dirac distributions. Another useful restriction over strategies is the non-use of memory: as before, a strategy σ_{Min} is said to be memoryless if for all finite plays ρ, ρ' ending in the same configuration, we have that $\sigma_{\mathsf{Min}}(\rho) = \sigma_{\mathsf{Min}}(\rho')$. A similar definition holds for Max.

Probability measure on plays.

We fix two randomised strategies σ_{Min} and σ_{Max} for both players, and an initial configuration (ℓ_0, ν_0). Our goal is to define a probability measure on plays. To do so, and following what we have done in the untimed setting and the work of [Ber+14] for stochastic timed automata, the set of plays starting from (ℓ_0, ν_0) and conforming to σ_{Min} and σ_{Max} can naturally be equipped with a structure of σ-algebra whose generators are all subsets of plays that start with a finite prefix following the same sequence π of transitions with some Borel-measurable constraints on the delays taken along π. The a priori idea is thus to define a probability measure $\mathbb{P}_{\ell_0,\nu_0}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}$ on such generators which extends uniquely as a probability measure over the whole σ-algebra, by Carathéodory's extension theorem.

Consider thus a finite sequence of transitions π, starting in location ℓ, and a play ρ ending in the same location ℓ. We define the probability $\mathbb{P}_{\rho}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}(\pi)$ taking into account all possible plays that start with ρ and continue according to π (we leave the Borel-measurable constraints on the delays for now, but discuss them later). It is defined by induction on the length of π by

$$\mathbb{P}_{\rho}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}(\varepsilon) = 1$$

and for all transitions $\delta = (\ell, g, Y, \ell') \in \Delta$,

$$\mathbb{P}_{\rho}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}(\delta\pi) = \int_{I(\rho,\delta)} \sigma_{\mathsf{Min}}^{\Delta}(\rho)(\delta) \times \mathbb{P}_{\rho \xrightarrow{t,\delta}}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}(\pi)\, \mathrm{d}\sigma_{\mathsf{Min}}^{\mathbb{R}_{\geqslant 0}}(\rho,\delta)(t)$$

This definition is very similar to the one in [Ber+14] except that we choose to decouple the distribution on pairs of $\Delta \times \mathbb{R}_{\geqslant 0}$ by first selecting a transition and then delay, whereas authors of [Ber+14] consider independent choices, the one on transitions being described by some weights on transitions (depending on the current region).

For modelling purposes, authors of [Ber+14] enforce that probability distributions on delays do not forbid any delays of the interval $I(\rho,\delta)$ of possible delays, thus ruling out singular distributions like Dirac ones that would consider taking a single possible delay (like deterministic strategies do). More formally, they require $\sigma_{\mathsf{Min}}^{\mathbb{R}_{\geqslant 0}}(\rho,\delta)$ to be absolutely continuous (i.e. equivalent to the Lebesgue measure) on interval $I(\rho,\delta)$. We claim that even with this assumption, the previous definition of the probability may not be well-founded, as demonstrated by the following example.

Example 5.12. Consider the weighted timed game of Figure 5.5, where only Min plays (so we can see it as a stochastic timed automaton), and the memoryless strategy σ_{Min} (partially) defined as follows. We let A be any non-Lebesgue-measurable subset of $[0,1)$.

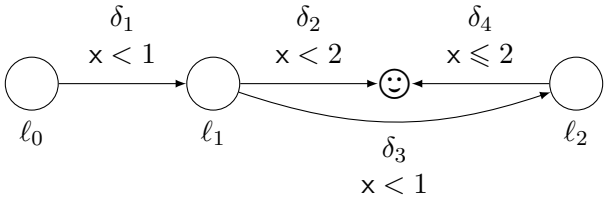

Figure 5.5 – A (one-player) weighted timed game with a single clock x and all weights equal to 0, in which a memoryless strategy for Min does not generate a probability measure.

We denote $\overline{A}$ the set $[0,1) \setminus A$ and $\mathbb{1}_A$ the characteristic function of the set A. We start by defining the delays to match as closely as possible the setting of [Ber+14] here. For delays in ℓ_0 and ℓ_2, we consider uniform probability distributions on the appropriate intervals. In ℓ_1, for all $t_1 \in [0,1)$, we let $\sigma_{\mathsf{Min}}^{\mathbb{R}_{\geqslant 0}}((\ell_1,t_1),\delta_2)$ be the uniform distribution on $[0,2-t_1]$ if $t_1 \in A$, and the truncated exponential distribution on $[0,2-t_1]$ with parameter $\lambda = 1$ otherwise. For the choice of transitions from ℓ_1 (the only place where there is a choice), for all $t_1 \in [0,1)$, we let $\sigma_{\mathsf{Min}}^{\Delta}(\ell_1,t_1)(\delta_2) = (3-t_1)/(4-2t_1) = f(t_1)$ if $t_1 \in A$ and $\sigma_{\mathsf{Min}}^{\Delta}(\ell_1,t_1)(\delta_2) = (1+e^{-(1-t_1)}-2e^{-(2-t_1)})/(2-2e^{-(2-t_1)}) = g(t_1)$ if $t_1 \in \overline{A}$. In the setting of [Ber+14], the strategy in ℓ_1 can be obtained by first choosing a delay similarly as ours, and then choosing transition δ_2 with either probability $1/2$ or 1 depending on whether transition δ_3 is fireable after letting the chosen delay elapse: authors of [Ber+14] would describe this by putting weight 1 on both transitions δ_2 and δ_3 in the stochastic timed automaton. The intricate formulas above for the transitions are thus simply a way to mimic their setting in ours. Let us try to compute the probability of the path $\pi = \delta_1\delta_2$ from configuration $(\ell_0,0)$:

$$\begin{aligned}\mathbb{P}_{\ell_0,0}^{\sigma_{\mathsf{Min}}}(\delta_1\delta_2) &= \int_{I((\ell_0,0),\delta_1)} \sigma_{\mathsf{Min}}^{\Delta}(\ell_0,0)(\delta_1) \times \mathbb{P}_{(\ell_0,0)\xrightarrow{t_1,\delta_1}}^{\sigma_{\mathsf{Min}}}(\delta_2)\, \mathrm{d}\sigma_{\mathsf{Min}}^{\mathbb{R}_{\geqslant 0}}((\ell_0,0),\delta_1)(t_1) \\ &= \int_0^1 1 \times \mathbb{P}_{(\ell_0,0)\xrightarrow{t_1,\delta_1}}^{\sigma_{\mathsf{Min}}}(\delta_2)\, \mathrm{d}t_1\end{aligned}$$

that requires $h\colon t_1 \mapsto \mathbb{P}_{(\ell_0,0)\xrightarrow{t_1,\delta_1}}^{\sigma_{\mathsf{Min}}}(\delta_2)$ to be a measurable function on $[0,1]$ to be well-defined. For $t_1 \in [0,1) \cap A$,

$$h(t_1) = \int_{I((\ell_1,t_1),\delta_1)} \sigma_{\mathsf{Min}}^{\Delta}(\ell_1,t_1)(\delta_2) \times 1\, \mathrm{d}\sigma_{\mathsf{Min}}^{\mathbb{R}_{\geqslant 0}}((\ell_1,t_1),\delta_2)(t_2) = \int_0^{2-t_1} f(t_1)\frac{\mathrm{d}t_2}{2-t_1} = f(t_1)$$

Similarly, if $t_1 \in [0,1) \cap A$, $h(t_1) = g(t_1)$. Functions f and g are measurable and never match over $[0,1)$. Thus, would h be measurable, so would be $(h-g)/(f-t)$ that is equal to the characteristic function of A: this contradicts the non-measurability of A. And thus, it is not possible to define the probability $\mathbb{P}_{\ell_0,0}^{\sigma_{\mathsf{Min}}}(\delta_1\delta_2)$ of path $\delta_1\delta_2$.

From this example, we see the importance to moreover enforce that the distributions $\sigma_{\mathsf{Min}}^{\Delta}(\rho)$ and $\sigma_{\mathsf{Min}}^{\mathbb{R}_{\geqslant 0}}(\rho,\delta)$ are "measurable with respect to the sequence of delays along the play ρ". This is easy to define for the transition part. For delays, since we want deterministic strategies to be a subset of stochastic strategies, we must be able to choose

delays by using Dirac distributions, and by extension discrete distributions (that are not absolutely continuous, as [Ber+14] requires). This results in the following hypothesis:

Hypothesis 5.13. A strategy σ_{Min} satisfies this hypothesis if

1. for all transitions $\delta_0, \dots, \delta_k, \delta$, the mapping

$$(t_0, \dots, t_{k-1}) \mapsto \sigma^{\Delta}_{\mathsf{Min}}((\ell_0, \nu_0) \xrightarrow{(t_0,\delta_0)\cdots(t_{k-1},\delta_{k-1})})(\delta)$$

   is measurable ($(\ell_0, \nu_0) \xrightarrow{(t_0,\delta_0)\cdots(t_{k-1},\delta_{k-1})}$ denotes the unique play starting from (ℓ, ν_0) and firing successively the edges defined by delay t_i and transition t_i);

2. for all plays ρ and transition δ, the probability distribution $\sigma^{\mathbb{R}_{\geqslant 0}}_{\mathsf{Min}}(\rho, \delta)$ (of the random variable t) is described by a cumulative distribution function (CDF) that is the sum of an absolutely continuous function $G(\rho, \delta)$ and Heaviside functions [2] $t \mapsto \sum_i \alpha_i(\rho, \delta) H(t - a_i(\rho, \delta))$. Moreover, for all transitions $\delta_0, \dots, \delta_{k-1}, \delta$, the mappings

$$(t_0, \dots, t_{k-1}, t) \mapsto G(\rho, \delta)(t), \quad (t_0, \dots, t_{k-1}) \mapsto \alpha_i(\rho, \delta), \quad \text{and} \quad (t_0, \dots, t_{k-1}) \mapsto a_i(\rho, \delta)$$

   must be measurable (where ρ denotes the play $(\ell_0, \nu_0) \xrightarrow{(t_0,\delta_0)\cdots(t_{k-1},\delta_{k-1})}$).

This hypothesis allows us to obtain, by induction on the length of the paths (reason why we need to not only define the probability starting from an initial configuration (ℓ_0, ν_0) but more generally from a finite play ρ):

Lemma 5.14. *If σ_{Min} and σ_{Max} are strategies satisfying Hypothesis 5.13, the probabilities $\mathbb{P}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{\rho}(\pi)$ of following a sequence of transitions π after the play ρ are well defined. It can be extended into a probability distribution over maximal paths π starting in the last location of ρ.*

The probability measure easily extends to unions of maximal paths: in particular, $\mathbb{P}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{\ell_0,\nu_0}(\mathsf{TPaths}_{\ell_0,\nu_0})$ is set as the sum $\sum_{\pi \in \mathsf{TPaths}_{\ell_0,\nu_0}} \mathbb{P}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{\ell_0,\nu_0}(\pi)$ of probabilities of all paths reaching L_{t} from ℓ_0. Authors of [Ber+14] go one step further, by using Carathéodory's theorem to extend the probability measure on paths $(\mathbb{P}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{\rho}(\pi))$ to a measure on plays, whose σ-algebra is generated by maximal plays with Borel-measurable constraints on the delays. We do not formally need this further extension and will only use such extension to give an intuitive introduction of the expected payoff below. In the following, we let $\mathsf{r}\Sigma_{\mathsf{Min}}$ and $\mathsf{r}\Sigma_{\mathsf{Max}}$ be the sets of (randomised) strategies satisfying Hypothesis 5.13, for both players. We let $\mathsf{m}\Sigma_{\mathsf{Min}}$ and $\mathsf{m}\Sigma_{\mathsf{Max}}$ be the respective subsets of memoryless strategies.

Expected payoff of plays.

As explained before, by Carathéodory's theorem, the set of plays can be equipped with a probability distribution, and we are interested in the expectation of the random variable $\mathbf{TP}(\rho)$ (where ρ conforms with two fixed strategies σ_{Min} and σ_{Max}). As in the untimed

2. We let H denote the mapping from $\mathbb{R}$ to $[0, 1]$ such that $H(t) = 0$ if $t < 0$ and $H(t) = 1$ otherwise. Recall that it is the CDF of the Dirac distribution choosing $t = 0$.

setting, this only makes sense if the probability to reach a target location is equal to 1. We thus now require that $\mathbb{P}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{\ell_0,\nu_0}(\mathsf{TPaths}_{\ell_0,\nu_0}) = 1$ (i.e. the probability to follow an infinite path is 0). We will see afterwards that this is not a sufficient condition to ensure that the expected payoff is finite.

Intuitively, we would like to let

$$\mathbb{E}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{\ell_0,\nu_0} = \int_{\rho} \mathbf{TP}(\rho)\ \mathrm{d}\mathbb{P}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{\ell_0,\nu_0}(\rho)$$

where the integral ranges over all plays starting in (ℓ_0,ν_0), conforming to σ_{Min} and σ_{Max} that reach the target. To make it more formal, we follow a small detour, consisting in mimicking the construction of the probability before: first define the expected payoff of all plays following a given sequence of transitions, and then sum over all possible such sequences. As before, we also extend the definition to not only fix an initial configuration (ℓ_0,ν_0) but a possibly longer finite play ρ. We would thus consider the expectation along a fixed path π reaching L_{t}:

$$\mathbb{E}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{\rho}(\pi) = \int_{\rho'} \mathbf{TP}(\rho')\ \mathrm{d}\mathbb{P}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{\rho}(\rho') \tag{5.2}$$

where the integral now ranges over the plays ρ' that can extend the prefix of play ρ, and that follow the path π. We would then let

$$\mathbb{E}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{\rho} = \sum_{\pi\in\mathsf{TPaths}_{\rho}} \mathbb{E}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{\rho}(\pi)$$

where TPaths_{ρ} is the set of paths reaching the target that can be put after the finite play ρ.

Once again, the integral in (5.2) is problematic, for the very same reason as above. However, now that we have a finite path π in hand, we can try to unravel it step-by-step. Indeed, if $\pi = \delta\pi'$, and supposing for instance that δ starts in a location ℓ of Min, we would expect $\mathbb{E}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{\rho}(\delta\pi')$ to be equal to:

$$\int_{I(\rho,\delta)} \sigma^{\Delta}_{\mathsf{Min}}(\rho)(\delta)\Big(\int_{\rho'} \mathbf{TP}(\xrightarrow{t,\delta}\rho')\ \mathrm{d}\mathbb{P}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{\rho\xrightarrow{t,\delta}}(\rho')\Big)\mathrm{d}\sigma^{\mathbb{R}_{\geqslant 0}}_{\mathsf{Min}}(\rho,\delta)(t)$$

By definition of the total-payoff and linearity of the integral, this can be rewritten as

$$\int_{I(\rho,\delta)} \sigma^{\Delta}_{\mathsf{Min}}(\rho)(\delta)\Big[\big(t\mathsf{wt}(\ell)+\mathsf{wt}(\delta)\big)\underbrace{\int_{\rho'} \mathrm{d}\mathbb{P}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{\rho\xrightarrow{t,\delta}}(\rho')}_{=\mathbb{P}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{\rho\xrightarrow{t,\delta}}(\pi')} + \underbrace{\int_{\rho'} \mathbf{TP}(\rho')\mathrm{d}\mathbb{P}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{\rho\xrightarrow{t,\delta}}(\rho')}_{=\mathbb{E}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{\rho\xrightarrow{t,\delta}}(\pi')}\Big]\mathrm{d}\sigma^{\mathbb{R}_{\geqslant 0}}_{\mathsf{Min}}(\rho,\delta)(t)$$

With this informal justification in mind, we now give a formal definition, that does not use the probability distribution on *plays*, but only requires the one on *paths*:

Definition 5.15. We define the expected weight $\mathbb{E}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{\rho}(\pi)$ of plays that can extend ρ (the cumulated weight of ρ is thus not counted in the expectation) and that follow the path π. It is defined by induction on the length of π by $\mathbb{E}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{\rho}(\varepsilon) = 0$ and for all transitions $\delta = (\ell, g, Y, \ell')$:

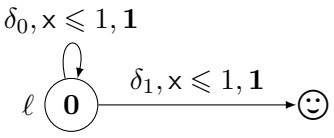

Figure 5.6 – A weighted timed game where Min has a strategy reaching the target with probability 1 but with an expected total payoff equal to $+\infty$

$$\mathbb{E}_\rho^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}(\delta\pi) = \int_{I(\rho,\delta)} \sigma_{\mathsf{Min}}^{\Delta}(\rho)(\delta)\Big[\big(t\,\mathsf{wt}(\ell)+\mathsf{wt}(\delta)\big)\mathbb{P}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{\rho\xrightarrow{t,\delta}}(\pi)+\mathbb{E}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}_{\rho\xrightarrow{t,\delta}}(\pi)\Big]\,\mathrm{d}\sigma_{\mathsf{Min}}^{\mathbb{R}_{\geqslant 0}}(\rho,\delta)(t)$$

We then define the expected weight

$$\mathbb{E}_\rho^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}} = \sum_\pi \mathbb{E}_\rho^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}(\pi) \tag{5.3}$$

when this sum converges.

Hypothesis 5.13 is sufficient to show the well-definition of all expectations $\mathbb{E}_\rho^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}(\pi)$, using similar reasons as in the finite case (probabilities exponentially decrease, while total-payoff only linearly increase).

However, the infinite sum in $\mathbb{E}_\rho^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}$ can be problematic, as demonstrated by the following example.

Example 5.16. Consider the weighted timed game of Figure 5.6, where only Min plays, and the memoryless strategy σ_{Min} defined as follows. For all $t \leqslant 1$, $\sigma_{\mathsf{Min}}^{\Delta}(\ell,t)(\delta_0) = t$, $\sigma_{\mathsf{Min}}^{\Delta}(\ell,t)(\delta_1) = 1-t$, and, for all $i \geqslant 2$ and $\delta \in \{\delta_0,\delta_1\}$, $\sigma_{\mathsf{Min}}^{\mathbb{R}_{\geqslant 0}}((\ell,1-1/i),\delta)$ is the Dirac distribution selecting the delay $t = 1/i - 1/(i+1)$. Notice that we can extend the delay distribution (continuously for instance) so that this strategy satisfies Hypothesis 5.13. For all $i \geqslant 1$, there is a unique play conforming to σ_{Min} of length i that reaches the target from configuration $(\ell, \frac{1}{2})$: it has a weight i and a probability $\frac{1}{i+1}\prod_{j=2}^{i}(1-1/j) = 1/[i(i+1)]$. In particular, $\mathbb{P}^{\sigma_{\mathsf{Min}}}_{\ell,1/2}(\mathsf{TPaths}_{\ell,1/2}) = \sum_{i\geqslant 1} 1/[i(i+1)] = 1$. Moreover $\mathbb{E}^{\sigma_{\mathsf{Min}}}_{\ell,1/2}(\delta_0^{i-1}\delta_1) = 1/(i+1)$. The overall expectation would thus be $\mathbb{E}^{\sigma_{\mathsf{Min}}}_{\ell,1/2} = \sum_{i\geqslant 1} 1/(i+1)$, which is a diverging series. This example can easily be adapted to only consider continuous distributions on the delays (instead of Dirac ones).

We thus need a stronger hypothesis to ensure its convergence. As in the untimed setting where we asked for the probability to reach the target to be fulfilled only by Min, we adopt here again an asymmetrical point of view, relying only on hypothesis on the strategy σ_{Min} of Min.

Definition 5.17. A strategy $\sigma_{\mathsf{Min}} \in \mathsf{r}\Sigma_{\mathsf{Min}}$ of Min is said *proper* if for all finite plays ρ and strategies $\sigma_{\mathsf{Max}} \in \mathsf{r}\Sigma_{\mathsf{Max}}$, $\mathbb{P}_\rho^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}(\mathsf{TPaths}_\rho) = 1$ and the infinite sum in (5.3) converges.

We let $\mathsf{r}\Sigma^{\mathsf{p}}_{\mathsf{Min}}$ be the set of proper strategies of Min, $\mathsf{m}\Sigma^{\mathsf{p}}_{\mathsf{Min}}$ the subset of memoryless proper strategies. Notice that a deterministic strategy of Min is proper as soon as it guarantees to reach the target set of locations (remember that we have ruled out configurations with a deterministic value $\mathsf{dVal}(\ell,\nu) = +\infty$ where Min cannot deterministically guarantee to reach the target L_{t}): this shows that proper strategies exist (even without using memory). For stochastic strategies, we have seen above that reaching the target set

of locations with probability 1 is a necessary but not sufficient condition to be proper. Not only we must reach the target almost surely, but we must do it *quickly enough* so that the expectation converges. We now give a sufficient condition for a strategy to be proper:

Hypothesis 5.18. A strategy $\sigma_{\mathsf{Min}} \in \mathsf{r}\Sigma_{\mathsf{Min}}$ of Min satisfies this hypothesis if there exists $m \in \mathbb{N}$ and $\alpha \in (0,1]$ such that for all finite plays ρ and strategies $\sigma_{\mathsf{Max}} \in \mathsf{r}\Sigma_{\mathsf{Max}}$, $\mathbb{P}_\rho^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}(\bigcup_{n\leqslant m} \mathsf{TPaths}_\rho^n) \geqslant \alpha$, where TPaths_ρ^n denotes the subset of TPaths_ρ containing only paths of length n.

This hypothesis can indeed be shown as a sufficient condition for a strategy to be proper.

Now that we have associated an expected payoff to each convenient pair of strategies, we are able to mimic the classical definition of *value* to stochastic strategies. Let ℓ be a location and ν be a valuation. For all $\sigma_{\mathsf{Min}} \in \mathsf{r}\Sigma^{\mathsf{p}}_{\mathsf{Min}}$ and $\sigma_{\mathsf{Max}} \in \mathsf{r}\Sigma_{\mathsf{Max}}$, we let

$$\mathsf{pVal}((\ell,\nu),\sigma_{\mathsf{Min}}) = \sup_{\sigma_{\mathsf{Max}}\in\mathsf{r}\Sigma_{\mathsf{Max}}} \mathbb{E}_{\ell,\nu}^{\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}}}$$

Then, we let

$$\overline{\mathsf{pVal}}(\ell,\nu) = \inf_{\sigma_{\mathsf{Min}}\in\mathsf{r}\Sigma^{\mathsf{p}}_{\mathsf{Min}}} \mathsf{pVal}((\ell,\nu),\sigma_{\mathsf{Min}})$$

once again only focussing on the value of Min. We also define the *memoryless values* $\mathsf{mVal}((\ell,\nu),\sigma_{\mathsf{Min}})$ and $\overline{\mathsf{mVal}}(\ell,\nu)$, where all strategies are taken memoryless.

Our main contribution is to relate these different notions of values, relying on a similar proof schema as in the untimed setting before (that we thus omit in this manuscript). It will thus be crucial to have in our hands ε-optimal deterministic strategies for Min that are switching strategies. Hopefully, we have shown the existence of such for divergent weighted timed games in Theorem 4.42 (where we can moreover check that strategies σ^1_{Min} and σ^2_{Min} can be taken so as fulfilling Hypotheses 5.13 and 5.18), which is why we stick to such divergent games. In this context, memory can thus be fully emulated with stochastic choices, and combining memory and stochastic choices does not bring more power to players, which we summarise by:

Theorem 5.19. *In all divergent weighted timed games and for all configurations* (ℓ,ν)*,*

$$\overline{\mathsf{pVal}}(\ell,\nu) = \mathsf{dVal}(\ell,\nu) = \overline{\mathsf{mVal}}(\ell,\nu)$$

Conclusion

We have studied the tradeoff between memory and randomisation in strategies of finite shortest-path games and weighted timed games, showing that Min guarantees the same value when they are allowed to use both memory and randomisation, or simply one of these two capabilities. In particular, we showed how randomisation can emulate memory in this context, based on the switching strategies, and vice versa. In the untimed setting, we have also studied the existence of optimal memoryless strategies, which turns out to be equivalent to the existence of optimal memoryless deterministic strategies, and testable in polynomial time.

For weighted timed games, one of the main challenge was to formally define the notion of expected payoff in this continuous uncountable system. The result we have obtained

about emulation of memory and randomisation is restricted to divergent weighted timed games, by our need of switching strategies as a technical tool. The next step is to try to extend our study to more general weighted timed games, like the class of almost-divergent ones for instance. We do not know yet if similar ε-optimal switching strategies exist also in this context.

Another question concerns the implementability of the randomised memoryless strategies, that have been shown to suffice to play almost-optimally in divergent weighted timed games: even if they use no memory, they still need to know the precise current clock valuation. In (non-weighted) timed games, previous work [CHP08b] aimed at removing this need for precision, by using stochastic strategies where the delays are chosen with probability distributions that do not require exact knowledge of the clocks measurements. In our setting, we aim at further studying the implementability of the randomised strategies of Min, e.g. by requiring them to be robust against small imprecisions. This crosses the path of some of our other works (not described in this manuscript) about the synthesis of robust controllers in timed automata [Bus+19].

Conclusion

This manuscript summarises some of my contributions in the area of controller synthesis in the untimed and timed setting, with quantitative objectives of shortest-path and total-payoff. The problem is phrased in the framework of zero-sum two player games, where one player models the controller, and the other an uncontrollable, and full antagonistic, environment.

Many of these contributions deal with some algorithms to compute the optimal value of such games, shortest-path games, total-payoff games or weighted timed games. A last chapter also studies various notions of optimal value, when allowing player to use randomisation (instead of memory).

Some perspectives have already been proposed in the end of each chapter. I describe now three additional possible developments, more ambitious and long term, that I have not yet investigated.

Interior-point method and polynomial-time algorithms

One concerns the computation of the optimal values in shortest-path games and divergent weighted timed games. We have explained how to compute such values, with value iteration techniques. The theoretical result we have obtained in the last chapter, relating those values with the ones when restricted to use only randomisation and no memory, could pave the way to new kind of computation (or approximation) algorithms, possibly with better complexity. In particular, we would like to investigate the use of interior-point methods that allow for polynomial-time algorithms solving linear programming, whereas simplex methods, resembling the value iteration paradigm, works with an exponential-time complexity. In our context, interior points could be the non-deterministic randomised strategies, while the corners of the simplex are indeed deterministic strategies: having a way to dynamically improve from a randomised strategy to a better one, in the same way as the interior-point method proposes a way to move in the interior of the set of valid points of the linear program, could enable new types of algorithms. Knowing the proximity of shortest-path games with mean-payoff games (and thus parity games), for which the search for polynomial-time algorithms is a long-standing open question, motivates the research in this direction.

Evolutionary game theory on graphs

Another perspective consists in drawing parallels between shortest-path or total-payoff games, possibly in the presence of time, and the realm of *evolutionary game theory*. This theory has been successfully applied in the context of biology or economics, where evolving populations with a mix of strategies compete. Most of the work has been performed in matrix games (and not games on graphs). Such dynamical aspects are crucial in this setting, and in contrast, most of the work in computer science has been focused on more

static and classical game theory on graphs, for controller synthesis purposes in particular as has been studied in this manuscript. Bridging the gap between evolutionary game theory and games on graphs seems thus necessary to propose new techniques for both communities. We have initiated the search in this direction in [Bri+19] by allowing players to repeatedly update their respective strategies (for instance, to improve the payoff with respect to the current strategy profile). This generates a dynamics in the game which may eventually stabilise to an equilibrium. We have used simulation relations and graph minors in order to characterise the termination of such dynamics, applying our techniques to the context of interdomain routing problems. Yet much still needs to be done, which requires understanding better the structure of strategies in games on graphs, that will be affected along the dynamics.

Robust strategies in weighted timed games

We have seen in this manuscript some examples where cumbersome behaviours, like Zeno convergence of delays, are necessary for players to play optimally. However we could claim that this is not a realistic behaviour in the real world, and thus try to study what happens when they are forbidden. This is strongly related with the search for robust controllers in timed automata (as we have studied in [Bus+19] for instance). Connecting these two faces of the prism of my work is a tempting direction of research that will certainly be very fruitful. The study of randomisation in strategies was one of the preliminary steps towards this, since it allows the players to blur the lines, and thus play in a more robust fashion. This was however preliminary work, since, for now, players may still use the precise information of the clock valuations. We would thus like to make this knowledge less precise, and see whether players can adapt their strategy, while not losing too much with respect to the optimal value.

Bibliography

[ABM04] 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)*. Vol. 3142. Lecture Notes in Computer Science. Springer, 2004, pp. 122–133 (cit. on pp. 6, 7, 34, 36, 70, 71, 73, 74, 77, 85, 93).

[AD94] Rajeev Alur and David L. Dill. "A Theory of Timed Automata". In: *Theoretical Computer Science* 126.2 (1994), pp. 183–235 (cit. on pp. 6, 34, 40, 42, 81).

[ALP04] Rajeev Alur, Salvatore La Torre, and George J. Pappas. "Optimal Paths in Weighted Timed Automata". In: *Theoretical Computer Science* 318.3 (2004), pp. 297–322 (cit. on p. 6).

[AM99] Eugene Asarin and Oded Maler. "As Soon as Possible: Time Optimal Control for Timed Automata". In: *Hybrid Systems: Computation and Control*. Vol. 1569. Lecture Notes in Computer Science. Springer, 1999, pp. 19–30 (cit. on pp. 6, 42).

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008 (cit. on pp. 98, 100).

[Beh+01] 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)*. Vol. 2034. Lecture Notes in Computer Science. Springer, 2001, pp. 147–161 (cit. on pp. 6, 36, 37, 43, 44).

[Bér+98] Béatrice Bérard, Antoine Petit, Volker Diekert, and Paul Gastin. "Characterization of the Expressive Power of Silent Transitions in Timed Automata". In: *Fundamenta Informaticae* 36.2-3 (1998), pp. 145–182 (cit. on p. 36).

[BCJ09] Jasper Berendsen, Taolue Chen, and David N. Jansen. "Undecidability of Cost-Bounded Reachability in Priced Probabilistic Timed Automata". In: *Proceedings of the 6th Annual Conference on Theory and Applications of Models of Computation (TAMC'09)*. Vol. 5532. Lecture Notes in Computer Science. Springer, 2009, pp. 128–137 (cit. on p. 8).

[Ber+18] Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, and Pierre P. Carlier. "When are stochastic transition systems tameable?" In: *Journal of Logical and Algebraic Methods in Programming* 99 (2018), pp. 41–96 (cit. on p. 104).

[Ber+14] Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, Quentin Menet, Christel Baier, Marcus Größer, and Marcin Jurdzinski. "Stochastic Timed Automata". In: *Log. Methods Comput. Sci.* 10.4 (2014) (cit. on pp. 95, 104–107).

[BT91] Dimitri P. Bertsekas and John N. Tsitsiklis. "An Analysis of Stochastic Shortest Path Problems". In: *Math. Oper. Res.* 16.3 (1991), pp. 580–595 (cit. on pp. 95, 98).

[BV07] Henrik Björklund and Sergei Vorobyov. "A Combinatorial Strongly Subexponential Strategy Improvement Algorithm for Mean Payoff Games". In: *Discrete Applied Mathematics* 155 (2007), pp. 210–229 (cit. on pp. 10, 12).

[Bou15] Patricia Bouyer. "On the Optimal Reachability Problem in Weighted Timed Automata and Games". In: *Proceedings of the 7th Workshop on Non-Classical Models of Automata and Applications (NCMA'15)*. Vol. 318. books@ocg.at. Austrian Computer Society, 2015, pp. 11–36 (cit. on p. 46).

[Bou+07] Patricia Bouyer, Thomas Brihaye, Véronique Bruyère, and Jean-François Raskin. "On the Optimal Reachability Problem of Weighted Timed Automata". In: *Formal Methods in System Design* 31.2 (2007), pp. 135–175 (cit. on pp. 6, 43).

[Bou+16] Patricia Bouyer, Thomas Brihaye, Pierre Carlier, and Quentin Menet. "Compositional Design of Stochastic Timed Automata". In: *Proceedings of the International Computer Science Symposium in Russia (CSR 2016)*. Vol. 9691. Lecture Notes in Computer Science. Springer, 2016. DOI: 10.1007/978-3-319-34171-2_9 (cit. on p. 104).

[BBM06] Patricia Bouyer, Thomas Brihaye, and Nicolas Markey. "Improved Undecidability Results on Weighted Timed Automata". In: *Information Processing Letters* 98.5 (2006), pp. 188–194 (cit. on p. 39).

[Bou+20] Patricia Bouyer, Thomas Brihaye, Mickael Randour, Cédric Rivière, and Pierre Vandenhove. "Decisiveness of Stochastic Systems and its Application to Hybrid Models". In: *Proceedings of the 11th International Symposium on Games, Automata, Logics, and Formal Verification (GandALF'20)*. Ed. by Davide Bresolin and Jean-François Raskin. Vol. 326. Electronic Proceedings in Theoretical Computer Science. 2020 (cit. on p. 104).

[BBL08] Patricia Bouyer, Ed Brinksma, and Kim G. Larsen. "Optimal Infinite Scheduling for Multi-Priced Timed Automata". In: *Formal Methods in System Design* 32.1 (2008), pp. 3–23 (cit. on p. 43).

[Bou+04a] 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)*. Vol. 3328. Lecture Notes in Computer Science. Springer, 2004, pp. 148–160 (cit. on pp. 6, 7).

[Bou+04b] 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)*. Vol. 3328. LNCS. Springer, 2004, pp. 148–160 (cit. on pp. 12, 34, 36, 37, 70, 71, 74, 93).

[BCM16] Patricia Bouyer, Maximilien Colange, and Nicolas Markey. "Symbolic Optimal Reachability in Weighted Timed Automata". In: *Proceedings of the 28th International Conference on Computer Aided Verification (CAV'16)*. Vol. 9779. Lecture Notes in Computer Science. Springer, 2016, pp. 513–530 (cit. on p. 44).

[BJM15] 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)*. Vol. 42. Leibniz International Proceedings in Informatics. Leibniz-Zentrum für Informatik, 2015, pp. 311–324. DOI: 10.4230/LIPIcs.CONCUR.2015.311 (cit. on pp. 7, 36, 39, 74–76, 83, 86, 90, 93).

[Bou+06] Patricia Bouyer, Kim G. Larsen, Nicolas Markey, and Jacob Illum Rasmussen. "Almost Optimal Strategies in One-Clock Priced Timed Games". In: *Proceedings of the 26th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'06)*. Vol. 4337. Lecture Notes in Computer Science. Springer, 2006, pp. 345–356 (cit. on pp. 6, 46, 47, 59, 64–66).

[BCR14] 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)*. ACM, 2014, pp. 283–292 (cit. on pp. 66, 68).

[Bri+13] T. Brihaye, L. Doyen, G. Geeraerts, J. Ouaknine, J.-F. Raskin, and J. Worrell. "Time-Bounded Reachability for Monotonic Hybrid Automata: Complexity and Fixed Points". In: *Automated Technology for Verification and Analysis*. Vol. 8172. LNCS. Springer, 2013, pp. 55–70 (cit. on p. 39).

[BBR05] 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)*. Vol. 3829. LNCS. Springer, 2005, pp. 49–64 (cit. on pp. 6, 39).

[BDS13] Thomas Brihaye, Julie De Pril, and Sven Schewe. "Multiplayer Cost Games with Simple Nash Equilibria". In: *Proceedings of the International Symposium on Logical Foundations of Computer Science (LFCS'13)*. Vol. 7734. Lecture Notes in Computer Science. Springer, 2013, pp. 59–73 (cit. on p. 33).

[Bri+16a] Thomas Brihaye, Amit Kumar Dhar, Gilles Geeraerts, Axel Haddad, and Benjamin Monmege. "Efficient Energy Distribution in a Smart Grid Using Multi-Player Games". In: *Proceedings of the Cassting Workshop on Games for the Synthesis of Complex Systems (Cassting'16) and the 3rd International Workshop on Synthesis of Complex Parameters (SynCoP'16)*. Vol. 220. Eindhoven,

Netherlands: EPTCS, Apr. 2016, pp. 1–12. DOI: 10.4204/EPTCS.220.1 (cit. on pp. 9, 12, 33).

[Bri+16b] Thomas Brihaye, Morgane Estiévenart, Gilles Geeraerts, Hsi-Ming Ho, Benjamin Monmege, and Nathalie Sznajder. "Real-Time Synthesis is Hard!" In: *Proceedings of the 14th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'16)*. Ed. by Martin Fränzle and Nicolas Markey. Vol. 9884. Lecture Notes in Computer Science. Quebec city, Canada: Springer, Aug. 2016, pp. 105–120. DOI: 10.1007/978-3-319-44878-7_7 (cit. on p. 8).

[Bri+15a] 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)*. Ed. by Prahladh Harsha and G. Ramalingam. Vol. 45. Leibniz International Proceedings in Informatics (LIPIcs). Bangalore, India: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dec. 2015, pp. 278–292. DOI: 10.4230/LIPIcs.FSTTCS.2015.278 (cit. on pp. 34, 46, 50).

[Bri+21] Thomas Brihaye, Gilles Geeraerts, Axel Haddad, Engel Lefaucheux, and Benjamin Monmege. "One-Clock Priced Timed Games with Arbitrary Weights". Submitted. 2021 (cit. on pp. 9, 15, 34).

[Bri+15b] 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)*. Ed. by Luca Aceto and David de Frutos Escrig. Vol. 42. LIPIcs. Madrid, Spain: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Sept. 2015, pp. 297–310. DOI: 10.4230/LIPIcs.CONCUR.2015.297 (cit. on pp. 9, 11).

[Bri+17a] Thomas Brihaye, Gilles Geeraerts, Axel Haddad, and Benjamin Monmege. "Pseudopolynomial Iterative Algorithm to Solve Total-Payoff Games and Min-Cost Reachability Games". In: *Acta Informatica* 54.1 (Feb. 2017), pp. 85–125. DOI: 10.1007/s00236-016-0276-z (cit. on pp. 9, 11, 14).

[Bri+15c] Thomas Brihaye, Gilles Geeraerts, Axel Haddad, Benjamin Monmege, Guillermo A. Pérez, and Gabriel Renault. "Quantitative Games under Failures". In: *Proceedings of the 35th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'15)*. Ed. by Prahladh Harsha and G. Ramalingam. Vol. 45. Leibniz International Proceedings in Informatics (LIPIcs). Bangalore, India: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dec. 2015, pp. 293–306. DOI: 10.4230/LIPIcs.FSTTCS.2015.293 (cit. on p. 8).

[Bri+19] Thomas Brihaye, Gilles Geeraerts, Marion Hallet, Benjamin Monmege, and Bruno Quoitin. "Dynamics on Games: Simulation-Based Techniques and Applications to Routing". In: *Proceedings of the 39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'19)*. Ed. by Arkadev Chattopadhyay and Paul Gastin.

Vol. 150. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Dec. 2019, 35:1–35:14. DOI: 10.4230/LIPIcs.FSTTCS.2019.35 (cit. on pp. 8, 113).

[Bri+17b] Thomas Brihaye, Gilles Geeraerts, Hsi-Ming Ho, and Benjamin Monmege. "MightyL: A Compositional Translation from MITL to Timed Automata". In: *Proceedings of the 29th International Conference on Computer Aided Verification, Part I (CAV'17)*. Ed. by Rupak Majumdar and Viktor Kunčak. Vol. 10426. Lecture Notes in Computer Science. Heidelberg, Germany: Springer, July 2017, pp. 421–440. DOI: 10.1007/978-3-319-63387-9_21 (cit. on p. 8).

[Bri+17c] Thomas Brihaye, Gilles Geeraerts, Hsi-Ming Ho, and Benjamin Monmege. "Timed-Automata-Based Verification of MITL over Signals". In: *Proceedings of the 24th International Symposium on Temporal Representation and Reasoning (TIME'17)*. Ed. by Sven Schewe, Thomas Schneider, and Jef Wijsen. Vol. 90. LIPIcs. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Oct. 2017, 7:1–7:19. DOI: 10.4230/LIPIcs.TIME.2017.7 (cit. on p. 8).

[Bri+14] 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)*. Ed. by Paolo Baldan and Daniele Gorla. Vol. 8704. Lecture Notes in Computer Science. Roma, Italy: Springer, Sept. 2014, pp. 560–575. DOI: 10.1007/978-3-662-44584-6_38 (cit. on pp. 6, 34, 39, 46, 51).

[Bri+18] Thomas Brihaye, Arthur Milchior, Gilles Geeraerts, Hsi-Ming Ho, and Benjamin Monmege. "Efficient algorithms and tools for MITL model-checking and synthesis". In: *Proceedings of the 23rd International Conference on Engineering of Complex Computer Systems (ICECCS'18)*. CPS, Dec. 2018, pp. 180–184. DOI: 10.1109/ICECCS2018.2018.00027 (cit. on p. 8).

[Bri+11] Luboš Brim, Jakub Chaloupka, Laurent Doyen, Rafaella Gentilini, and Jean-François Raskin. "Faster Algorithms for Mean-Payoff Games". In: *Formal Methods for System Design* 38.2 (2011), pp. 97–118 (cit. on p. 10).

[Bus19] Damien Busatto-Gaston. "Symbolic controller synthesis for timed systems: robustness and optimality". Theses. Aix Marseille Université, Dec. 2019. URL: https://hal.archives-ouvertes.fr/tel-02436831 (cit. on p. 85).

[BMR17] Damien Busatto-Gaston, Benjamin Monmege, and Pierre-Alain Reynier. "Optimal Reachability in Divergent Weighted Timed Games". In: *Proceedings of the 20th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'17)*. Ed. by Javier Esparza and Andrzej S. Murawski. Vol. 10203. Lecture Notes in Computer Science. Uppsala, Sweden: Springer, Apr. 2017, pp. 162–178. DOI: 10.1007/978-3-662-54458-7_10 (cit. on pp. 9, 11, 70).

[BMR18] Damien Busatto-Gaston, Benjamin Monmege, and Pierre-Alain Reynier. "Symbolic Approximation of Weighted Timed Games". In: *Proceedings of the 38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'18)*. Ed. by Sumit Ganguly and Paritosh Pandya. Vol. 122. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dec. 2018, 28:1–28:16. DOI: 10.4230/LIPIcs.FSTTCS.2018.28 (cit. on p. 70).

[BMR21] Damien Busatto-Gaston, Benjamin Monmege, and Pierre-Alain Reynier. "Optimal Controller Synthesis for Timed Systems". Submitted to Logical Methods in Computer Science. Mar. 2021 (cit. on pp. 32, 34, 70).

[Bus+19] Damien Busatto-Gaston, Benjamin Monmege, Pierre-Alain Reynier, and Ocan Sankur. "Robust Controller Synthesis in Timed Büchi Automata: A Symbolic Approach". In: *31st International Conference on Computer Aided Verification (CAV 2019)*. Ed. by Isil Dillig and Serdar Tasiran. Vol. 11561. Lecture Notes in Computer Science. Springer, July 2019, pp. 572–590. DOI: 10.1007/978-3-030-25540-4_33 (cit. on pp. 8, 111, 113).

[CAH04] 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. Washington, DC, USA: IEEE Computer Society, 2004, pp. 206–217. ISBN: 0-7695-2185-1 (cit. on pp. 7, 94).

[CHJ05] Krishnendu Chatterjee, Thomas A. Henzinger, and Marcin Jurdziński. "Mean-Payoff Parity Games". In: *Proceedings of the 20th Annual Symposium on Logic in Computer Science (LICS'05)*. IEEE Computer Society Press, 2005, pp. 178–187 (cit. on p. 6).

[CHP08a] 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*. 2008, pp. 87–100 (cit. on pp. 7, 94).

[CHP08b] Krishnendu Chatterjee, Thomas A. Henzinger, and Vinayak S. Prabhu. "Trading Infinite Memory for Uniform Randomness in Timed Games". In: *Hybrid Systems: Computation and Control*. Ed. by Magnus Egerstedt and Bud Mishra. Springer, 2008, pp. 87–100 (cit. on p. 111).

[CRR14] Krishnendu Chatterjee, Mickael Randour, and Jean-François Raskin. "Strategy Synthesis for Multi-Dimensional Quantitative Objectives". In: *Acta Informatica* 51 (2014), pp. 129–163. DOI: https://doi.org/10.1007/s00236-013-0182-6 (cit. on pp. 7, 94).

[Che+13] Taolue Chen, Vojtěch Forejt, Marta Kwiatkowska, David Parker, and Aistis Simaitis. "Automatic Verification of Competitive Stochastic Systems". In: *Formal Methods in System Design* 43.1 (2013), pp. 61–92 (cit. on pp. 10, 27).

[CR15] Carlo Comin and Romeo Rizzi. *An Improved Pseudo-Polynomial Upper Bound for the Value Problem and Optimal Strategy Synthesis in Mean Payoff Games*. Tech. rep. 1503.04426. arXiv, 2015 (cit. on p. 10).

[EM79] Andrzej Ehrenfeucht and Jan Mycielski. "Positional Strategies for Mean Payoff Games". In: *International Journal of Game Theory* 8.2 (1979), pp. 109–113 (cit. on p. 9).

[FIS20] John Fearnley, Rasmus Ibsen-Jensen, and Rahul Savani. "One-Clock Priced Timed Games are PSPACE-hard". In: *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Sciences (LICS'20)*. ACM, 2020, pp. 397–409. DOI: 10.1145/3373718.3394772 (cit. on p. 46).

[FGR12] Emmanuel Filiot, Rafaella Gentilini, and Jean-François Raskin. "Quantitative Languages Defined by Functional Automata". In: *Proceedings of the 23rd International Conference on Concurrency theory (CONCUR'12)*. Vol. 7454. Lecture Notes in Computer Science. Springer, 2012, pp. 132–146 (cit. on p. 12).

[GS09] Thomas Martin Gawlitza and Helmut Seidl. "Games through Nested Fixpoints". In: *Proceedings of the 21st International Conference on Computer Aided Verification (CAV'09)*. Vol. 5643. Lecture Notes in Computer Science. Springer, 2009, pp. 291–305 (cit. on pp. 10, 16).

[GZ04] Hugo Gimbert and Wiesław Zielonka. "When Can You Play Positionally?" In: *Proceedings of the 29th International Conference on Mathematical Foundations of Computer Science (MFCS'04)*. Vol. 3153. Lecture Notes in Computer Science. Springer, 2004, pp. 686–698 (cit. on pp. 5, 9, 10, 15, 16).

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke. *Automata, Logics, and Infinite Games: A Guide to Current Research*. Vol. 2500. Lecture Notes in Computer Science. Springer, 2002 (cit. on p. 5).

[HM15] Axel Haddad and Benjamin Monmege. *Why Value Iteration Runs in Pseudo-Polynomial Time for Discounted-Payoff Games*. Technical note. Université libre de Bruxelles, June 2015 (cit. on p. 8).

[HM18] Serge Haddad and Benjamin Monmege. "Interval Iteration Algorithm for MDPs and IMDPs". In: *Theoretical Computer Science* 735 (July 2018), pp. 111–131. DOI: 10.1016/j.tcs.2016.12.003 (cit. on p. 8).

[HIM13] 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)*. Vol. 8052. Lecture Notes in Computer Science. Springer, 2013, pp. 531–545 (cit. on pp. 6, 46, 47, 65, 69).

[Imm81] Neil Immerman. "Number of Quantifiers is Better Than Number of Tape Cells". In: *Journal of Computer and System Sciences* 22.3 (1981), pp. 384–406 (cit. on p. 29).

[Imm88] Neil Immerman. "Nondeterministic Space is Closed Under Complementation". In: *SIAM Journal on Computing* 17 (1988), pp. 935–938 (cit. on pp. 32, 81).

[JT07] 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)*. Vol. 4596. LNCS. Springer, 2007, pp. 838–849 (cit. on pp. 6, 42).

[Kha+08] 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". In: *Theory of Computing Systems* 43 (2008), pp. 204–233 (cit. on pp. 6, 12).

[Kli+12] Miroslav Klimoš, Kim G. Larsen, Filip Štefaňák, and Jeppe Thaarup. "Nash Equilibria in Concurrent Priced Games". In: *Proceedings of the 6th international conference on Language and Automata Theory and Applications (LATA'12)*. Vol. 7183. Lecture Notes in Computer Science. Springer, 2012, pp. 363–376 (cit. on p. 33).

[LMS04] François Laroussinie, Nicolas Markey, and Philippe Schnoebelen. "Model Checking Timed Automata with One or Two Clocks". In: *Proceedings of CONCUR'04*. 2004, pp. 387–401 (cit. on p. 48).

[Lar+01] Kim G. Larsen, Gerd Behrmann, Ed Brinksma, Ansgar Fehnker, Thomas Hune, Paul Pettersson, and Judi Romijn. "As cheap as possible: Efficient cost-optimal reachability for priced timed automata". In: *Proceedings of the 13th International Conference on Computer Aided Verification (CAV'01)*. Vol. 2102. Lecture Notes in Computer Science. Springer, 2001, pp. 493–505 (cit. on pp. 43, 44).

[Lef15] Engel Lefaucheux. "Negative Prices for Priced Timed Games". Internship report. 2015 (cit. on p. 51).

[LMT19] Théodore Lopez, Benjamin Monmege, and Jean-Marc Talbot. "Determinisation of Finitely-Ambiguous Copyless Cost Register Automata". In: *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*. Ed. by Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen. Vol. 138. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Sept. 2019, 75:1–75:15. DOI: 10.4230/LIPIcs.MFCS.2019.75 (cit. on p. 8).

[Mar75] Donald A. Martin. "Borel Determinacy". In: *Annals of Mathematics* 102.2 (1975), pp. 363–371 (cit. on p. 15).

[Mar98] Donald A. Martin. "The Determinacy of Blackwell Games". In: *The Journal of Symbolic Logic* 63.4 (1998), pp. 1565–1581 (cit. on p. 97).

[Mat02] Jiri Matousek. *Lectures on Discrete Geometry*. Berlin, Heidelberg: Springer-Verlag, 2002. ISBN: 0387953744 (cit. on p. 73).

[MPR20] Benjamin Monmege, Julie Parreaux, and Pierre-Alain Reynier. "Reaching Your Goal Optimally by Playing at Random with No Memory". In: *Proceedings of the 31st International Conference on Concurrency Theory (CONCUR 2020)*. Ed. by Igor Konnov and Laura Kovács. Vol. 171. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Sept. 2020, 26:1–26:21. DOI: 10.4230/LIPIcs.CONCUR.2020.26 (cit. on p. 94).

[MPR21] Benjamin Monmege, Julie Parreaux, and Pierre-Alain Reynier. "Playing Stochastically in Weighted Timed Games to Emulate Memory". In: *ICALP'21*. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2021 (cit. on pp. 70, 94).

[Nas50] John F. Nash. "Equilibrium Points in n-Person Games". In: *Proceedings of the National Academy of Sciences of the United States of America* 36.1 (1950), pp. 48–49 (cit. on p. 94).

[OW10] J. Ouaknine and J. Worrell. "Towards a Theory of Time-Bounded Verification". In: *Automata, Languages and Programming*. Vol. 6199. LNCS. Springer, 2010, pp. 22–37 (cit. on p. 39).

[Pur00] Anuj Puri. "Dynamical Properties of Timed Automata". In: *Discrete Event Dynamic Systems* 10.1-2 (2000), pp. 87–113 (cit. on p. 45).

[Put94] Martin L. Puterman. *Markov Decision Processes*. New York, NY: John Wiley & Sons, Inc., 1994 (cit. on p. 17).

[Rut11] 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)*. Vol. 57. Electronic Proceedings in Theoretical Computer Science. 2011, pp. 31–46 (cit. on pp. 6, 46, 47, 59, 64, 65, 69).

[Sav70] Walter J. Savitch. "Relationships Between Nondeterministic and Deterministic Tape Complexities". In: *Journal of Computer and System Sciences* 4.2 (1970), pp. 177–192 (cit. on p. 81).

[Str66] Ralph E. Strauch. "Negative Dynamic Programming". In: *The Annals of Mathematical Statistics* 37 (1966), pp. 871–890 (cit. on p. 17).

[Sze88] Róbert Szelepcsényi. "The Method of Forced Enumeration for Nondeterministic Automata". In: *Acta Informatica* 26.3 (1988), pp. 279–284 (cit. on pp. 32, 81).

[Tar72] Robert E. Tarjan. "Depth First Search and Linear Graph Algorithms". In: *SIAM Journal on Computing* 1.2 (1972), pp. 146–160 (cit. on p. 28).

[Tho95] Wolfgang Thomas. "On the Synthesis of Strategies in Infinite Games". In: *Symposium on Theoretical Aspects of Computer Science (STACS'95)*. Vol. 900. Lecture Notes in Computer Science. Springer, 1995, pp. 1–13 (cit. on p. 18).

[TV87] F. Thuijsman and O. J. Vrieze. "The bad match; a total reward stochastic game". In: *Operations-Research-Spektrum* 9.2 (1987), pp. 93–99. DOI: 10.1007/BF01732644 (cit. on p. 10).

[ZP96] Uri Zwick and Michael S. Paterson. "The Complexity of Mean Payoff Games". In: *Theoretical Computer Science* 158 (1996), pp. 343–359 (cit. on pp. 9, 15–17).