Average Reward Timed Games[*]

Bo Adler[1], Luca de Alfaro[1], and Marco Faella[1,2]

[1] School of Engineering, Universitity of California, Santa Cruz, USA
[2] Dipartimento di Scienze Fisiche, Università di Napoli "Federico II", Italy

Technical Report UCSC-CRL-05-03
University of California, Santa Cruz

Abstract. We consider real-time games where the goal consists, for each player, in maximizing the average amount of reward he or she receives per time unit. We consider zero-sum rewards, so that a reward of $+r$ to one player corresponds to a reward of $-r$ to the other player. The games are played on discrete-time game structures which can be specified using a two-player version of timed automata whose locations are labeled by rewards. Even though the rewards themselves are zero-sum, the games are not, due to the requirement that time must progress along a play of the game.
Since we focus on control applications, we define the value of the game to a player to be the maximal average reward per time unit that the player can ensure. We show that in general the values to players 1 and 2 do not sum to zero. We provide algorithms for computing the value of the game for either player; the algorithms are based on the relationship between the original, infinite-round, game, and a derived game that is played for only finitely many rounds. As positional optimal strategies exist for both players in both games, we show that the problem of computing the value of the game is in NP$\cap$coNP.

1 Introduction

Games provide a setting for the study of control problems. It is natural to view a system and its controller as two players in a game; the problem of synthesizing a controller given a control goal can be phrased as the problem of finding a controller strategy that enforces the goal, regardless of how the system behaves [Chu63,RW89,PR89]. In the control of real-time systems, the games must not only model the interaction steps between the system and the controller, but also the amount of time that elapses between these steps. This leads to *timed games*, a model that was first applied to the synthesis of controllers for safety, reachability, and other ω-regular goals. The aim was the synthesis of controllers that guarantee the *correctness* of the behavior of the resulting controlled system [MPS95,AH97,AMAS98,HHM99,dAFH$^+$03].

More recently, the problem of designing *efficient* controllers for real-time systems has been addressed, via the consideration of *priced* versions of timed games [BCFL04,ABM04]. In such priced versions of timed games, price rates (or, symmetrically, reward rates) are associated with the states of the game, and prices (or rewards) with its transitions. The problem that has so far been addressed is the synthesis of

[*] This research was partly supported by the NSF CAREER grant CCR-0132780, by the ONR grant N00014-02-1-0671, and by an ARP award.

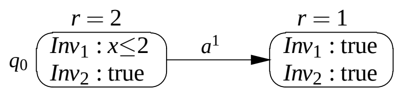

Fig. 1. A game automaton where player 1 can freeze time to achieve a higher average reward.

minimum-cost controllers for reachability goals [BCFL04,ABM04]. In this paper, we focus instead on the problem of synthesizing controllers that maximize the average reward[1] per time unit accrued along an infinite play of the game. This is an expressive and widely applicable efficiency goal, since many real-time systems are modeled as non-terminating systems which exhibit infinite behaviors. Moreover, the synthesis of controllers that maximize the long-run average reward is a natural complement to the synthesis of controllers for safety goals. In fact, a controller for a safety goal leaves the system free to roam in a controllable and safe region of its state space. By solving a control problem for long-run average reward restricted to this controllable, safe region, we can design controllers that are both safe and efficient.

We consider timed games played between two players over *discrete-time game structures* with finite state space. At each round, both players independently choose a move. We distinguish between *immediate moves*, which correspond to control actions or system transitions and take 0 time, and *timed moves*. There are two timed moves: the move Δ_0, which signifies the intention to wait for 0 time, and the move Δ_1, which signifies the intention of waiting for 1 time unit. While the move Δ_0 is always available, the move Δ_1 may not be: if unavailable, the player is forced to either stutter (move Δ_0), or take an immediate move. The two moves chosen by the players jointly determine the successor state: roughly, immediate moves take the precedence over timed ones, and unit-length time steps occur only when both players play Δ_1. These game structures can be specified using a notation similar to that of timed automata. Each location is labeled by two invariants (rather than one), which specify how long the two players can stay at the location. The actions labeling the edges correspond to immediate moves, and each one of them belongs to one of the two players.

With each state of a discrete-time game structure is associated a reward rate, which specifies the reward obtained when staying at the state for one time unit. We consider zero-sum rewards, so that a reward of $+r$ to one player corresponds to a reward of $-r$ to the other player. The goal of a player in the game is to maximize the long-run average reward it receives per time unit. However, the goal of maximizing the reward is subordinate to the requirement that players should not "block" the progress of time by playing forever zero-delay moves (immediate moves, or Δ_0). As an example, consider the game of Figure 1. The strategy that maximizes the reward per time unit calls for player 1 staying forever at q_0: this yields an average reward per time unit of 4. However, such a strategy would block time, since the clock x would not be able to increase beyond the value 2, due to the player-1 invariant $x \leq 2$ at q_0. If player 1 plays move a^1, time can progress, but the average reward per time unit is 1.

To prevent players from blocking time in their pursuit of higher average reward, we define the value of a play of the game in a way that enforces time progress. If time diverges along the play, the value of the play is the average reward per time unit

[1] With a sign change, this is obviously equivalent to minimizing the average cost.

obtained along it. If time does not diverge along the play, there are two cases. If a player contributes to blocking the progress of time, then the value of the play to the player is $-\infty$; if the progress of time is blocked entirely by the other player, then the value of the play to the player is $+\infty$. These definitions are based on the treatment of time divergence in timed games of [dAFH$^+$03,dAHS02]. Thus, even though the reward rate is zero-sum, and time-divergent plays have zero-sum values (if the average reward is $+\bar{r}$ for one player, it is $-\bar{r}$ for the other one), the games are not zero-sum, due to the treatment of time divergence. Since we are interested in the problem of controller design, we define the value of a game to a player to be the maximal play value that the player is able to secure, regardless of how the adversary plays. Perhaps unsurprisingly, these games are not determined, in the sense that the values that the two players can secure do not sum to zero. We show that this is intrinsic in the nature of the games: there is no formulation that can at the same time enforce time progress, and lead to a determined setting.

We provide algorithms for computing the value of the game for either player. The algorithms are based on the relationship between the original, infinite-round, game, and a derived game that is played on the same discrete-time game structure, but for only finitely many rounds. As in [EM79], the derived game terminates whenever one of the two players closes a loop; our construction however differs from [EM79] in how it assigns a value to the loops, due to our different notion of value of a play. We show that a player can achieve the same value in the finite game, as in the original infinite-round game. Our proof is inspired by the argument in [EM79], but it is more complex, as our games are not zero-sum; the proof also closes some small gaps in the proof of [EM79].

The equivalence between finite and infinite games provides a PSPACE algorithm for computing the value of average reward discrete-time games. We improve this result by showing that both finite and infinite games admit positional optimal strategies for each player. Once we fix a positional strategy for a player, the game is reduced to a graph. We provide a polynomial-time algorithm that enables the computaton of the value of the graph for the other player. The algorithm is based on polynomial-time graph transformations, followed by the application of Karp's algorithm for computing the minimum/maximal average cost of a cycle [Kar78]. The positional strategies, and this algorithm, provide us with a polynomial witness, and with a polynomial-time algorithm for checking the witness. Since this analysis can be done both for the winning strategies of a player, and for the "spoiling" strategies of the opponent, we conclude that the problem of computing the value of an average-reward timed game, for both players, is in NP$\cap$coNP. This matches the best known bounds for several other classes of games, among which are turn-based deterministic parity games [EJ91] and turn-based stochastic reachability games [Con92].

Compared to other work on priced timed games [BCFL04,ABM04], our models for timed games are simplified in two ways. First, rewards can only be accrued by staying at a state, and not by taking transitions. Second, we study the problem in discrete time. On the other hand, our models are more general in that unlike [BCFL04,ABM04] we do not impose structural constraints on the game structures that ensure the progress of time. There is a tradeoff between imposing structural constraints and allowing rewards for transitions: had we introduced constraints that ensure time progress, we could have easily accommodated for rewards on the transitions. The restriction to discrete-time limits somewhat the expressiveness of the models. Nevertheless, control problems where the control actions can be issued only at discrete points in time are very common: most real controllers are driven by a periodic clock; hence, the discrete-time restriction is not unduly limiting as far as the controller actions are concerned. We note that there are also many cases where also the system actions can be considered to occur in discrete-time: this is the case, for instance, whenever the state of the controller is also sampled regularly in time.

2 Discrete-Time Game Structures

We define *discrete-time game structures* as a discrete-time version of the timed game structures of [dAFH$^+$03]. A discrete-time game structure represents a game between two players, which we denote by 1, 2; we indicate by $\sim i$ the opponent of $i \in \{1,2\}$ (that is, player $3-i$). A *discrete-time game structure* is a tuple $\mathscr{G} = (S, Acts_1, Acts_2, \Gamma_1, \Gamma_2, \delta, r)$, where:

- S is a finite set of states.
- $Acts_1$ and $Acts_2$ are two disjoint sets of actions for player 1 and player 2, respectively. We assume that $\Delta_0, \Delta_1 \notin Acts_i$ and write $M_i = Acts_i \cup \{\Delta_0, \Delta_1\}$ for the sets of moves of player $i \in \{0,1\}$.
- For $i = 1,2$, the function $\Gamma_i : S \mapsto 2^{M_i} \setminus \emptyset$ is an enabling condition, which assigns to each state s a set $\Gamma_i(s)$ of moves available to player i in that state.
- $\delta : S \times (M_1 \cup M_2) \mapsto S$ is a destination function that, given a state and a move of either player, determines the next state in the game.
- $r : S \mapsto \mathbb{Z}$ is a function that associates with each state $s \in S$ the *reward rate* of s: this is the reward that player 1 earns for staying for one time unit at s.

The move Δ_0 represents a stuttering move that takes 0 time. We require that Δ_0 is always enabled: for $s \in S$ and $i \in \{1,2\}$, we have $\Delta_0 \in \Gamma_i(s)$. When taken, the move Δ_0 does not cause a state change: for all $s \in S$, we have $\delta(s, \Delta_0) = s$. The moves in $\{\Delta_0\} \cup Acts_1 \cup Acts_2$ are known as the *zero-time* moves. The move Δ_1 represents the decision of waiting for 1 time units. We do not require that Δ_1 be always enabled: if we have $\Delta_1 \notin \Gamma_i(s)$ for player $i \in \{1,2\}$ at a state $s \in S$, then player i cannot wait, but must immediately play a zero-time move. We define the *size* of a discrete-time game structure by $|\mathscr{G}| = \sum_{s \in S}(|\Gamma_1(s) + \Gamma_2(s))$.

2.1 Move Outcomes and Runs

A timed game proceeds as follows. At each state $s \in S$, player 1 chooses a move $a^1 \in \Gamma_1(s)$, and simultaneously and independently, player 2 chooses a move $a^2 \in \Gamma_2(s)$. The successor state $\widetilde{\delta}(s, a^1, a^2)$ is then determined according to the following rules.

- *Actions take precedence over stutter steps and time steps.* If $a^1 \in Acts_1$ or $a^2 \in Acts_2$, then the game takes an action a selected nondeterministically from $\{a^1, a^2\} \cap (Acts_1 \cup Acts_2)$, and the game proceeds to location $\widetilde{\delta}(s, a^1, a^2) \stackrel{\text{def}}{=} \delta(s, a)$.
- *Stutter steps take precedence over time steps.* If $a^1, a^2 \in \{\Delta_0, \Delta_1\}$, there are two cases.

- If $a^1 = \Delta_0$ or $a^2 = \Delta_0$, the game performs a stutter step, and $\widetilde{\delta}(s, a^1, a^2) \overset{\text{def}}{=} s$.
- If $a^1 = a^2 = \Delta_1$, then the game performs a time step of duration 1, and the game proceeds to $\widetilde{\delta}(s, a^1, a^2) \overset{\text{def}}{=} \delta(s, \Delta_1)$.

An *infinite run* (or simply *run*) of the discrete-time game structure $\mathscr{G}$ is a sequence $s_0, \langle a_1^1, a_1^2 \rangle, s_1, \langle a_2^1, a_2^2 \rangle, s_2, \ldots$ such that $s_k \in S$, $a_{k+1}^1 \in M_1(s_k)$, $a_{k+1}^2 \in M_2(s_k)$, and $s_{k+1} \in \widetilde{\delta}(s_k, a_{k+1}^1, a_{k+1}^2)$ for all $k \geq 0$. A *finite run* σ is a finite prefix of a run that terminates at a state s, we then set $last(\sigma) = s$. We denote by *FRuns* the set of all finite runs of the game structure, and by *Runs* the set of its infinite runs. For a finite or infinite run σ, and a number $k < |\sigma|$, we denote by $\sigma_{\leq k}$ the prefix of σ up to and including state σ_k. A state s' is *reachable* from another state s if there exists a finite run $s_0, \langle a_1^1, a_1^2 \rangle, s_1, \ldots, s_n$ such that $s_0 = s$ and $s_n = s'$.

2.2 Strategies and Game Outcomes

A *strategy* π_i for player $i \in \{1, 2\}$ is a mapping $\pi_i : FRuns \mapsto M_i$ that associates with each finite run $s_0, \langle a_1^1, a_1^2 \rangle, s_1, \ldots, s_n$ the move $\pi_i(s_0, \langle a_1^1, a_1^2 \rangle, s_1, \ldots, s_n)$ to be played at s_n. We require that the strategy only selects enabled moves, that is, $\pi_i(\sigma) \in M_i(last(\sigma))$ for all $\sigma \in FRuns$. For $i \in \{1, 2\}$, let Π_i denote the set of all player i strategies. For strategies $\pi_1 \in \Pi_1$ and $\pi_2 \in \Pi_2$, we say that a run $s_0, \langle a_1^1, a_1^2 \rangle, s_1, \ldots$ is *consistent* with π_1 and π_2 if, for all $n \geq 0$ and $i = 1, 2$, we have $\pi_i(s_0, \langle a_1^1, a_1^2 \rangle, s_1, \ldots, s_n) = a_{n+1}^i$. We denote by $Outcomes(s, \pi_1, \pi_2)$ the set of all runs that start in s and are consistent with π_1, π_2. Note that in our timed games, two strategies and a start state yield a *set* of outcomes, because if the players both propose actions, a nondeterministic choice between the two moves is made. According to this definition, strategies can base their choices on the entire history of the game, consisting of both past states and moves.

2.3 Discrete-Time Game Automata

We specify discrete-time game structures via *discrete-time game automata*, which are a discrete-time version of the *timed automaton games* of [dAFH$^+$03]; both models are two-player versions of timed automata [AD94]. A *clock condition* over a set C of clocks is a boolean combination of formulas of the form $x \preceq c$ or $x - y \preceq c$, where c is an integer, $x, y \in C$, and $\preceq$ is either $<$ or $\leq$. We denote the set of all clock conditions over C by $ClkConds(C)$. A *clock valuation* is a function $\kappa : C \mapsto \mathbb{R}_{\geq 0}$, and we denote by $K(C)$ the set of all clock valuations for C.

A *discrete-time game automaton* is a tuple $\mathscr{A} = (Q, C, Acts_1, Acts_2, E, \theta, \rho, Inv_1, Inv_2, Rew)$, where:

- Q is a finite set of locations.
- C is a finite set of clocks.
- $Acts_1$ and $Acts_2$ are two disjoint, finite sets of actions for player 1 and player 2, respectively.
- $E \subseteq Q \times (Acts_1 \cup Acts_2) \times Q$ is an edge relation.
- $\theta : E \mapsto ClkConds(C)$ is a mapping that associates with each edge a clock condition that specifies when the edge can be traversed. We require that for all $(q, a, q_1), (q, a, q_2) \in E$ with $q_1 \neq q_2$, the conjunction $\theta(q, a, q_1) \wedge \theta(q, a, q_2)$ is unsatisfiable. In other words, the game move and clock values determine uniquely the successor location.

– $\rho : E \mapsto 2^C$ is a mapping that associates with each edge the set of clocks to be reset when the edge is traversed.
– $\mathit{Inv}_1, \mathit{Inv}_2 : Q \mapsto \mathit{ClkConds}(C)$ are two functions that associate with each location an invariant for player 1 and 2, respectively.
– $\mathit{Rew} : Q \mapsto \mathbb{Z}$ is a function that assignes a reward $\mathit{Rew}(q) \in \mathbb{Z}$ with each $q \in Q$.

Given a clock valuation $\kappa : C \mapsto \mathrm{I\!R}_{\geq 0}$, we denote by $\kappa + 1$ the valuation defined by $(\kappa+1)(x) = \kappa(x) + 1$ for all clocks $x \in C$. The clock valuation $\kappa : C \mapsto \mathrm{I\!R}_{\geq 0}$ *satisfies* the clock constraint $\alpha \in \mathit{ClkConds}(C)$, written $\kappa \models \alpha$, if α holds when the clocks have the values specified by κ. For a subset $C' \subseteq C$ of clocks, $\kappa[C' := 0]$ denotes the valuation defined by $\kappa[C' := 0](x) = 0$ if $x \in C'$, and by $\kappa[C' := 0](x) = \kappa(x)$ otherwise.

The discrete-time game automaton $\mathscr{A}$ induces a discrete-time game structure $[\![\mathscr{A}]\!]$, whose states consist of a location of $\mathscr{A}$ and a clock valuation over C. The idea is the following. The move Δ_0 is always enabled at all states $\langle q, \kappa \rangle$, and leads again to $\langle q, \kappa \rangle$. The move Δ_1 is enabled for player $i \in \{1,2\}$ at state $\langle q, \kappa \rangle$ if $\kappa + 1 \models \mathit{Inv}_i(q)$; the move leads to state $\langle q, \kappa+1 \rangle$. For player $i \in \{1,2\}$ and $a \in \mathit{Acts}_i$, the move a is enabled at a state $\langle q, \kappa \rangle$ if there is a transition (q, a, q') in E which is enabled at $\langle q, \kappa \rangle$, and if the invariant $\mathit{Inv}_i(q')$ holds for the destination state $\langle q', \kappa[\rho(q,a,q') := 0] \rangle$. If the values of the clocks can grow unboundedly, this translation would yield an infinite-state discrete-time game structure. However, we can define *clock regions* similarly to timed automata [AD94], and we can include in the discrete-time game structure only one state per clock region; as usual, this leads to a finite state space.

3 The Average Reward Condition

In this section, we consider a discrete-time game structure $\mathscr{G} = (S, \mathit{Acts}_1, \mathit{Acts}_2, \Gamma_1, \Gamma_2, \delta, r)$, unless otherwise noted.

3.1 Long-Run Average Reward of a Run

We consider games where the goal for player 1 consists in maximizing the average reward per time unit obtained along a game outcome. The goal for player 2 is symmetrical, and it consists in minimizing the average reward per time unit obtained along a game outcome. To make these goals precise, consider a finite run $\sigma = s_0, \langle a_1^1, a_1^2 \rangle, s_1, \ldots, s_n$, and for all $k = 0, \ldots n$, denote by σ_k its k-th state s_k, and by σ_k^1, σ_k^2 the moves a_k^1 and a_k^2 played by players 1 and 2 at the k-th position of σ. The time D_k elapsed at step k of the run is defined by $D_k(\sigma) = 1$ if $\sigma_k^1 = \sigma_k^2 = \Delta_1$, and $D_k(\sigma) = 0$ otherwise; the reward R_k accrued at step k of the run is given by $R_k(\sigma) = r(\sigma_k) \cdot D_k$. The time elapsed during σ and the reward achieved during σ are defined in the obvious way, by $D(\sigma) = \sum_{k=0}^{n} D_k(\sigma)$ and $R(\sigma) = \sum_{k=0}^{n} R_k(\sigma)$. Finally, we define the long-run average reward of an infinite run σ' by:

$$\overline{r}(\sigma') = \liminf_{n \to \infty} \frac{R(\sigma'_{\leq n})}{D(\sigma'_{\leq n})}.$$

3.2 The Value of the Game

A first attempt to define the goal of the game consists in asking for the maximum value of this long-run average reward that player 1 can secure. According to this approach, the value for player 1 of the game at a state s would be defined by

$$\widetilde{v}(s) = \sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} \inf\{\overline{r}(\sigma) \mid \sigma \in \mathit{Outcomes}(s, \pi_1, \pi_2)\}.$$

However, this approach fails to take into account the fact that, in timed games, players must not only play in order to achieve the goal, but must also play realistic strategies that guarantee the advancement of time. As an example, consider the game of Figure 1. We have $\widetilde{v}(\langle q_0, [x := 0]\rangle) = 4$, and the optimal strategy of player 1 consists in staying at q_0 forever, never playing the move a^1. Due to the invariant $x \leq 2$, such a strategy blocks the progress of time: once $x = 2$, the only move player 1 can play is Δ_0. It is easy to see that the only strategies of player 1 that do not block time eventually play move a^1, and have value 1. Note that the game does not contain any blocked states, i.e., from every reachable state there is a run that is time-divergent: the lack of time progress of the above-mentioned strategy is due to the fact that player 1 values more obtaining high average reward, than letting time progress.

To ensure that winning strategies do not block the progress of time, we modify the definition of value of a run, so that ensuring time divergence has higher priority than maximizing the average reward. Following [dAFH$^+$03], we introduce the following predicates:

– For $i \in \{1, 2\}$, we denote by $\mathit{blameless}^i(\sigma)$ ("*blameless i*") the predicate defined by $\exists n \geq 0. \forall k \geq n. \sigma^i_k = \Delta_1$. Intuitively, $\mathit{blameless}^i(\sigma)$ holds if, along σ, player i beyond a certain point does not play any moves that block the progress of time.
– We denote by $\mathit{td}(\sigma)$ ("time-divergence") the predicate defined by $\forall n \geq 0\,.\,\exists k \geq n\,.\,[(\sigma^1_k = \Delta_1) \wedge (\sigma^2_k = \Delta_1)]$.

We define the value of a run $\sigma \in \mathit{Runs}$ for player $i \in \{1, 2\}$ by:

$$w_i(\sigma) = \begin{cases} +\infty & \text{if } \mathit{blameless}^i(\sigma) \wedge \neg \mathit{td}(\sigma); \\ (-1)^{(i+1)}\, \overline{r}(\sigma) & \text{if } \mathit{td}(\sigma); \\ -\infty & \text{if } \neg \mathit{blameless}^i(\sigma) \wedge \neg \mathit{td}(\sigma). \end{cases} \tag{1}$$

It is easy to check that, for each run, exactly one of the three cases of the above definition applies. Notice that if $\mathit{td}(\sigma)$ holds, then $w_1(\sigma) = -w_2(\sigma)$, so that the value of time-divergent runs is defined in a zero-sum fashion. We define the value of the game for player i at $s \in S$ as follows:

$$v_i(\mathscr{G}, s) = \sup_{\pi_i \in \Pi_i} \inf_{\pi_{\sim i} \in \Pi_{\sim i}} \inf\{w_i(\sigma) \mid \sigma \in \mathit{Outcomes}(s, \pi_1, \pi_2)\}. \tag{2}$$

We omit the argument $\mathscr{G}$ from $v_i(\mathscr{G}, s)$ when clear from the context.

♣ note: dropped well-formedness ♣ Since we desire games where time progresses, we consider only games where $v_i(s) > -\infty$. Notice that this implies $v_{\sim i}(s) < +\infty$, so time must diverge when both players use their optimal strategy.

3.3 Determinacy

A game is *determined* if, for all $s \in S$, we have $v_1(s) + v_2(s) = 0$: this means that if player $i \in \{1, 2\}$ cannot enforce a reward $c \in \mathbb{R}$, then player $\sim i$ can enforce at least reward $-c$. The following theorem provides a strong non-determinacy result for average-reward discrete-time games.

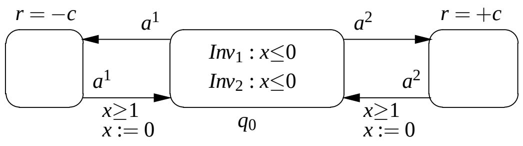

Fig. 2. A game automaton. Unspecified guards and invariants are "true".

Theorem 1. *(non-determinacy) For all* $c > 0$*, there exists a game structure* $\mathcal{G} = (S, Acts_1, Acts_2, \Gamma_1, \Gamma_2, \delta, r)$ *with a state* $s \in S$*, and two "spoiling" strategies* $\pi_1^* \in \Pi_1$*,* $\pi_2^* \in \Pi_2$*, such that the following holds:*

$$\sup_{\pi_1 \in \Pi_1} \sup\{w_1(\sigma) \mid \sigma \in Outcomes(s, \pi_1, \pi_2^*)\} \leq -c$$

$$\sup_{\pi_2 \in \Pi_2} \sup\{w_2(\sigma) \mid \sigma \in Outcomes(s, \pi_1^*, \pi_2)\} \leq -c.$$

As a consequence, $v_1(s) \leq -c$ *and* $v_2(s) \leq -c$*.*

Note that in the theorem we take sup, rather than inf as in (2), over the set of outcomes arising from the strategies. Hence, the theorem states that, even if the chose among actions is resolved in favor of the player trying to achieve the value, there is a game with a state s where $v_1(s) + v_2(s) \leq 2c < 0$. Moreover, in the theorem, the adversary strategies are fixed, again providing an advantage to the player trying to achieve the value.

Proof. Consider the game of Figure 2. We take for $\pi_1^* \in \Pi_1$ and $\pi_2^* \in \Pi_2$ the strategies that play always Δ_0. Let $s_0 = \langle q_0, [x := 0] \rangle$, and consider the value

$$\widehat{v}_1(s_0) = \sup_{\pi_1 \in \Pi_1} \sup\{w_1(\sigma) \mid \sigma \in Outcomes(s_0, \pi_1, \pi_2^*)\}.$$

There are two cases. If eventually player 1 plays forever Δ_0 in s_0, player 1 obtains the value $-\infty$, as time does not progress, and player 1 is not blameless. If player 1, whenever at s_0, eventually plays a^1, then the value of the game to player 1 is $-c$. Hence, we have $\widehat{v}_1(s_0) = -c$. The analysis for player 2 is symmetrical. ∎

The example of Figure 2, together with the above analysis, indicates that we cannot define the value of an average reward discrete-time game in a way that leads to determinacy, and enforces time progress. In fact, consider again the case in which player 2 plays always Δ_0 at s_0. If, beyond some point, player 1 plays forever Δ_0 in s_0, time does not progress, and the situation is symmetrical wrt. players 1 and 2: they both play forever Δ_0. Hence, we must rule out this combination of strategies (either by assigning value $-\infty$ to the outcome, as we do, or by some other device). Once this is ruled out, the other possibility is that player 1, whenever in s_0, eventually plays a^1. In this case, time diverges, and the average value to player 1 is $-c$. As the analysis is symmetrical, the value to both players is $-c$, contradicting determinacy.

4 Solution of Average Reward Timed Games

In this section, we solve the problem of computing the value of an average reward timed game with respect to both players. First, we define a turn-based version of the timed game. Such version is equivalent to the first one when one is concerned with the value achieved by a specific player. Then, following [EM79], we define a finite game and we prove that it has the same value as the turn-based infinite game. This will lead to a PSPACE algorithm for computing the value of the game. We then show that the finite and, consequently, the infinite game admit positional optimal strategies for both players; as mentioned in the introduction, this will enable us to show that the problem of computing the value of the game is in NP∩coNP.

♣ note: dropped well-formedness. Next sentence requires use of optimal strategies, instead. ♣ In the remainder of this section, we consider a fixed discrete-time game structure $\mathcal{G} = (S, Acts_1, Acts_2, \Gamma_1, \Gamma_2, \delta, r)$, and we assume that there are optimal strategies for each player such that time must diverge. We focus on the problem of computing $v_1(s)$, as the problem of computing $v_2(s)$ is symmetrical. For a finite run σ and a finite or infinite run σ' such that $last(\sigma) = first(\sigma')$, we denote by $\sigma \cdot \sigma'$ their concatenation, where the common state is included only once.

4.1 Turn-based Timed Game

We describe a turn-based version of the timed game, where at each round player 1 chooses his move before player 2. Player 2 can thus use her knowledge of player 1's move to choose her own. Moreover, when both players choose an action, the action chosen by player 2 is carried out. This accounts for the fact that in the definition of $v_1(s)$, nondeterminism is resolved in favor of player 2 (see (2)). Notice that if player 2 prefers to carry out the action chosen by player 1, she can reply with the stuttering move Δ_0. Definitions pertaining this game have a "t∞" superscript that stands for "turn-based infinite".

We define the *turn-based joint destination function* $\widetilde{\delta}^{\mathrm{t}} : S \times M_1 \times M_2 \mapsto S$ by

$$\widetilde{\delta}^{\mathrm{t}}(s, a^1, a^2) = \begin{cases} \delta(s, \Delta_1) & \text{if } a^1 = a^2 = \Delta_1 \\ \delta(s, \Delta_0) & \text{if } \{a^1, a^2\} \subseteq \{\Delta_0, \Delta_1\} \text{ and } a^1 = \Delta_0 \text{ or } a^2 = \Delta_0 \\ \delta(s, a^1) & \text{if } a^1 \in Acts_1 \text{ and } a^2 \in \{\Delta_0, \Delta_1\} \\ \delta(s, a^2) & \text{if } a^2 \in Acts_2 \end{cases}$$

As before, a run is an infinite sequence $s_0, \langle a_1^1, a_1^2 \rangle, s_1, \langle a_2^1, a_2^2 \rangle, s_2, \ldots$ such that $s_k \in S$, $a_{k+1}^1 \in M_1(s_k)$, $a_{k+1}^2 \in M_2(s_k)$, and $s_{k+1} \in \widetilde{\delta}^{\mathrm{t}}(s_k, a_{k+1}^1, a_{k+1}^2)$ for all $k \geq 0$. A 1*-run* is a finite prefix of a run ending in a state s_k, while a 2*-run* is a finite prefix of run ending in a move $a \in M_1$, so it is a sequence of the type $s_0, \langle a_1^1, a_1^2 \rangle, s_1, \ldots, s_n, \langle a_{n+1}^1 \rangle$. We set $last(s_0, \langle a_1^1, a_1^2 \rangle, s_1, \ldots, s_n, \langle a_{n+1}^1 \rangle) = s_n$. For $i \in \{1,2\}$, we denote by $FRuns_i$ the set of all i-runs. Intuitively, i-runs are runs where it is player i's turn to move.

In the turn-based game, a *strategy* π_i for player $i \in \{1,2\}$ is a mapping $\pi_i : FRuns_i \mapsto M_i$. As before, we require that the strategy only selects enabled moves, that is, $\pi_i(\sigma) \in M_i(last(\sigma))$ for all $\sigma \in FRuns_i$. For $i \in \{1,2\}$, let Π_i^{t} denote the set of all player i strategies. Notice that $\Pi_1^{\mathrm{t}} = \Pi_1$.

For strategies $\pi_1 \in \Pi_1^{\mathrm{t}}$ and $\pi_2 \in \Pi_2^{\mathrm{t}}$, we say that a run $s_0, \langle a_1^1, a_1^2 \rangle, s_1, \ldots$ is *consistent* with π_1 and π_2 if, for all $n \geq 0$ and $i = 1,2$, we have $\pi_1(s_0, \langle a_1^1, a_1^2 \rangle, s_1, \ldots, s_n) = a_{n+1}^1$ and $\pi_2(s_0, \langle a_1^1, a_1^2 \rangle, s_1, \ldots, s_n, \langle a_{n+1}^1 \rangle) = a_{n+1}^2$. Since $\widetilde{\delta}^{\mathrm{t}}$ is deterministic, for all $s \in S$, there is a unique run that starts in s and is consistent with π_1 and π_2. We denote this run by $\mathit{outcome}^{\mathrm{t}\infty}(s,\pi_1,\pi_2)$.

The value assigned to a run, to a strategy and to the whole game are defined as follows. We set $w_1^{\mathrm{t}\infty}(\sigma) = w_1(\sigma)$, and

$$v_1^{\mathrm{t}\infty}(s,\pi_1) = \inf_{\pi_2 \in \Pi_2^{\mathrm{t}}} w_1^{\mathrm{t}\infty}(\mathit{outcome}^{\mathrm{t}\infty}(s,\pi_1,\pi_2)); \qquad v_1^{\mathrm{t}\infty}(s) = \sup_{\pi_1 \in \Pi_1^{\mathrm{t}}} v_1^{\mathrm{t}\infty}(s,\pi_1).$$

The following theorem is proved in the appendix.

Theorem 2. *For all $s \in S$, it holds $v_1(s) = v_1^{\mathrm{t}\infty}(s)$.*

4.2 Turn-based Finite Game

We now define a finite turn-based game that can be played on a discrete-time game structure. Definitions pertaining this game have a "tf" superscript that stands for "turn-based finite". The finite game ends as soon as a loop is closed. A *maximal run* in the finite game is a 1-run $\sigma = s_0, \langle a_1^1, a_1^2 \rangle, s_1, \ldots, s_n$ such that s_n is the first state that is repeated in σ. Formally, n is the least number such that $s_n = s_j$, for some $j < n$. We set $\mathit{loop}(\sigma)$ to be the suffix of σ: $s_j, \langle a_{j+1}^1, a_{j+1}^2 \rangle, \ldots, s_n$. For $\pi_1 \in \Pi_1^{\mathrm{t}}$, $\pi_2 \in \Pi_2^{\mathrm{t}}$, and $s \in S$, we denote by $\mathit{outcome}^{\mathrm{tf}}(s,\pi_1,\pi_2)$ the unique maximal run that starts in s and is consistent with π_1 and π_2.

In the finite game, a maximal run σ ending with the loop λ is assigned the value of the infinite run obtained by repeating λ forever. Formally, $w_1^{\mathrm{tf}}(\sigma) = w_1(\sigma \cdot \lambda^\omega)$, where λ^ω denotes the concatenation of numerably many copies of λ. The value assigned to a strategy $\pi_1 \in \Pi_1^{\mathrm{t}}$ and the value assigned to the whole game are defined as follows.

$$v_1^{\mathrm{tf}}(s,\pi_1) = \inf_{\pi_2 \in \Pi_2^{\mathrm{t}}} w_1^{\mathrm{tf}}(\mathit{outcome}^{\mathrm{tf}}(s,\pi_1,\pi_2)); \qquad v_1^{\mathrm{tf}}(s) = \sup_{\pi_1 \in \Pi_1^{\mathrm{t}}} v_1^{\mathrm{tf}}(s,\pi_1).$$

Notice that since this game is finite and turn-based, for all $s \in S$, it holds:

$$\sup_{\pi_1 \in \Pi_1} \inf_{\pi_2 \in \Pi_2} w_1^{\mathrm{tf}}(\mathit{outcome}^{\mathrm{tf}}(s,\pi_1,\pi_2)) = \inf_{\pi_2 \in \Pi_2} \sup_{\pi_1 \in \Pi_1} w_1^{\mathrm{tf}}(\mathit{outcome}^{\mathrm{tf}}(s,\pi_1,\pi_2)). \tag{3}$$

4.3 Mapping Strategies

In this section, we introduce definitions that allow us to relate the finite game to the infinite one. For a 1-run $\sigma = s_0, \langle a_1^1, a_1^2 \rangle, s_1, \ldots, s_n$, let $\mathit{firstloop}(\sigma)$ be the operator that returns the first simple loop (if any) occurring in σ. Similarly, let $\mathit{loopcut}(\sigma)$ be the operator that removes the first simple loop (if any) from σ. Formally, if σ is a simple run (i.e. it contains no loops) we set $\mathit{firstloop}(\sigma) = \varepsilon$ (the empty sequence), and $\mathit{loopcut}(\sigma) = \sigma$. Otherwise, let $k \geq 0$ be the smallest number such that $\sigma_j = \sigma_k$, for some $j < k$; we set

$$\begin{aligned} \mathit{firstloop}(\sigma) &= \sigma_j, \langle a_{j+1}^1, a_{j+1}^2 \rangle, \ldots, \langle a_k^1, a_k^2 \rangle, \sigma_k; \\ \mathit{loopcut}(\sigma) &= \sigma_0, \langle a_1^1, a_1^2 \rangle, \ldots, \sigma_j, \langle a_{k+1}^1, a_{k+1}^2 \rangle, \ldots, \sigma_n. \end{aligned}$$

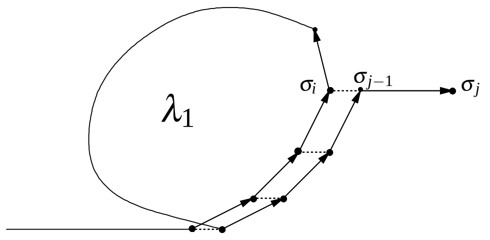

Fig. 3. Nodes linked by dashed lines represent the same state of the game.

We now define the *quasi-segmentation* $QSeg(\sigma)$ to be the sequence of simple loops obtained by applying *firstloop* repeatedly to σ.

$$QSeg(\sigma) = \begin{cases} \varepsilon & \text{if } firstloop(\sigma) = \varepsilon \\ firstloop(\sigma) \cdot QSeg(loopcut(\sigma)) & \text{otherwise} \end{cases}$$

For an infinite run σ, we set $QSeg(\sigma) = \lim_{n\to\infty} QSeg(\sigma_{\leq n})$. Given a finite run σ, *loopcut* can only be applied a finite number of times before it converges to a fixpoint. We call this fixpoint $residual(\sigma)$. Notice that for all runs σ, $residual(\sigma)$ is a simple path and therefore its length is bounded by $|S|$.

For simplicity, we developed the above definitions for 1-runs. The corresponding definitions of $residual(\sigma)$ and $QSeg(\sigma)$ for 2-runs σ are similar.

For all $i \in \{1,2\}$ and all strategies $\pi \in \Pi_i^{\mathrm{t}}$, we define the strategy $\tilde{\pi}$ as $\tilde{\pi}(\sigma) = \pi(residual(\sigma))$ for all $\sigma \in FRuns_i$. Intuitively, $\tilde{\pi}$ behaves like π until a loop is formed. At that point, $\tilde{\pi}$ *forgets* the loop, behaving as if the whole loop had not occurred. We now give two technical lemmas.

Lemma 1. *Let $\pi_1 \in \Pi_1^{\mathrm{t}}$, $\pi_2 \in \Pi_2^{\mathrm{t}}$, and $\sigma = outcome^{\mathrm{t}\infty}(s, \tilde{\pi}_1, \pi_2)$. For all $k > 0$, $residual(\sigma_{\leq k})$ is a prefix of a finite run consistent with π_1. Formally, there is $\pi_2' \in \Pi_2^{\mathrm{t}}$ and $\sigma' = outcome^{\mathrm{tf}}(s, \pi_1, \pi_2')$ such that $\sigma' = residual(\sigma_{\leq k}) \cdot \rho$.*

Similarly, let $\sigma = outcome^{\mathrm{t}\infty}(s, \pi_1, \tilde{\pi}_2)$. For all $k > 0$, there is $\pi_1' \in \Pi_1^{\mathrm{t}}$ and $\sigma' = outcome^{\mathrm{tf}}(s, \pi_1', \pi_2)$ such that $\sigma' = residual(\sigma_{\leq k}) \cdot \rho$.

Proof. We prove the first statement, as the second one is analogous. We proceed by induction on the length of $QSeg(\sigma_{\leq k})$. If $QSeg(\sigma_{\leq k})$ is the empty sequence (i.e. $\sigma_{\leq k}$ contains no loops), the result is easily obtained, as $\tilde{\pi}_1$ coincides with π_1 until a loop is formed. So, we can take $\pi_2' = \pi_2$ and obtain the conclusion.

On the other hand, suppose $QSeg(\sigma_{\leq k}) = \lambda_1, \ldots, \lambda_n$. For simplicity, suppose $\lambda_1 \neq \lambda_2$. As illustrated in Figure 3, let σ_j be the first state after λ_1 that does not belong to λ_1. Then, σ_{j-1} belongs to λ_1 and there is another index $i < j-1$ such that $\sigma_i = \sigma_{j-1}$. So, the game went twice through σ_{j-1} and two different successors were taken. However, player 1 must have chosen the same move in σ_i and σ_{j-1}, as by construction $\tilde{\pi}_1(\sigma_{\leq i}) = \tilde{\pi}_1(\sigma_{\leq j-1})$. Therefore, the change must be due to a different choice of π_2. It is easy to devise π_2' that coincides with π_2, except that λ_1 may be skipped when playing against $\tilde{\pi}_1$. We can then obtain a run $\rho = outcome^{\mathrm{t}\infty}(s, \tilde{\pi}_1, \pi_2')$ and an integer $k' \geq 0$ such that $QSeg(\rho_{\leq k'}) = \lambda_2, \ldots, \lambda_n$ and $residual(\rho_{\leq k'}) = residual(\rho)$. The thesis is obtained by applying the inductive hypothesis to ρ and k'. ∎

The next lemma shows that, for a strategy $\pi_1 \in \Pi_1$, each loop occurring in the infinite game under $\tilde{\pi}_1$ can also occur in the finite game under π_1. The proof can be found in the appendix.

Lemma 2. *Let $\pi_1 \in \Pi_1^{\mathrm{t}}$, $\pi_2 \in \Pi_2^{\mathrm{t}}$, and $\sigma = outcome^{\mathrm{t}\infty}(s, \tilde{\pi}_1, \pi_2)$. For all $\lambda \in QSeg(\sigma)$, λ can occur as the final loop in a maximal run of the finite game. Formally, there is $\pi_2' \in \Pi_2^{\mathrm{t}}$ and $\sigma' = outcome^{\mathrm{tf}}(s, \pi_1, \pi_2')$ such that $\lambda = loop(\sigma')$.*

Similarly, let $\sigma = outcome^{\mathrm{t}\infty}(s, \pi_1, \tilde{\pi}_2)$. For all $\lambda \in QSeg(\sigma)$, there is $\pi_1' \in \Pi_1^{\mathrm{t}}$ and $\sigma' = outcome^{\mathrm{tf}}(s, \pi_1', \pi_2)$ such that $\lambda = loop(\sigma')$.

The next theorem states that if the strategy π_1 of player 1 achieves value v in the finite turn-based game, the strategy $\tilde{\pi}_1$ achieves at least as much in the infinite turn-based game.

Theorem 3. *For all $s \in S$ and $\pi_1 \in \Pi_1^{\mathrm{t}}$, it holds $v_1^{\mathrm{t}\infty}(s, \tilde{\pi}_1) \geq v_1^{\mathrm{tf}}(s, \pi_1)$.*

Proof. Let $v = v_1^{\mathrm{tf}}(s, \pi_1)$. We show that $\tilde{\pi}_1$ can ensure reward v in the infinite game. The result is trivially true if $v = -\infty$. So, in the following we assume that $v > -\infty$.

Fix a player 2 strategy π_2, and let $\sigma = outcome^{\mathrm{t}\infty}(s, \tilde{\pi}_1, \pi_2)$. Let $QSeg(\sigma) = \lambda_1, \lambda_2 \ldots$. We distinguish two cases, according to whether time diverges or not in σ. If time diverges, all loops λ_j that contain no tick give no contribution to the value of σ and can therefore be ignored.

For all λ_j containing (at least) a time step, by Lemma 2, λ_j is a possible terminating loop for the finite game under π_1. Thus, $R(\lambda_j) \geq v \cdot D(\lambda_j)$. Now, the value of σ can be split as the value due to loops containing time steps, plus the value due to the residual. For all $n \geq 0$, let m_n be the number of loops in $QSeg(\sigma_{\leq n})$. We obtain:

$$w_1^{\mathrm{t}\infty}(\sigma) = \liminf_{n\to\infty} \frac{R(\sigma_{\leq n})}{D(\sigma_{\leq n})} = \lim_{n\to\infty} \frac{R(residual(\sigma_{\leq n})) + \sum_{j=1}^{m_n} R(\lambda_j)}{D(residual(\sigma_{\leq n})) + \sum_{j=1}^{m_n} D(\lambda_j)} = \lim_{n\to\infty} \frac{\sum_{j=1}^{m_n} R(\lambda_j)}{\sum_{j=1}^{m_n} D(\lambda_j)} \geq v.$$

Consider now the case when σ contains only finitely many time steps. Let $k \geq 0$ be such that no time steps occur in σ after σ_k. Consider a loop λ_j entirely occurring after σ_k. Obviously λ_j contains no time steps. Moreover, by Lemma 2, λ_j is a terminating loop for a maximal run ρ in the finite game under π_1. Since $v_1^{\mathrm{tf}}(s, \pi_1) > -\infty$, it must be $w_1^{\mathrm{tf}}(\rho) = +\infty$. Consequently, it holds $blameless^1(\rho)$ and in particular player 1 is blameless in all edges in λ_j.

Now, let $k' \geq 0$ be such that each state (and edge) after $\sigma_{k'}$ will eventually be part of a loop of $QSeg(\sigma)$. Let $k'' = \max\{k, k'\}$. Then, all edges that occur after k'' will eventually be part of a loop where player 1 is blameless. Consequently, k'' is a witness to the fact that $blameless^1(\sigma)$, and therefore $w_1^{\mathrm{t}\infty}(\sigma) = +\infty \geq v$. ∎

Theorem 4. *For all $s \in S$ and $\pi_2 \in \Pi_2^{\mathrm{t}}$, it holds $v_1^{\mathrm{t}\infty}(s, \tilde{\pi}_2) \leq v_1^{\mathrm{tf}}(s, \pi_2)$.*

Proof. Let $v = v_1^{\mathrm{tf}}(s, \pi_2)$. Similarly to Theorem 3, we can rule out the case $v = +\infty$ as trivial.

Fix a player 1 strategy π_1, and let $\sigma = outcome^{\mathrm{t}\infty}(s, \pi_1, \tilde{\pi}_2)$. We show that $w_1^{\mathrm{t}\infty}(\sigma) \leq v$. If time diverges on σ, the proof is similar to the analogous case in Theorem 3.

Otherwise, let $k \geq 0$ be such that no time steps occur in σ after σ_k. Consider a loop $\lambda \in QSeg(\sigma)$, entirely occurring after σ_k. Obviously λ contains no time steps.

Moreover, by Lemma 2, λ is a terminating loop for a maximal run ρ in the finite game under π_1. Since $v_1^{\mathrm{tf}}(s,\pi_1) < +\infty$, it must be $w_1^{\mathrm{tf}}(\rho) = -\infty$. Consequently, it holds $\neg blameless^1(\rho)$ and in particular player 1 is blamed in some edge of λ. This shows that $\neg blameless^1(\sigma)$, and consequently $w_1^{\mathrm{t}\infty}(\sigma) = -\infty \leq v$. ∎

Theorems 3 and 4 show that the infinite game is no harder than the finite one, for both players. Considering also (3), we obtain the following result.

Theorem 5. *For all $s \in S$, $v_1^{\mathrm{t}\infty}(s) = v_1^{\mathrm{tf}}(s)$.*

Theorems 2 and 5 allow us to use the finite game to compute the value of the original timed game. The length of the finite game is bounded by $|S|$. It is well-known that a recursive, backtracking algorithm can compute the value of such game in PSPACE.

Theorem 6. *For all $s \in S$, $v_1(s)$ can be computed in PSPACE.*

4.4 Memory

We show that memoryless strategies suffice for both players to reach their value. Notice that it is sufficient to show that the finite game has memoryless optimal strategies, as the result follows from Theorems 3 and 4. To this end, following [EM79] we define a modified version of the finite game, which is still played on the same discrete-time game structure. For a state $s \in S$, the s-forgetful game is played as the finite game, except that the first time s is encountered while playing, the history up to s is forgotten for the purpose of checking game termination. Formally, a maximal run in the s-forgetful game is a finite run $\sigma = s_0, \langle a_1^1, a_1^2 \rangle, s_1, \ldots, s_n$ such that:

– either s does not occur in σ and s_n is the first state that is repeated in σ,
– or σ_k is the first occurrence of s in σ, and s_n is the first state that is repeated in $\sigma_{\geq k}$.

We define $loop(\sigma)$ to be the final loop λ of σ, and we set $w_1^{\mathrm{tf},s}(\sigma) = w_1(\sigma \cdot \lambda^\omega)$. We denote by $outcome^{\mathrm{tf},s}(t,\pi_1,\pi_2)$ the unique maximal run in the s-forgetful game that starts in t and is consistent with π_1, π_2. The value assigned to a strategy $\pi_1 \in \Pi_1^{\mathrm{t}}$ and the value assigned to the whole game are defined as usual:

$$v_1^{\mathrm{tf},s}(t,\pi_1) = \inf_{\pi_2 \in \Pi_2^{\mathrm{t}}} w_1^{\mathrm{tf},s}(outcome^{\mathrm{tf},s}(t,\pi_1,\pi_2)); \qquad v_1^{\mathrm{tf},s}(t) = \sup_{\pi_1 \in \Pi_1^{\mathrm{t}}} v_1^{\mathrm{tf},s}(t,\pi_1).$$

Lemma 3. *For all $s,t \in S$, $v_1^{\mathrm{tf}}(t) = v_1^{\mathrm{tf},s}(t)$.*

Proof. Let $\pi_1 \in \Pi_1^{\mathrm{t}}$ be a strategy for player 1 such that $v_1^{\mathrm{tf}}(t,\pi_1) = v_1^{\mathrm{tf}}(t) = v$. We use strategy $\tilde{\pi}_1$ to play the s-forgetful game. Clearly, runs of the forgetful game that do not contain s may also have occurred in the normal finite game, so those runs have value at least v.

Consider now a run $\sigma = outcome^{\mathrm{tf},s}(t,\tilde{\pi}_1,\pi_2)$ which contains state s. The value $w_1^{\mathrm{tf},s}(\sigma)$ is entirely determined by the final loop $\lambda = loop(\sigma)$. One can check that there is $\pi_2' \in \Pi_2^{\mathrm{t}}$ such that the run $\sigma' = outcome^{\mathrm{t}\infty}(t,\tilde{\pi}_1,\pi_2')$ ends with the sequence λ^ω. By Theorem 5, we have $w_1^{\mathrm{tf},s}(\sigma) = w_1(\sigma') \geq v_1(t) = v_1^{\mathrm{tf}}(t) = v$. This proves that $v_1^{\mathrm{tf}}(t) \geq v_1^{\mathrm{tf},s}(t)$. The converse inequality is proved by a symmetrical argument, exchanging the roles of the two players. ∎

The next theorem states that memoryless optimal strategies exist for both players. The proof, based on Lemma 3, can be found in the appendix.

Theorem 7. *For all $i \in \{1,2\}$, and $t \in S$, there exists a memoryless optimal strategy for player i. Formally, there exists $\pi_i \in \Pi_i$ such that $v_1(t,\pi_i) = v_1(t)$.*

4.5 Improved Algorithms

We show that, given $s \in S$, $v \in \mathbb{Q}$ and $i \in \{1,2\}$, the problem of checking whether $v_i^{\mathrm{tf}}(s) \geq v$ is in NP∩coNP. We exploit the existence of positional optimal strategies for both players, as such strategies can be "guessed" in polynomial time. Consider the problem of checking whether $v_1^{\mathrm{tf}}(s) \geq v$. The problem is in NP because, once the optimal strategy of player 1 has been guessed, we can compute in polynomial time the value $v_1^{\mathrm{tf}}(s,\pi_1)$. The problem is also in coNP because, once a strategy of player 2 is fixed, we can compute in polynomial time the value $v_1^{\mathrm{tf}}(s,\pi_2)$.

Once we fix a positional strategy for player i, the finite game is reduced to a graph, where all the choices belong to player $\sim i$. Each edge e in this graph can be labeled with a pair of moves $(a^1,a^2) \in M_1 \times M_2$. Then, e is labeled with *tick* whenever $a^1 = a^2 = \Delta_1$, and with bl_i whenever $a^i \in Acts_i \cup \{\Delta_0\}$. To determine the value of this graph, we first transform the graph into a graph where all edges are labeled with *tick*; we then apply the algorithm of [Kar78] for finding miminum average cost cycles in a graph. We proceed depending on whether player 1, or player 2, fixes a positional strategy. When player 1 fixes a positional strategy:

1. Find all the loops λ in the graph where all the edges are labeled with $\neg tick$ and $\neg bl_1$. Since player 2 will want to avoid λ, collapse it into a single graph vertex, without self-loop, and with incoming and outgoing edges that correspond to all edges incoming and outgoing from the loop.
   Some of these collapsed vertices may have no outgoing edges. Then their value, and the value of all vertices from which one cannot avoid entering them, is $+\infty$; remove all such vertices from the graph.
2. Find all the loops where all the edges are labeled with $\neg tick$. Due to the collapsing in the above step, each of these loops contains at least one edge labeled bl_1, so its value is $-\infty$. These loops, and all vertices that can reach them, have value $-\infty$. Remove them from the graph.
3. From the resulting graph G, construct a multigraph G' with the same vertices as G. For each simple path in G of the from $u_0,\ldots,u_n,u_{n+1}$ where the edges $(u_0,u_1),\ldots,(u_{n-1},u_n)$ are labeled by $\neg tick$, and the edge (u_n,u_{n+1}) is labeled by *tick*, we insert in G' an edge (u_0,u_{n+1}) labeled by the same reward as (u_n,u_{n+1}).
4. Use the algorithm of [Kar78] to find the loop with minimal average reward in G' (the algorithm of [Kar78] is phrased for graphs, but it can be trivially adapted to multigraphs). If r is the ratio of the loop thus found, all the vertices of the loop, and all the vertices that can reach the loop, have value r. Remove them from G'. Repeat this step until s is removed from the graph.

Clearly, the stage at which s is removed from the graph determines the value of the game at s. Similarly (but not symmetrically), if player 2 fixes a positional strategy, we can compute the value for player 1 as follows:

1. Find all the loops where all the edges are labeled with $\neg tick$ and $\neg bl_1$. These loops, and all vertices that can reach them, have value $+\infty$. Remove them from the graph.
2. Find all the loops λ where all the edges are labeled with $\neg tick$. Due to the previous step, λ contains at least one edge labeled bl_1, so player 1 will want to avoid it. Collapse λ into a single graph vertex, without self-loop, and with incoming and outgoing edges that correspond to all edges incoming and outgoing from λ.
   Some of these collapsed vertices may have no outgoing edges. Then their value, and the value of all vertices from which one cannot avoid entering them, is $-\infty$; remove all such vertices from the graph.
3. From the resulting graph G, construct a graph G' as in step 3 of the previous case.
4. This step is the same as step 4 of the previous case, except that in each iteration we find the loop with *maximal* average reward.

Since the algorithm of [Kar78], as well as the above graph manipulations, can all be done in polynomial time, we have the following result.

Theorem 8. *The problem of computing the value to player $i \in \{1,2\}$ of a discrete-time average reward game is in NP$\cap$coNP.*

References

[ABM04] R. Alur, M. Bernadsky, and P. Madhusudan. Optimal reachability for weighted timed games. In *Proc. 31st Int. Colloq. Aut. Lang. Prog.*, volume 3142 of *Lect. Notes in Comp. Sci.* Springer-Verlag, 2004.

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theor. Comp. Sci.*, 126:183–235, 1994.

[AH97] R. Alur and T.A. Henzinger. Modularity for timed and hybrid systems. In *CONCUR 97: Concurrency Theory. 8th Int. Conf.*, volume 1243 of *Lect. Notes in Comp. Sci.*, pages 74–88. Springer-Verlag, 1997.

[AMAS98] E. Asarin, O. Maler, A.Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *Proc. IFAC Symposium on System Structure and Control*, pages 469–474. Elsevier, 1998.

[BCFL04] P. Bouyer, F. Cassez, E. Fleury, and K.G. Larsen. Optimal strategies in priced timed game automata. In *Found. of Software Technology and Theoretical Comp. Sci.*, volume 3328 of *Lect. Notes in Comp. Sci.*, pages 148–160. Springer-Verlag, 2004.

[Chu63] A. Church. Logic, arithmetics, and automata. In *Proc. International Congress of Mathematicians, 1962*, pages 23–35. Institut Mittag-Leffler, 1963.

[Con92] A. Condon. The complexity of stochastic games. *Information and Computation*, 96:203–224, 1992.

[dAFH$^+$03] L. de Alfaro, M. Faella, T.A. Henzinger, R. Majumdar, and M. Stoelinga. The element of surprise in timed games. In *CONCUR 03: Concurrency Theory. 14th Int. Conf.*, volume 2761 of *Lect. Notes in Comp. Sci.*, pages 144–158. Springer-Verlag, 2003.

[dAHS02] L. de Alfaro, T.A. Henzinger, and M. Stoelinga. Timed interfaces. In *Proceedings of the Second International Workshop on Embedded Software (EMSOFT 2002)*, Lect. Notes in Comp. Sci., pages 108–122. Springer-Verlag, 2002.

[EJ91] E.A. Emerson and C.S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *Proc. 32nd IEEE Symp. Found. of Comp. Sci.*, pages 368–377. IEEE Computer Society Press, 1991.

[EM79] A. Ehrenfeucht and J. Mycielsky. Positional strategies for mean payoff games. *Int. Journal of Game Theory*, 8(2):109–113, 1979.

[HHM99] T.A. Henzinger, B. Horowitz, and R. Majumdar. Rectangular hybrid games. In *CONCUR'99: Concurrency Theory. 10th Int. Conf.*, volume 1664 of *Lect. Notes in Comp. Sci.*, pages 320–335. Springer-Verlag, 1999.

[Kar78] R.M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23:309–311, 1978.

[MPS95] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *Proc. of 12th Annual Symp. on Theor. Asp. of Comp. Sci.*, volume 900 of *Lect. Notes in Comp. Sci.*, pages 229–242. Springer-Verlag, 1995.

[PR89] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proceedings of the 16th Annual Symposium on Principles of Programming Languages*, pages 179–190. ACM Press, 1989.

[RW89] P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *IEEE Transactions on Control Theory*, 77:81–98, 1989.

A Additional Proofs

A.1 Proof of Theorem 2

Consider any $\pi_1 \in \Pi_1$. Since $\Pi_1 = \Pi_1^{\mathrm{t}}$, player 1 can employ π_1 in both the original timed game and in the turn-based game. We show that in the turn-based game π_1 can achieve at least the same value as in the original timed game. Consider any $\pi_2 \in \Pi_2^{\mathrm{t}}$ and $\sigma \in \mathit{outcome}^{\mathrm{t}\infty}(s, \pi_1, \pi_2)$. Player 2 cannot directly use π_2 in the original timed game, because in that game she cannot base her decisions on the current move of player 1. However, since π_1 is fixed, we can find $\pi_2 \in \Pi_2$ that *guesses* the move of player 1 at each step, effectively simulating the behavior of π_2 when played against π_1. It then holds that $\sigma \in \mathit{Outcomes}(s, \pi_1, \pi_2')$. Since this holds for any π_1, we have that $v_1(s) \leq v_1^{\mathrm{t}\infty}(s)$.

The inequality $v_1(s) \geq v_1^{\mathrm{t}\infty}(s)$ is immediate, since it is clearly an advantage for player 1 to conceal his move from player 2 at each round. ∎

A.2 Proof of Lemma 2

We prove the first statement, regarding the comparison between π_1 and $\tilde{\pi}_1$, as the other statement is analogous. Let $\mathit{QSeg}(\sigma) = \lambda_1, \lambda_2, \ldots$, and $\lambda = \lambda_j$. Let k be the largest integer such that $\mathit{QSeg}(\sigma_{\leq k}) = \lambda_1, \ldots, \lambda_{j-1}$. Clearly, this means that the last edge of λ_j is $\sigma_k, \langle a_{k+1}^1, a_{k+1}^2 \rangle, \sigma_{k+1}$. Also, $\mathit{residual}(\sigma_{\leq k})$ contains at the end all the edges of λ_j, except the last one. By applying Lemma 1 to $\sigma_{\leq k}$, we obtain $\pi_2' \in \Pi_2^{\mathrm{t}}$ and $\sigma' = \mathit{outcome}^{\mathrm{tf}}(s, \pi_1, \pi_2')$ such that $\sigma' = \mathit{residual}(\sigma_{\leq k}) \cdot \rho$. It remains to prove that, under π_1, the suffix ρ of σ' can be replaced by the edge $\sigma_k, \langle a_{k+1}^1, a_{k+1}^2 \rangle, \sigma_{k+1}$. This is easily obtained by defining π_2'' to coincide with π_2', except for $\pi_2''(\mathit{residual}(\sigma_{\leq k}) \cdot \langle a_{k+1}^1 \rangle) = a_{k+1}^2$. ∎

A.3 Proof of Theorem 7

We prove the statement for $i = 1$, as the other case is symmetrical. We develop our arguments for the finite turn-based game, as the conclusion for the original timed game follows from from Theorem 3. We proceed by complete induction on $n = \sum_{s \in S} |\Gamma_1(s)| - |S|$. When $n = 0$, only one move is available to player 1 at each state and there is obviously a memoryless optimal strategy.

Suppose that the statement is true for all integers up to $n \geq 0$, and consider the situation where $\sum_{s \in S} |\Gamma_1(s)| - |S| = n + 1$; in this case, there is at least one state, s, where $|\Gamma_1(s)| > 1$. In this game structure (call it $\mathscr{G}$), we can play the s-forgetful game, and Lemma 3 states that the game value will be the same as that of the normal turn-based finite game, $v_1^{\mathrm{tf},s}(\mathscr{G}, t) = v_1^{\mathrm{tf}}(\mathscr{G}, t)$, which can be ensured by some strategy, π_1. By Theorems 2 and 5, $v_1^{\mathrm{tf}}(\mathscr{G}, t) = v_1^{\mathrm{t}\infty}(\mathscr{G}, t) = v_1(\mathscr{G}, t)$, so strategy π_1 is able to ensure $v_1(\mathscr{G}, t)$ in the s-forgetful game. Let $v = v_1(\mathscr{G}, t)$.

Clearly, π_1 only plays one move at state s in the s-forgetful game (call this move $a \in \Gamma_1(s)$), because any return to s would end the game. Therefore, we can set $\Gamma_1'(s) = \{a\}$ (and leave the enabled moves for all other states the same) as part of a new discrete-time game structure $\mathscr{G}' = (S, \mathit{Acts}_1, \mathit{Acts}_2, \Gamma_1', \Gamma_2, \delta, r)$, and still be consistent with how π_1 desires to play the s-forgetful game.

Since $\mathscr{G}'$ is consistent with π_1 in the s-forgetful game, π_1 is able to ensure at least the same value as in $\mathscr{G}$, namely v. Moreover, since $\mathscr{G}'$ contains less choices for player 1, no player 1 strategy can achieve a value greater than v. So, $v_1^{\mathrm{tf},S}(\mathscr{G}',t) = v$. By again appealing to Lemma 3 and Theorems 2 and 5, we find that $v = v_1^{\mathrm{tf},S}(\mathscr{G}',t) = v_1^{\mathrm{tf}}(\mathscr{G}',t) = v_1^{\mathrm{t}\infty}(\mathscr{G}',t) = v_1(\mathscr{G}',t)$.

To summarize, we have constructed a reduced game structure, $\mathscr{G}'$, with value $v_1(\mathscr{G}',t) = v$. Notice that our inductive hypothesis applies to $\mathscr{G}'$, since $\sum_{s\in S}|\Gamma_1'(s)| - |S| \leq n$. Therefore, there exists some memoryless strategy, $\pi_1^\star$, which is able to ensure v when played on $\mathscr{G}'$. But then $\pi_1^\star$ is also able to ensure v in the original $\mathscr{G}$ since all moves that $\pi_1^\star$ might play exist in $\mathscr{G}$. This demonstrates the desired conclusion for player 1. ∎