Measuring Permissivity in Finite Games

Patricia Bouyer[1], Marie Duflot[2], Nicolas Markey[1], and Gabriel Renault[3]

[1] LSV, CNRS & ENS Cachan, France
{bouyer,markey}@lsv.ens-cachan.fr
[2] LACL, Université Paris 12, France
duflot@univ-paris12.fr
[3] Département Informatique, ENS Lyon, France
gabriel.renault@ens-lyon.fr

Abstract. In this paper, we extend the classical notion of strategies in turn-based finite games by allowing several moves to be selected. We define and study a quantitative measure for permissivity of such strategies by assigning penalties when blocking transitions. We prove that for reachability objectives, most permissive strategies exist, can be chosen memoryless, and can be computed in polynomial time, while it is in $\mathsf{NP} \cap \mathsf{coNP}$ for discounted and mean penalties.

1 Introduction

Finite games. Finite games have found numerous applications in computer science [Tho02]. They extend finite automata with several players interacting on the sequence of transitions being fired. This provides a convenient way for reasoning about open systems (subject to *uncontrollable actions* of their environment), and for verifying their correctness. In that setting, *correctness* generally means the existence of a controller under which the system always behaves according to a given specification. A controller, in that terminology, is nothing but a strategy in the corresponding game, played on the automaton of the system, against the environment.

Our framework. In this paper, we propose a new framework for computing *permissive* controllers in finite-state systems. We assume the framework of two-player turn-based games (where the players are Player $\Diamond$ and Player $\Box$, with the controller corresponding to Player $\Diamond$). The classical notion of (deterministic) strategy in finite (turn-based) games is extended into the notion of *multi-strategy*, which allows several edges to be enabled. The permissivity of such a multi-strategy is then measured by associating penalties to blocking edges (each edge may have a different penalty). A strategy is more permissive than an other one if its penalty is weaker, *i.e.*, if it blocks fewer (or less expensive) edges.

We focus on reachability objectives for the controller, that is, the first aim of Player $\Diamond$ will be to reach a designated set of winning states (whatever Player $\Box$ does). The second aim of Player $\Diamond$ will be to minimize the penalty assigned to the set of outcomes generated by the multi-strategy.

Formally we consider *weighted (finite) games*, which are turn-based finite games with non-negative weights on edges. In each state, the penalty assigned to a multi-strategy is the sum of the weights of the edges *blocked* by the multi-strategy. Several ways of measuring the penalty of a strategy can then be considered: in this paper, we consider three ways of counting penalties along outcomes (sum, discounted sum, and mean value) and then set the penalty of a multi-strategy as the maximal penalty of its outcomes.

We will be interested in several problems: (*i*) does Player $\Diamond$ have a winning multi-strategy for which the penalty is no more than a given threshold? (*ii*) compute the infimal penalty that Player $\Diamond$ can ensure while reaching her goal; (*iii*) synthesize (almost-)optimal winning multi-strategies, and characterize them (in terms of memory and regularity).

Our results. We first prove that our games with penalties can be transformed into classical weighted games [ZP96,LMO06] with an exponential blowup, and that the converse reduction can be polynomial.

Then, we prove that we can compute optimal and memoryless multi-strategies for optimal reachability in PTIME. The proof is in three steps: first, using our transformation to weighted games and results of [LMO06], we obtain the existence of optimal memoryless multi-strategies; we then propose a polynomial-time algorithm for computing an optimal winning multi-strategy *with* memory; finally, we show how we can get rid of the memory in such a multi-strategy, which yields the expected result.

We then focus on two other ways of computing penalties, namely the discounted sum and the mean value, and we prove that optimal multi-strategies may not exist, or may require memory. We further prove that we can compute the optimal discounted penalty in $\mathsf{NP} \cap \mathsf{coNP}$, and that we can search for almost-optimal winning multi-strategies as a pair (σ_1, σ_2) of memoryless multi-strategies and that we need to play σ_1 for some time before following σ_2 in order to reach the goal. The longer we play σ_1, the closer we end up to the optimal discounted penalty. The same holds for the mean penalty before reaching the goal.

As side-results, we obtain the complexity of computing strategies in weighted games with a combined objective of reaching a goal state and optimizing the accumulated cost. This can be seen as the *game version* of the shortest path problem in weighted automata. Regarding "plain" costs, this was already a by-product of [LMO06]; we show here that for discounted and mean costs, optimal or memoryless optimal strategies do not necessarily exist, but almost-optimal strategies can be obtained as a "pair" of memoryless strategies.

Related work. This quantitative approach to permissivity is rather original, and does not compare to either of the approaches found in the literature [BJW02,PR05]. Indeed classical notions of permissivity imply the largest sets of generated plays. This is not the case here, where an early cut of a branch/edge of the game may avoid a large penalty later on for blocking many edges. However our notion of multi-strategy coincides with the non-deterministic strategies of [BJW02] and [Lut08].

Our work also meets the problem proposed in [CHJ05] of considering mixed winning objectives, one which is qualitative (parity in their special case), and one which is quantitative (mean-payoff in their special case). The same kind of mixed objectives is considered when extending ATL with quantitative constraints [LMO06,HP06].

The rest of this paper is organized as follows: in the next section, we introduce our formalism of multi-strategies and penalties. We also explain the link with the classical framework of games with costs. Section 3 is devoted to our polynomial-time algorithm for computing most permissive strategies. Section 4 deals with the case of discounted and mean penalty. By lack of space, several proofs have been postponed to the technical appendix.

2 Weighted Games with Reachability Objectives

2.1 Basic definitions

Weighted games. A *(finite) weighted game* is a tuple $\mathsf{G} = (V_\Box, V_\Diamond, E, \mathsf{weight})$ where $V_\Box$ and $V_\Diamond$ are finite sets of states (said to belong to Player $\Box$ and Player $\Diamond$, resp.); writing $V = V_\Box \cup V_\Diamond \cup \{\☺, \☹\}$ where ☺ and ☹ are two distinguished states not belonging to $V_\Box \cup V_\Diamond$, $E \subseteq V \times V$ is a finite set of edges; and $\mathsf{weight}\colon E \to \mathbb{N}$ is a function assigning a weight to every edge. We assume (w.l.o.g.) that the states ☺ and ☹ have no outgoing edges (they are respectively the winning and losing states). If $v \in V$, we write vE (resp. Ev) for $E \cap (\{v\} \times V)$ (resp. $E \cap (V \times \{v\})$) for the set of edges originating from (resp. targetting to) v.

A *run* ϱ in G is a finite or infinite sequence of states $(v_i)_{0 \le i \le p}$ (for some $p \in \mathbb{N} \cup \{\infty\}$) such that $e_i = (v_{i-1}, v_i) \in E$ when $p \ge i > 0$. We may also write for such a run $\varrho = (v_0 \to v_1 \to v_2 \cdots)$, or $\varrho = (e_i)_{i \ge 1}$ [4], or the word $\varrho = v_0\, v_1\, v_2 \ldots$ The length of ϱ, denoted by $|\varrho|$, is $p+1$. For finite-length runs, we write $\mathsf{last}(\varrho)$ for the last state v_p. Given $r < |\varrho|$, the r-th prefix of $\varrho = (v_i)_{0 \le i \le p}$ is the run $\varrho_{\le r} = (v_i)_{0 \le i \le r}$. Given a finite run $\varrho = (v_i)_{0 \le i \le p}$ and a transition $e = (v, v')$ with $v = v_p$, we write $\varrho \xrightarrow{e}$, or $\varrho \to v'$, for the run $\varrho = (v_i)_{0 \le i \le p+1}$ with $v_{p+1} = v'$.

We write $\mathsf{Runs}_\mathsf{G}^{<\omega}$ (resp. $\mathsf{Runs}_\mathsf{G}^{\omega}$) for the set of finite (resp. infinite) runs in G, and $\mathsf{Runs}_\mathsf{G} = \mathsf{Runs}_\mathsf{G}^{<\omega} \cup \mathsf{Runs}_\mathsf{G}^{\omega}$. In the sequel, we omit the subscript G when no ambiguity may arise.

Multi-strategies. A *multi-strategy* for Player $\Diamond$ is a function

$$\sigma\colon \left\{\varrho \in \mathsf{Runs}^{<\omega} \mid \mathsf{last}(\varrho) \in V_\Diamond\right\} \to 2^E$$

such that, for all $\varrho \in \mathsf{Runs}^{<\omega}$, we have $\sigma(\varrho) \subseteq vE$ with $v = \mathsf{last}(\varrho)$. A multi-strategy is *memoryless* if $\sigma(\varrho) = \sigma(\varrho')$ as soon as $\mathsf{last}(\varrho) = \mathsf{last}(\varrho')$. A memoryless multi-strategy σ can be equivalently represented as a mapping $\sigma'\colon V_\Diamond \to 2^E$, with $\sigma(\varrho) = \sigma'(\mathsf{last}(\varrho))$.

Multi-strategies extend the usual notion of *strategies* by selecting several possible moves (classically, a strategy is a multi-strategy σ such that for every

[4] These notations are equivalent since we assume that there can only be one edge between two states.

$\varrho \in \mathsf{Runs}^{<\omega}$ with $\mathsf{last}(\varrho) \in V_\Diamond$, the set $\sigma(\varrho)$ is a singleton). The aim of this paper is to compare multi-strategies and to define and study a notion of *permissivity* of a multi-strategy.

Given a multi-strategy σ for Player $\Diamond$, the set of outcomes of σ, denoted $\mathsf{Out}(\sigma) \subseteq \mathsf{Runs}$, is defined as follows:

- for every state $v \in V$, the run v is in $\mathsf{Out}^{<\omega}(\sigma)$;
- if $\varrho \in \mathsf{Out}^{<\omega}(\sigma)$ and $\sigma(\varrho)$ is defined and non-empty, then for every $e \in \sigma(\varrho)$, the run $\varrho \xrightarrow{e}$ is in $\mathsf{Out}^{<\omega}(\sigma)$;
- if $\varrho \in \mathsf{Out}^{<\omega}(\sigma)$ and $\mathsf{last}(\varrho) = v \in V_\Box$, then for every edge $e \in vE$, the run $\varrho \xrightarrow{e}$ is in $\mathsf{Out}^{<\omega}(\sigma)$;
- if $\varrho \in \mathsf{Runs}^{\omega}$ and if all finite prefixes ϱ' of ϱ are in $\mathsf{Out}^{<\omega}(\sigma)$, then $\varrho \in \mathsf{Out}^{\omega}(\sigma)$.

We write $\mathsf{Out}(\sigma) = \mathsf{Out}^{<\omega}(\sigma) \cup \mathsf{Out}^{\omega}(\sigma)$. A run ϱ in $\mathsf{Out}(\sigma)$ is *maximal* whenever it is infinite, or it is finite and either $\sigma(\varrho) = \varnothing$, or $\mathsf{last}(\varrho)$ has no outgoing edge (*i.e.*, the set vE, with $v = \mathsf{last}(\varrho)$, is empty). If ϱ_0 is a finite outcome of σ, we write $\mathsf{Out}(\sigma, \varrho_0)$ (resp. $\mathsf{Out}^{\max}(\sigma, \varrho_0)$) for the set of outcomes (resp. maximal outcomes) of σ having ϱ_0 as a prefix. A multi-strategy σ is *winning* after ϱ_0 if every run $\varrho \in \mathsf{Out}^{\max}(\sigma, \varrho_0)$ is finite and has $\mathsf{last}(\varrho) = ☺$. A finite run ϱ_0 is *winning* if it admits a winning multi-strategy after ϱ_0. Last, a strategy is winning if it is winning from any winning state (seen as a finite run).

Penalties for multi-strategies. We define a notion of permissivity of a multi-strategy by counting the weight of transitions that the multi-strategy blocks along its outcomes. If σ is a multi-strategy and ϱ_0 is a finite run, the *penalty* of σ after ϱ_0, denoted $\mathsf{penalty}(\sigma, \varrho_0)$, is defined as $\sup\{\mathsf{penalty}_{\sigma,\varrho_0}(\varrho) \mid \varrho \in \mathsf{Out}^{\max}(\sigma, \varrho_0)\}$ where $\mathsf{penalty}_{\sigma,\varrho_0}(\varrho)$ is defined inductively, for every finite run $\varrho \in \mathsf{Out}(\sigma, \varrho_0)$, by:

- $\mathsf{penalty}_{\sigma,\varrho_0}(\varrho_0) = 0$;
- if $\mathsf{last}(\varrho) \notin V_\Diamond$ and $(\mathsf{last}(\varrho), v) \in E$, then $\mathsf{penalty}_{\sigma,\varrho_0}(\varrho \to v) = \mathsf{penalty}_{\sigma,\varrho_0}(\varrho)$;
- if $\mathsf{last}(\varrho) \in V_\Diamond$ and $(\mathsf{last}(\varrho), v) \in \sigma(\varrho)$, then

$$\mathsf{penalty}_{\sigma,\varrho_0}(\varrho \to v) \;=\; \mathsf{penalty}_{\sigma,\varrho_0}(\varrho) \;+ \sum_{(\mathsf{last}(\varrho),v') \in (E \smallsetminus \sigma(\varrho))} \mathsf{weight}(\mathsf{last}(\varrho), v');$$

- if $\varrho \in \mathsf{Out}(\sigma, \varrho_0) \cap \mathsf{Runs}^{\omega}$, then $\mathsf{penalty}_{\sigma,\varrho_0}(\varrho) = \lim_{n \to +\infty} \mathsf{penalty}_{\sigma,\varrho_0}(\varrho_{\leq n})$.

The first objective of Player $\Diamond$ is to win the game (*i.e.*, reach ☺), and her second objective is to minimize the penalty. In our formulation of the problem, Player $\Box$ has no formal objective, but her aim is to play against Player $\Diamond$ (this is a zero-sum game), which implicitly means that Player $\Box$ tries to avoid reaching ☺, and if this is not possible, she tries to maximize the penalty before reaching ☺.

We write $\mathsf{opt\_penalty}(\varrho_0)$ for the optimal penalty Player $\Diamond$ can ensure after ϱ_0 while reaching ☺:

$$\mathsf{opt\_penalty}(\varrho_0) = \inf\{\mathsf{penalty}(\sigma', \varrho_0) \mid \sigma' \text{ winning multi-strategy after } \varrho_0\}.$$

It is equal to $+\infty$ if and only if Player $\Diamond$ has no winning multi-strategy after ϱ_0.

The following lemma is rather obvious, and shows that we only need to deal with the optimal penalty from a state.

Lemma 1. *Let* G *be a weighted game, let* ϱ *and* ϱ' *be two runs in* G *such that* $\mathsf{last}(\varrho) = \mathsf{last}(\varrho')$. *Then* $\mathsf{opt\_penalty}(\varrho) = \mathsf{opt\_penalty}(\varrho')$.

Given $\varepsilon \geq 0$, a winning multi-strategy σ is *ε-optimal after* ϱ_0 if $\mathsf{penalty}(\sigma, \varrho_0) \leq \mathsf{opt\_penalty}(\varrho_0) + \varepsilon$. It is *optimal after* ϱ_0 when it is 0-optimal after ϱ_0. If σ is ε-optimal from any winning state, then we say that σ is ε-optimal.

Classical weighted games. This way of associating values to runs and (multi-) strategies is rather non-standard, and usually it is rather a notion of *accumulated cost* along the runs which is considered. It is defined inductively as follows:

- $\mathsf{cost}(v) = 0$ for single-state runs;
- $\mathsf{cost}(\varrho \xrightarrow{e}) = \mathsf{cost}(\varrho) + \mathsf{weight}(e)$ otherwise.

Then again, if σ is a multi-strategy and ϱ_0 is a finite outcome, $\mathsf{cost}(\sigma, \varrho_0) = \sup\{\mathsf{cost}(\varrho) - \mathsf{cost}(\varrho_0) \mid \varrho \in \mathsf{Out}^{\max}(\sigma, \varrho_0)\}$, and notions of ($\varepsilon$-)optimal strategies are defined in the expected way.

Example 1. A weighted game is depicted on Fig. 1. For this example, it can be easily seen that the optimal strategy w.r.t. costs from state a consists in going through b, resulting in a weight of 6.

Regarding penalties and multi-strategies, the situation is more difficult. From state b, there is only one way of winning, with penalty 6 (because the strategy *must* block the transition to the losing state). From d, we have two possible winning multi-strategies: either block the transition to b, with penalty 2, or keep it; in the latter case, we will then have penalty 6 in state d, as explained above. In d, the best multi-strategy thus amounts to blocking the transition to b, so that we can win with penalty 2. Now, from a, it seems natural to try to go winning *via* d. This requires blocking both transitions to b and c, and results in a global penalty of $8 (= 5+1+2)$ for winning. However, allowing both transitions to b and d is better, as the global (worst case) penalty in this case is $7 (= 1 + 6)$. Note that in that case, it is also possible to allow transition to c for some time, since the loop between a and c will add no extra penalty. But if we allow it forever, it will not be winning, this transition to c has thus to be blocked at some point in order to win.

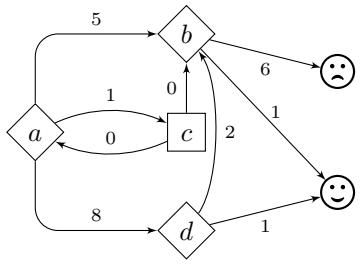

Fig. 1. A weighted game

Computation and decision problems. We let $\mathsf{G} = (V_\Box, V_\Diamond, E, \mathsf{weight})$ be a weighted game. Given $v \in V$, we will be interested in computing the value $\mathsf{opt\_penalty}(v)$, and if an optimal winning multi-strategy exists, in computing it. We will also be interested in computing for every $\varepsilon > 0$, an ε-optimal winning multi-strategy.

Formally, the optimal reachability problem with penalty we consider is the following: given a weighted game G, a rational number c and a state $v \in V$, does there exist a multi-strategy σ for Player $\Diamond$ such that $\mathsf{penalty}(\sigma, v) \leq c$.

2.2 From penalties to costs, and back

Penalties and costs assume very different points of view: in particular, cost-optimality can obviously be achieved with "deterministic" strategies (adding extra outcomes can only increase the overall cost of the strategy), while penalty-optimality generally requires multi-strategies. Still, there exists a tight link between both approaches, which we explain now. The proofs may be found in the appendix. We only explain the transformations on two examples (Figs. 2 and 3).

Lemma 2. *For every weighted game* $\mathsf{G} = (V_\Box, V_\Diamond, E, \mathsf{weight})$*, we can construct an exponential-size weighted game* $\mathsf{G}' = (V'_\Box, V'_\Diamond, E', \mathsf{weight}')$ *such that* $V_\Box \subseteq V'_\Box$*,* $V_\Diamond \subseteq V'_\Diamond$ *and, for any state* $v \in V_\Box \cup V_\Diamond$ *and any bound* c*, Player* $\Diamond$ *has a winning multi-strategy with penalty* c *from* v *in* G *iff she has a winning strategy with cost* c *from* v *in* G'*.*

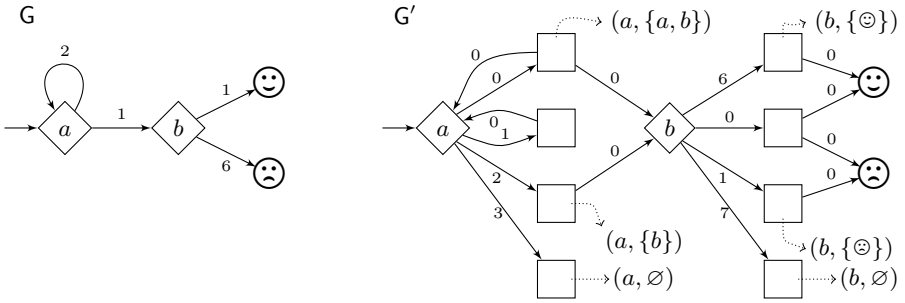

Fig. 2. From penalties (and multi-strategies) to costs (and strategies)

Applying results from [LMO06], we get:

Corollary 3. *For any finite weighted game* G*, there exists a memoryless optimal multi-strategy (w.r.t. penalties).*

Lemma 4. *For every weighted game* $\mathsf{G}' = (V'_\Box, V'_\Diamond, E', \mathsf{weight}')$*, we can construct a polynomial-size weighted game* $\mathsf{G} = (V_\Box, V_\Diamond, E, \mathsf{weight})$ *such that* $V'_\Box \subseteq V_\Box$*,* $V'_\Diamond \subseteq V_\Diamond$*, and for any state* $v \in V'_\Box \cup V'_\Diamond$ *and any value* c*, Player* $\Diamond$ *has a winning strategy with cost* c *from* v *in* G' *iff she has a winning multi-strategy with penalty* c *from* v *in* G*.*

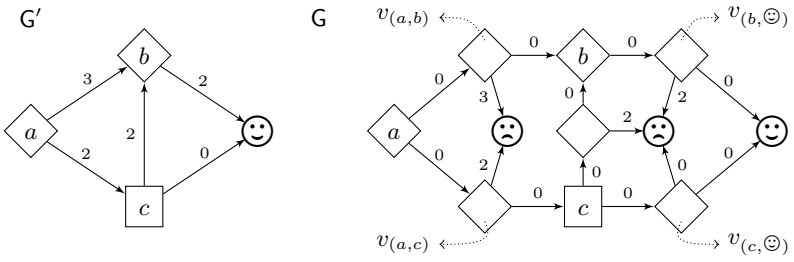

Fig. 3. From costs (and strategies) to penalties (and multi-strategies)

3 Optimal Reachability in Penalty Games

Corollary 3 would allow us to solve the optimal reachability problem with penalty in NP: it suffices to guess a memoryless multi-strategy, and check that it is winning and has penalty less than the given threshold. We prove here that it can be done more efficiently, *via* a two-step algorithm: we first build an optimal strategy, and then make it memoryless. Both steps will be in PTIME.

In the sequel, we let $\mathsf{G} = (V_\Box, V_\Diamond, E, \mathsf{weight})$ be a weighted game.

3.1 Construction of an optimal winning multi-strategy

In this section, we give a polynomial-time algorithm for computing an optimal winning multi-strategy (which requires memory). The idea is to inductively compute the penalty for winning in j steps, for each j less than the number of states. This will be sufficient as we know that there exists a memoryless optimal multi-strategy (remember Corollary 3), which wins in $|V|$ from the winning states.

Due to the transformation presented in Lemma 2, there is *a priori* an exponential blowup for computing the best move in one step (because Player $\Diamond$ can select any subset of the outgoing edges of the current state, and will choose 'the best' subset), but we will show that choices satisfy some monotonicity property that will help making the best choice in polynomial time only.

For any integer k, we say that a multi-strategy σ is k-step if, for every run ϱ of length (strictly) larger than k with $\mathsf{last}(\varrho) \in V_\Diamond$, we have $\sigma(\varrho) = \varnothing$. For instance, a memoryless winning multi-strategy σ' naturally induces a $|V|$-step winning multi-strategy (all outcomes of σ' have length $\leq |V|$ and for all the other (useless) runs we can set $\sigma(\varrho) = \varnothing$). We say that a state v is winning in k steps if there is a k-step multi-strategy which is winning from v.

The algorithm will proceed as follows: for every $0 \leq j \leq |V|$, we build a j-step multi-strategy σ_j which will be winning from all states that are winning in j steps, and optimal among all those winning j-step multi-strategies. We also compute, for each state $v \in V$, a value $c_{v,j}$ which is either the penalty of strategy σ_j from v (*i.e.* $\mathsf{penalty}(\sigma_j, v)$), or $+\infty$ in case σ_j is not winning from v.

Since we know that memoryless multi-strategies suffice to win optimally, we conclude that there exists a $|V|$-step multi-strategy, which is winning and optimal, and the multi-strategy $\sigma_{|V|}$ which we build will then be optimal and winning. It follows that $c_{v,|V|}$ will be equal to $\mathsf{opt\_penalty}(v)$.

When $j = 0$, we let $\sigma_0(\varrho) = \varnothing$ for any ϱ ending in a $V_\Diamond$-state. It is the only 0-step multi-strategy, so that it clearly is optimal among these. Clearly, $c_{v,0} = +\infty$ for all $v \neq \smiley$, $c_{\smiley,0} = 0$, and $\smiley$ is the only state from which we can win with a 0-step multi-strategy.

We assume we have built σ_j ($0 \leq j < |V|$), and we now define σ_{j+1}. Let $\varrho = v_0 \to v_1 \to v_2 \ldots \to v_k$ be a run ending in $V_\Diamond$. If $k \geq j+1$, we let $\sigma_{j+1}(\varrho) = \varnothing$. Otherwise, if $k \geq 1$, we let $\sigma_{j+1}(v_0 \to v_1 \to v_2 \ldots \to v_k) = \sigma_j(v_1 \to v_2 \ldots \to v_k)$. Finally, when $k = 0$ and $\varrho = v$, we let $\{u_1, \ldots, u_p\}$ be the set of successors

of v, assuming that they are ordered in such a way that $c_{u_r,j} \leq c_{u_s,j}$ if $r \leq s$. Now, let

$$f_{v,j+1} \colon I \subseteq [\![1,p]\!] \mapsto \sum_{s \notin I} \mathsf{weight}(v,u_s) + \max_{s \in I} c_{u_s,j},$$

and let $I \neq \varnothing$ be a subset of $[\![1,p]\!]$ realizing the minimum of $f_{v,j+1}$ over the non-empty subsets of $[\![1,p]\!]$. Assume that there exist two integers $l < m$ in $[\![1,p]\!]$ such that $l \notin I$ and $m \in I$. Since $u_l \leq u_m$, we have

$$f_{v,j+1}(I \cup \{l\}) - f_{v,j+1}(I) = -\mathsf{weight}(v,u_l).$$

This entails that $I \cup \{l\}$ is also optimal. By repeating the process, we can prove that there exists an interval $[\![1,q]\!]$ realizing the minimum of $f_{v,j+1}$. As a consequence, finding the minimum of $f_{v,j+1}$ can be done in polynomial time (by checking all intervals of the form $[\![1,q]\!]$). We write $T_{v,j+1}$ for a corresponding set of states, whose indices realize the minimum of $f_{v,j+1}$. We then define $\sigma_{j+1}(v) = \{(v,v') \mid v' \in T_{v,j+1}\}$, and $c_{v,j+1} = f_{v,j+1}(T_{v,j+1})$ for all $v \in V_\diamond$. It is easy to check that $c_{v,j+1} = \mathsf{penalty}(\sigma_{j+1}, v)$ if σ_{j+1} is winning from v, and $c_{v,j+1} = +\infty$ otherwise.

We now prove that for every $0 \leq j \leq |V|$, σ_j is optimal among all j-step winning multi-strategies. Assume that, for some $0 \leq j \leq |V|$, there is a j-step multi-strategy σ' that is winning and strictly better than σ_j from some winning state v. We pick the smallest such index j. We must have $j > 0$ since σ_0 is optimal among the 0-step multi-strategies. Consider the set of successors $\{u_1, \ldots, u_p\}$ of v ordered as above, and let T be the set of indices such that $\sigma'(v) = \{(v,u_t) \mid t \in T\}$. Then after one step, the multi-strategy σ' is $(j-1)$-step and winning from any u_t satisfying $(v,u_t) \in \sigma'(v)$, and its penalty is thus not smaller than that of the multi-strategy σ_{j-1} (by minimality of j, we have $\mathsf{penalty}(\sigma', v \to u_t) \geq c_{u_t,j-1}$). Hence:

$$\mathsf{penalty}(\sigma', v) \geq \sum_{s \notin T} \mathsf{weight}(v,u_s) + \max_{t \in T} c_{u_t,j-1} = f_{v,j}(T)$$

On the other hand, as σ' is strictly better than σ_j we must have

$$\mathsf{penalty}(\sigma', v) < c_{v,j} = f_{v,j}(T_{v,j}) \leq f_{v,j}(T)$$

because $T_{v,j}$ achieves the minimum of $f_{v,j}$. This is a contradiction, and from every state v from which there is a j-step winning multi-strategy, σ_j is winning optimally (in j steps).

As stated earlier, due to the existence of memoryless optimal winning multi-strategies, $|V|$-step multi-strategies are sufficient and $\sigma_{|V|}$ is optimal winning. ⌟

3.2 Deriving a memoryless winning multi-strategy

In this section we compute, from any winning multi-strategy σ, a memoryless winning multi-strategy σ' which has lower penalty for Player $\diamond$. The idea is to represent σ as a (finite) forest (it is finite because σ is winning) where a node corresponds to a finite outcome, and to select a state v for which σ is not memoryless yet. This state should be chosen carefully[5] so that we will be able

[5] An appropriate measure will be assigned to every node of the forest.

to "plug" the subtree (*i.e.*, play the multi-strategy) rooted at some node ending in v at all nodes ending in v while preserving the winningness of all states, and while decreasing (or at least leaving unchanged) the penalty of all states. This transformation will be repeated until the multi-strategy is memoryless from all states. That way, if σ was originally optimal, then so will σ' be.

Let Σ be a finite alphabet. A *Σ-forest* is a tuple $\mathcal{T} = (T, R)$ where $T \subseteq \Sigma^+$ is a set of non-empty finite words on Σ (called *nodes*) such that, for each $t \cdot a \in T$ with $a \in \Sigma$ and $t \in \Sigma^+$, it holds $t \in T$ (T is closed by non-empty prefix) ; $R \subseteq \Sigma \cap T$ is the set of roots. Given $a \in \Sigma$, a node t such that $t = u \cdot a$ is called an *occurrence* of a. Given a node $t \in T$, the *depth* of t is $|t| - 1$ (where $|t|$ is the length of t seen as a word on Σ), and its height, denoted $\mathit{height}_{\mathcal{T}}(t)$, is

$$\sup\{|u| \mid u \in \Sigma^* \text{ and } t \cdot u \in T\}.$$

In particular, $\mathit{height}_{\mathcal{T}}(t) = +\infty$ when t is the prefix of an infinite branch in $\mathcal{T}$.

A Σ-tree is a Σ-forest with one single root. Given a forest $\mathcal{T} = (T, R)$ and a node $t \in T$, the *subtree of $\mathcal{T}$ rooted at t* is the tree $\mathcal{S} = (S, \{n\})$ where $n = \mathsf{last}(t)$ and $s \in S$ iff $t \cdot s \in T$.

Let $\mathsf{G} = (V_\Box, V_\Diamond, E, \mathsf{weight})$ be a weighted game. A winning multi-strategy σ for Player $\Diamond$ in G and a winning state $v \in V$ naturally define a finite V-tree $\mathcal{T}_{\sigma,v}$ with root v: given a state s, a word $t = u \cdot s$ is in $\mathcal{T}_{\sigma,v}$ iff $u \in \mathcal{T}_{\sigma,v}$ and, seeing u as a finite run, we have either $\mathsf{last}(u) = v' \in V_\Diamond$ and $(v', s) \in \sigma(u)$, or $\mathsf{last}(u) = v' \in V_\Box$ and $(v', s) \in E$. In this tree, the height of the root coincides with the length of a longest run generated by the multi-strategy σ from v. Since the multi-strategy σ is winning from v, all branches are finite, and all leaves of $\mathcal{T}_{\sigma,v}$ are occurrences of ☺. The union of all trees $\mathcal{T}_{\sigma,v}$ (for v a winning state) defines a forest $\mathcal{T}_\sigma$.

Conversely, every V-forest $\mathcal{T} = (T, W)$ with $W \subseteq V$ satisfying the following conditions naturally defines a winning multi-strategy $\sigma_{\mathcal{T}}$ (viewing each node $t \in T$ as a run of G):

- if $\mathsf{last}(t) = v' \in V_\Box$, $t \cdot s \in T$ iff $(v', s) \in E$;
- if $\mathsf{last}(t) = v' \in V_\Diamond$ and $t \cdot s \in T$, then $(v', s) \in E$. In that case we set $\sigma_{\mathcal{T}}(t) = \{(v', s) \in E \mid t \cdot s \in T\}$;
- if t is maximal, then $\mathsf{last}(t) = ☺$.

Lemma 5. *Assume that we are given an optimal winning multi-strategy σ. We can effectively construct in polynomial time a memoryless multi-strategy σ', which is winning and optimal.*

Proof. Assume that W is the set of winning states. Let $\mathcal{T}$ be the forest representing the multi-strategy σ (its set of roots is W). Since σ is winning from every state in W, all branches of the forest are finite. For every node t of $\mathcal{T}$, we define $\gamma_{\mathcal{T}}(t)$ as the *residual penalty* of σ after prefix t. Formally, $\gamma_{\mathcal{T}}(t) = \mathsf{penalty}(\sigma, t)$. Obviously, for all $v \in V$, we have $\mathsf{penalty}(\sigma, v) = \gamma_{\mathcal{T}}(v)$.

We will consider a measure $\mu_{\mathcal{T}}$ on the set of nodes of the forest $\mathcal{T}$ as follows: if t is a node of $\mathcal{T}$, we let $\mu_{\mathcal{T}}(t) = (\gamma_{\mathcal{T}}(t), \mathit{height}_{\mathcal{T}}(t))$.

We say that *no memory is required* for state v in $\mathcal{T}$ if, for every two nodes t and t' that are occurrences of v, the subtree of $\mathcal{T}$ rooted at t and the subtree of $\mathcal{T}$ rooted at t' are identical. Note that in that case, $\mu_{\mathcal{T}}(t) = \mu_{\mathcal{T}}(t')$.

For every $0 \leq i \leq |W|$, we inductively build (see Appendix) in polynomial time a forest $\mathcal{T}^i$ and a set $M_i \subseteq W$ containing i elements, such that:

(a) $\mathcal{T}^i$ represents an optimal winning multi-strategy from all the states of W;
(b) for every $v \in M_i$, no memory is required for v in $\mathcal{T}^i$, and for every node t' which is a descendant of some node that is an occurrence of v, letting $v' = \mathsf{last}(t')$, it holds $v' \in M_i$.

Intuitively, each $\mathcal{T}^i$ will be the forest of a winning optimal multi-strategy σ_i, and each M_i will be a set of states from which σ_i is memoryless (*i.e.*, σ_i is memoryless from the states in M_i, and from the states that occur in the outcomes from these states). In the end, the forest $\mathcal{T}^{|W|}$ represents a multi-strategy σ' which is memoryless, optimal and winning from every state of the game. ⌟

4 Discounted and Mean Penalty Games

4.1 Discounted and mean penalties of multi-strategies

We have proposed a way to measure the permissivity of winning strategies in games, by summing penalties for blocking edges in the graph. It can be interesting to consider that blocking an edge *early* in a run is more restrictive than blocking an edge *later*. A classical way to represent this is to consider a *discounted* version of the penalty of a multi-strategy, which we now define.

Discounted penalties of multi-strategies. Let $\mathsf{G} = (V_\Box, V_\Diamond, E, \mathsf{weight})$ be a weighted game, σ be a winning (w.r.t. the reachability objective) multi-strategy, and ϱ_0 be a finite outcome of σ. Given a discount factor $\lambda \in (0, 1)$, the *discounted penalty* of σ after ϱ_0 (w.r.t. λ), denoted $\mathsf{penalty}^\lambda(\sigma, \varrho_0)$, is defined as $\sup\{\mathsf{penalty}^\lambda_{\sigma,\varrho_0}(\varrho) \mid \varrho \in \mathsf{Out}^{\max}_{\mathsf{G}}(\sigma, \varrho_0)\}$, where $\mathsf{penalty}^\lambda_{\sigma,\varrho_0}(\varrho)$ is inductively defined for all $\varrho \in \mathsf{Out}_{\mathsf{G}}(\sigma, \varrho_0)$ as follows:

- $\mathsf{penalty}^\lambda_{\sigma,\varrho_0}(\varrho_0) = 0$;
- if $\mathsf{last}(\varrho) \notin V_\Diamond$ and $(\mathsf{last}(\varrho), v) \in E$, then $\mathsf{penalty}^\lambda_{\sigma,\varrho_0}(\varrho \to v) = \mathsf{penalty}^\lambda_{\sigma,\varrho_0}(\varrho)$;
- if $\mathsf{last}(\varrho) \in V_\Diamond$ and $(\mathsf{last}(\varrho), v) \in \sigma(\varrho)$, then $\mathsf{penalty}^\lambda_{\sigma,\varrho_0}(\varrho \to v)$ is defined as

$$\mathsf{penalty}^\lambda_{\sigma,\varrho_0}(\varrho) \;+\; \lambda^{|\varrho|-|\varrho_0|} \cdot \sum_{(\mathsf{last}(\varrho),v') \in (E \setminus \sigma(\varrho))} \mathsf{weight}(\mathsf{last}(\varrho), v').$$

We also define the discounted penalty along infinite runs, as being the limit (which necessarily exists as $\lambda < 1$) of the penalties along the finite prefixes.

We write $\mathsf{opt\_penalty}^\lambda(\varrho_0)$ for the optimal discounted penalty (w.r.t. λ) Player $\Diamond$ can ensure after ϱ_0 while reaching ☺:

$$\mathsf{opt\_penalty}^\lambda(\varrho_0) = \inf\{\mathsf{penalty}^\lambda(\sigma, \varrho_0) \mid \sigma \text{ winning multi-strategy after } \varrho_0\}.$$

Given $\varepsilon \geq 0$ and $\lambda \in (0,1)$, a multi-strategy σ is said *ε-optimal* for discount factor λ after ϱ_0 if it is winning after ϱ_0 and

$$\mathsf{penalty}^\lambda(\sigma, \varrho_0) \leq \mathsf{opt\_penalty}^\lambda(\varrho_0) + \varepsilon.$$

Again, optimality is a shorthand for 0-optimality. Finally, a multi-strategy is ε-optimal for discount factor λ if it is ε-optimal for λ from any winning state.

Discounted cost in weighted games. As in Section 2.1, we recall the usual notion cost^λ of discounted cost of runs in a weighted game [ZP96][6]:

- $\mathsf{cost}^\lambda(v) = 0$;
- $\mathsf{cost}^\lambda(\varrho \xrightarrow{e}) = \mathsf{cost}^\lambda(\varrho) + \lambda^{|\varrho|-1} \cdot \mathsf{weight}(e)$;

Then we define $\mathsf{cost}^\lambda(\sigma, \varrho_0) = \sup\{\mathsf{cost}^\lambda(\varrho) \mid \varrho \in \mathsf{Out}_{\mathsf{G}}^{\max}(\sigma, \varrho_0)\}$. Those games are symmetric, and later we will sometimes take the point-of-view of Player $\Box$ whose objective will be to maximize the discounted cost: given a strategy σ for Player $\Box$, we then define $\mathsf{cost}^\lambda(\sigma, \varrho_0) = \inf\{\mathsf{cost}^\lambda(\varrho) \mid \varrho \in \mathsf{Out}_{\mathsf{G}}^{\max}(\sigma, \varrho_0)\}$.

Computation and decision problems. As in the previous section, our aim is to compute (almost-)optimal multi-strategies. The optimal reachability problem with discounted penalty we consider is the following: given a weighted game G, a rational number c, a discount factor $\lambda \in (0,1)$, and a state $v \in V$, does there exist a multi-strategy σ for Player $\diamond$ such that $\mathsf{penalty}^\lambda(\sigma, v) \leq c$. The transformations between penalties and costs depicted in Section 2.2 are still possible in the discounted setting. The only point is that in both cases, each single transition gives rise to two consecutive transitions, so that we must consider $\sqrt{\lambda}$ as the new discounting factor[7].

4.2 Some examples

As far as the existence of an optimal multi-strategy is concerned, the discounted case is more challenging as the results of previous section do not hold. In particular, we exemplify in this section the fact that optimal multi-strategies do not always exist, and when they exist, they cannot always be made memoryless. Figs 4 and 5 display two such examples. The non-existence proofs are given in the Appendix.

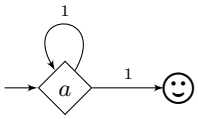

Fig. 4. No optimal discounted multi-strategy

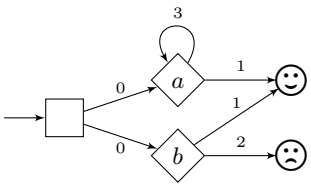

Fig. 5. No memoryless optimal discounted multi-strategy

[6] Notice that we have dropped the normalization factor $(1-\lambda)$, which is only important when we make λ tend to 1 [ZP96].

[7] For the reduction of Lemma 4, the penalty is also multiplied by $\sqrt{\lambda}$

4.3 A pair of memoryless strategies is sufficient

We prove here that there always exist ε-optimal multi-strategies that are made of two memoryless multi-strategies. Roughly, the first multi-strategy aims at lengthening the path (so that the coefficient $\lambda^{|\varrho|}$ will be small) without increasing the penalty, and the second multi-strategy aims at reaching the final state.

To this aim, we need to first study the multi-strategy problem in the setting where there is no reachability objective. Let G be a finite weighted game, $\lambda \in (0,1)$, and $c \in \mathbb{Q}$. The optimal discounted-penalty problem consists in deciding whether there is a multi-strategy for Player $\diamond$ for which the λ-discounted penalty along any maximal (finite or infinite) outcome is less than or equal to c.

Theorem 8. *The optimal discounted-penalty problem is in* $\mathsf{NP} \cap \mathsf{coNP}$*, and is* PTIME*-hard.*

The proof of this theorem relies on known results in classical discounted games [ZP96,Jur98], uses the transformation of Lemma 2 and monotonicity properties already used in the proof given in section 3.1.

Proof. We let $\mathsf{G} = (V_\Box, V_\diamond, E, \mathsf{weight})$ be a finite weighted game with no incoming transitions to $\circlearrowleft$, and let $c \in \mathbb{Q}$. Applying the transformation of Lemma 2 to the discounted case, we get an exponential-size weighted game $\mathsf{G}' = (V'_\Box, V'_\diamond, E', \mathsf{weight}')$ with $V_\Box \subseteq V'_\Box$ and $V_\diamond = V'_\diamond$ such that for every $v \in V_\Box \cup V_\diamond$, Player $\diamond$ has a winning multi-strategy from v in G with discounted penalty no more than c (for discount λ) iff Player $\diamond$ has a winning strategy from v in G' with discounted cost no more than c (for discount $\sqrt{\lambda}$).

From [ZP96], Player $\diamond$ has a memoryless optimal strategy in G'. The NP algorithm is then as follows: guess such a memoryless strategy $\sigma_\diamond$ for Player $\diamond$, *i.e.* for every $v \in V_\diamond$ guess a subset $F \subseteq vE$ and set $\sigma_\diamond(v) = (v, F)$. Removing from G' transitions that have not been chosen by $\sigma_\diamond$ yields a polynomial-size graph G'', in which we can compute in polynomial time the maximal discounted cost, which corresponds to $\mathsf{cost}^{\sqrt{\lambda}}(\sigma_\diamond, v)$. The graph G'' can be computed directly from G without explicitly building G', so that our procedure runs in polynomial time.

The converse is harder. We only sketch the ideas of the proof here, and defer the whole proof in the appendix. The game G' is memoryless determined [ZP96], which means that for every $c \in \mathbb{Q}$, for every state $v \in V'_\Box \cup V'_\diamond$, either Player $\diamond$ has a memoryless strategy $\sigma_\diamond$ with $\mathsf{cost}^{\sqrt{\lambda}}(\sigma_\diamond, v) \leq c$, or Player $\Box$ has a memoryless strategy $\sigma_\Box$ with $\mathsf{cost}^{\sqrt{\lambda}}(\sigma_\Box, v) > c$. Our coNP algorithm consists in guessing a memoryless strategy for Player $\Box$ that achieves cost larger than c. However, Player $\Box$ controls exponentially many states in G', so that we will guess a succinct encoding of her strategy, based on the following observation: there is a (preference) order on the states in $V_\Box \cup V_\diamond$[8] so that, in states of the form (v, F), the optimal strategy for Player $\Box$ consists in playing the "preferred" state of F (w.r.t. the

[8] Which will be given by ordering the values given by the classical optimality equations [Jur98] in G'.

order). In other words, the strategy in those states can be defined in terms of an order on the states, which can be guessed in polynomial time.

Once such a strategy has been chosen non-deterministically, it then suffices to build a polynomial-size graph G'' in which the cost of the strategy $\sigma_\Box$ corresponds to the minimal discounted cost of Player $\Box$ in G'.

Hardness in PTIME directly follows from Lemma 4. ⌟

Remark 1. We could define another version of this problem, with an extra safety condition: the aim is then to minimize the discounted penalty while avoiding some bad states. An easy adaptation of the previous proof yields the very same results for this problem.

Definition 9. *Let σ_1 and σ_2 be two memoryless multi-strategies, and $k \in \mathbb{N}$. The multi-strategy $\sigma = \sigma_1^k \cdot \sigma_2^*$ is defined, for each ϱ such that $\mathsf{last}(\varrho) \in V_\Diamond$, as:*

- *if $|\varrho| < k$, then $\sigma(\varrho) = \sigma_1(\varrho)$;*
- *if $|\varrho| \geq k$, then $\sigma(\varrho) = \sigma_2(\varrho)$.*

We now prove the result mentioned above:

Theorem 10. *Let $\mathsf{G} = (V_\Box, V_\Diamond, E, \mathsf{weight})$ be a finite weighted game with a reachability objective, and $\lambda \in (0,1)$. Then there exist two memoryless multi-strategies σ_1 and σ_2 such that, for any $\varepsilon > 0$, there is an integer k such that the multi-strategy $\sigma_1^{k'} \cdot \sigma_2^*$ is ε-optimal (w.r.t. λ-discounted penalties) for any $k' \geq k$.*

Proof. This is proved together with the following lemma:

Lemma 11. *Let $\mathsf{G} = (V_\Box, V_\Diamond, E, \mathsf{weight})$ be a finite weighted game with a reachability objective, $\lambda \in (0,1)$, and $c \in \mathbb{Q}$. Then (G, λ, c) is a positive instance of the optimal discounted-penalty problem iff for any $\varepsilon > 0$, $(\mathsf{G}, \lambda, c+\varepsilon)$ is a positive instance of the optimal reachability problem with discounted penalty*

Proof. From the remark following the proof of Theorem 8, there is a memoryless optimal multi-strategy σ_1 all of whose maximal outcomes have λ-discounted penalty less than or equal to c, and never visit losing states. Let σ_2 be a memoryless winning multi-strategy for the reachability objective, and let c_2 be the maximal penalty accumulated along an outcome of σ_2. Let $\varepsilon > 0$, and $k \in \mathbb{N}$ such that $\lambda^k \cdot c_2 \leq \varepsilon$. Then for any $k' > k$, the multi-strategy $\sigma_1^{k'} \cdot \sigma_2^*$ is winning, and the λ-discounted penalty of any outcome is at most $c + \lambda^{k'} \cdot c_2 \leq c + \varepsilon$.

Conversely, let $\varepsilon > 0$, and σ be a winning multi-strategy achieving discounted penalty no more than $c + \varepsilon$. Then in particular, σ achieves discounted penalty $\leq c + \varepsilon$ along all of its outcomes, so that $(G, \lambda, c+\varepsilon)$ is a positive instance of the optimal discounted-penalty problem (for any $\varepsilon > 0$). From Theorem 8, this problem admits a (truly) optimal memoryless multi-strategy, so that there must exist a multi-strategy achieving discounted penalty less than or equal to c along all of its outcomes. ⌟⌟

Theorem 12. *The optimal reachability problem with discounted penalty is in* NP ∩ coNP*, and is* PTIME*-hard.*

Remark 2. It can be observed that the results of this section extend to discounted-cost games with reachability objectives (without the exponential gap due to the first transformation of weighted games with penalties). In particular, those games admit almost-optimal strategies made of two memoryless strategies, and the corresponding decision problem is equivalent to classical discounted-payoff games.

4.4 Extension to the mean penalty of multi-strategies

We also define the *mean penalty* of a multi-strategy σ from state v, denoted $\mathsf{mean\_penalty}(\sigma, v)$, as $\sup\{\mathsf{mean\_penalty}_\sigma(\varrho) \mid \varrho \in \mathsf{Out}_\mathsf{G}(\sigma, v),\ \varrho \text{ maximal}\}$, where

$$\mathsf{mean\_penalty}_\sigma(\varrho) = \begin{cases} \frac{\mathsf{penalty}_\sigma(\varrho)}{|\varrho|} & \text{if } |\varrho| < \infty \\ \limsup_{n \to +\infty} \mathsf{mean\_penalty}_\sigma(\varrho_{|\leq n}) & \text{otherwise} \end{cases}$$

where $\varrho_{|\leq n}$ is the prefix of length n of ϱ. The notion of ε-optimality, for $\varepsilon \geq 0$, is defined as in the previous cases. Using the same lines orf arguments as in the previous section, we get:

Theorem 13. *Let $\mathsf{G} = (V_\Box, V_\Diamond, E, \mathsf{weight})$ be a finite weighted game with reachability objectives. We assume that all the states in $V_\Box \cup V_\Diamond$ are winning. Then there exist two memoryless multi-strategies σ_1 and σ_2 such that, for any $\varepsilon > 0$, there is an integer k such that the multi-strategy $\sigma_1^{k'} \cdot \sigma_2^*$ is ε-optimal (w.r.t. mean penalties) for any $k' \geq k$.*

Theorem 14. *The optimal reachability problem with mean-penalty is in* $\mathsf{NP} \cap \mathsf{coNP}$ *and is* PTIME*-hard.*

Remark 3. Again, this result extends to mean-cost games with reachability objectives, which thus admit almost-optimal strategies made of two memoryless strategies. Surprisingly, the same phenomenon has been shown to occur in mean-payoff parity games [CHJ05], but the corresponding strategy can be made fully optimal thanks to the infiniteness of the outcomes.

5 Conclusion and future work

We have proposed an original quantitative approach to the permissivity of (multi-)strategies in two-player games with reachability objectives, through a natural notion of penalty given to the player for blocking edges. We have proven that most permissive strategies exist and can be chosen memoryless in the case where penalties are added up along the outcomes, and proposed a PTIME algorithm for computing such an optimal strategy. When considering discounted sum or mean penalty, we have proved that we must settle for almost-optimal strategies, which are built from two memoryless strategies. The resulting algorithm is in $\mathsf{NP} \cap \mathsf{coNP}$. This is rather surprising as the natural way of encoding multi-strategies in classical weighted games entails an exponential blowup.

Besides the naturalness of multi-strategies, our initial motivation underlying this work (and the aim of our future works) is in the domain of timed

games [AMPS98,BCD+07]: in that setting, strategies are often defined as functions from executions to pairs (t, a) where t is a real number and a an action. This way of defining strategies goes against the paradigm of *implementability* [DDR04], as it requires infinite precision. We plan to extend the work reported here to the timed setting, where penalties would depend on the precision needed to apply the strategy.

Also, as stated in [CHJ05], we believe that games with mixed objectives are interesting on their own, which gives another direction of research for future work. This catches up with related works on quantitative extensions of ATL.

References

[AMPS98] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *Proc. 5th IFAC Cconf. System Structure and Control (SSC'98)*, p. 469–474. Elsevier, July 1998.

[BCD+07] G. Behrmann, A. Cougnard, A. David, E. Fleury, K. G. Larsen, and D. Lime. UPPAAL-Tiga: Time for playing games! In *Proc. 19th Intl Conf. Computer Aided Verification (CAV'07)*, LNCS 4590, p. 121–125. Springer, July 2007.

[BJW02] J. Bernet, D. Janin, and I. Walukiewicz. Permissive strategies: from parity games to safety games. *RAIRO – Theor. Inf. Appl.*, 36(3):261–275, 2002.

[CHJ05] K. Chatterjee, T. A. Henzinger, and M. Jurdziński. Mean-payoff parity games. In *Proc. 20th Ann. Symp. Logic in Computer Science (LICS'05)*, p. 178–187. IEEE Comp. Soc. Press, July 2005.

[DDR04] M. De Wulf, L. Doyen, and J.-F. Raskin. Almost ASAP semantics: From timed models to timed implementations. In *Proc. 7th International Workshop on Hybrid Systems: Computation and Control (HSCC'04)*, Lecture Notes in Computer Science 2993, p. 296–310. Springer, 2004.

[HP06] T. A. Henzinger and V. S. Prabhu. Timed alternating-time temporal logic. In *Proc. 4th Intl Conf. Formal Modelling and Analysis of Timed Systems, (FORMATS'06)*, LNCS 4202, p. 1–17. Springer, September 2006.

[Jur98] M. Jurdziński. Deciding the winner in parity games is in $\mathsf{UP} \cap \mathsf{coUP}$. *Information Processing Letters*, 68(3):119–124, 1998.

[LMO06] F. Laroussinie, N. Markey, and G. Oreiby. Model checking timed ATL for durational concurrent game structures. In *Proc. 4th Intl Conf. Formal Modelling and Analysis of Timed Systems (FORMATS'06)*, LNCS 4202, p. 245–259. Springer, September 2006.

[Lut08] M. Luttenberger. Strategy iteration using non-deterministic strategies for solving parity games. Research Report cs.GT/0806.2923, arXiv, June 2008.

[PR05] S. Pinchinat and S. Riedweg. You can always compute maximally permissive controllers under partial observation when they exist. In *Proc. 24th American Control Conference (ACC'05)*, p. 2287–2292, 2005.

[Tho02] W. Thomas. Infinite games and verification. In *Proc. 14th International Conference on Computer Aided Verification (CAV'02)*, LNCS 2404, p. 58–64. Springer, 2002.

[ZP96] U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *TCS*, 158(1-2):343–359, May 1996.

A Proof of Lemmas 2 and 4

Lemma 2. *For every weighted game* $\mathsf{G} = (V_\Box, V_\Diamond, E, \mathsf{weight})$*, we can construct an exponential-size weighted game* $\mathsf{G}' = (V'_\Box, V'_\Diamond, E', \mathsf{weight}')$ *such that* $V_\Box \subseteq V'_\Box$*,* $V_\Diamond \subseteq V'_\Diamond$ *and, for any state* $v \in V_\Box \cup V_\Diamond$ *and any bound* c*, Player* $\Diamond$ *has a winning multi-strategy with penalty* c *from* v *in* G *iff she has a winning strategy with cost* c *from* v *in* G'*.*

Proof. The weighted game $\mathsf{G}' = (V'_\Box, V'_\Diamond, E', \mathsf{weight}')$ is defined as follows (see Fig. 2 for an example):

- the set of states of Player $\Diamond$ is unchanged, while the set of states of Player $\Box$ is augmented with $\{(v, F) \mid v \in V_\Diamond,\ F \subseteq vE\}$;
- E' is the (disjoint) union of three kinds of transitions:
  (1) transitions in $E \cap (V_\Box \times V)$,
  (2) transitions of the form $((v, F), v')$ for each $(v, v') \in F$,
  (3) transitions of the form $(v, (v, F))$ for each $v \in V_\Diamond$ and $F \subseteq vE$;
- the weight function weight' is defined as follows:
  - each edge e of type (1) has weight $\mathsf{weight}(e)$;
  - transitions of type (2) have weight 0;
  - each transition $(v, (v, F))$ of type (3) has weight $\sum_{e \in vE \setminus F} \mathsf{weight}(e)$.

We now prove the correctness of this construction. Let σ be a winning multi-strategy for Player $\Diamond$ in G. Given a run ϱ' in G' such that $\mathsf{last}(\varrho') \in V'_\Diamond$, we consider the run ϱ obtained from ϱ' by removing every second state. This is a finite run of G, ending in $V_\Diamond$. We let $\sigma'(\varrho') = (\mathsf{last}(\varrho'), \sigma(\varrho))$. By construction, the cost of this transition corresponds to the penalty of playing σ from ϱ. Then Player $\Box$ in G' can choose one of the edges in $\sigma(\varrho)$, and switch to the corresponding state. In the end, any outcome ϱ' of σ' in G' with cost d corresponds to an outcome of σ in G with penalty d, and conversely. This entails that σ' is winning and has cost the penalty of σ.

For the other direction, we define σ inductively (on the length of the history) from σ', enforcing that the outcomes of σ in G of length n correspond (one-to-one) to the outcomes of σ' in G' of length $2n - 1$, and that this correspondence preserves the quantity (cost vs. penalty) of the outcomes:

- for $v \in V_\Diamond$, letting $\sigma'(v) = (v, F)$, we define $\sigma(v) = F$ (which is a set of edges leaving v). By construction, the penalty assigned to σ in v is the weight of the transition (v, F). The correspondence between outcomes is thus satisfied;
- given an outcome ϱ of σ of length n, we define $\sigma(\varrho)$ as the set of edges indicated by $\sigma'(\varrho')$, where ϱ' is the outcome of σ' corresponding to ϱ. Again, the correspondence between outcomes holds with this definition.

The definition of σ on histories that are not outcomes of σ is irrelevant and can be set arbitrarily. Thanks to the correspondence between outcomes, σ is winning (assuming that σ' is) and its penalty is the cost of σ'. ⌟

Lemma 4. *For every weighted game* $\mathsf{G}' = (V'_\Box, V'_\Diamond, E', \mathsf{weight}')$*, we can construct a polynomial-size weighted game* $\mathsf{G} = (V_\Box, V_\Diamond, E, \mathsf{weight})$ *such that* $V'_\Box \subseteq V_\Box$*,* $V'_\Diamond \subseteq V_\Diamond$*, and for any state* $v \in V'_\Box \cup V'_\Diamond$ *and any value* c*, Player* $\Diamond$ *has a winning strategy with cost* c *from* v *in* G' *iff she has a winning multi-strategy with penalty* c *from* v *in* G*.*

Proof. The penalty game $\mathsf{G} = (V_\Box, V_\Diamond, E, \mathsf{weight})$ is defined as follows:

- the set of states of Player $\Box$ is unchanged, while $V_\Diamond = V'_\Diamond \cup V_{E'}$ where $V_{E'} = \{v_e \mid e \in E'\}$ is a set disjoint from $V'_\Diamond$. In other words we add one state for each edge of G';
- E is the (disjoint) union of three kinds of edges:
  (1) edges of the form (v, v_e) for each $v \in V$ and each $e \in vE'$,
  (2) edges of the form (v_e, v) for each $v \in V$ and each $e \in E'v$,
  (3) edges of the form $(v_e, \frownie)$ for each $e \in E'$;
- the weight function weight is defined as follows:
  - each edge (v, v_e) or (v_e, v) of type (1) or (2) has weight 0,
  - each edge $(v_e, \frownie)$ of type (3) has weight $\mathsf{weight}'(e)$.

Now, a winning strategy σ' for Player $\Diamond$ in the original game G' can be transformed into a winning multi-strategy σ (which is actually a deterministic strategy) in G such that $\mathsf{cost}(\sigma', v) = \mathsf{penalty}(\sigma, v)$ for all $v \in V'_\Box \cup V'_\Diamond$. The transformation is defined inductively (on the length of the history) as follows:

- if $v \in V'_\Diamond$, the state corresponds to a state in G' and we set $\sigma(v) = (v, v_e)$ where $e = \sigma'(v)$;
- if $v = v_e \in V_{E'}$ and $e = (v', v")$ then $\sigma(v) = (v_e, v")$;
- if $\varrho \in \mathsf{Out}_\mathsf{G}(\sigma, v)$ with $\mathsf{last}(\varrho) = v_e \in V_{E'}$ and $e = (v', v")$ then $\sigma(\varrho) = (v_e, v")$;
- if $\varrho \in \mathsf{Out}_\mathsf{G}(\sigma, v)$ with $\mathsf{last}(\varrho) = v' \in V'_\Diamond$, then by removing every second state from ϱ we have a run $\varrho' \in \mathsf{Out}_{\mathsf{G}'}(\sigma', v)$, and we can thus define $\sigma(\varrho) = (v', v_e)$ with $e = \sigma'(\varrho')$. Furthermore, it is easy to see that the cost of ϱ' is precisely the penalty of ϱ in G;
- if $\forall v,\ \varrho \notin \mathsf{Out}_\mathsf{G}(\sigma, v)$ then $\sigma(\varrho)$ doesn't matter (we can take $\sigma(\varrho) = \varnothing$).

The strategy σ' is winning in G' iff σ is winning in G, and the cost/penalty of both strategies coincide.

Conversely, a winning multi-strategy σ for Player $\Diamond$ in G can be transformed into a winning strategy σ' in G' such that $\mathsf{cost}(\sigma', v) = \mathsf{penalty}(\sigma, v)$ for all $v \in V'_\Box \cup V'_\Diamond$.

Since we want to build a strategy from a multi-strategy we first need to fix the non-deterministic choices allowed in σ. We thus define a deterministic[9] multi-strategy σ_1 in G such that for every $v \in V_\Box \cup V_\Diamond$, $\mathsf{penalty}(\sigma_1, v) = \mathsf{penalty}(\sigma, v)$. By definition of penalties, and given that σ is winning, for every $v \in V'_\Box \cup V'_\Diamond$ the penalty $\mathsf{penalty}(\sigma, v)$ is the maximum over all (finite) paths in $\mathsf{Out}^{\max}_\mathsf{G}(\sigma, v)$ of the penalty along a path. Let ϱ_v be a path achieving such a maximum for v. We now give the multi-strategy σ_1:

[9] In the sense that each finite run is associated at most one edge.

– if $\varrho \in \mathsf{Out}_{\mathsf{G}}(\sigma, v)$ with $\mathsf{last}(\varrho) = v' \in V_{E'}$ then $\sigma(\varrho)$ contains just one edge (the one not leading to the bad state) and is 'deterministic'. In that case we set $\sigma_1(\varrho) = \sigma(\varrho)$;
– if $\varrho \in \mathsf{Out}_{\mathsf{G}}(\sigma, v)$ with $\mathsf{last}(\varrho) = v' \in V'_{\Diamond}$ and ϱ is a prefix of ϱ_v (*i.e.* $\varrho_v = \varrho v_e \varrho'$), we choose for $\sigma_1(\varrho)$ the following edge along ϱ_v, *i.e.* $\sigma_1(\varrho) = \{(v', v_e)\}$;
– if $\varrho \in \mathsf{Out}_{\mathsf{G}}(\sigma, v)$ with $\mathsf{last}(\varrho) = v' \in V'_{\Diamond}$ and ϱ is not a prefix of ϱ_v, then we choose arbitrarily an edge e in $\sigma(\varrho)$ and set $\sigma_1(\varrho) = e$;
– if for every $v \in V$, $\varrho \notin \mathsf{Out}_{\mathsf{G}}(\sigma, v)$, since $\mathsf{Out}_{\mathsf{G}}(\sigma_1, v) \subseteq \mathsf{Out}_{\mathsf{G}}(\sigma, v)$ the definition of $\sigma_1(\varrho)$ doesn't matter and we set $\sigma_1(\varrho) = \varnothing$.

We clearly have a deterministic multi-strategy, but need to prove that the penalties have been preserved. First $\mathsf{Out}^{\max}_{\mathsf{G}}(\sigma_1, v) \subseteq \mathsf{Out}^{\max}_{\mathsf{G}}(\sigma, v)$, and since we have only removed edges of weight 0, the penalties of the remaining runs have not changed. Taking the maximum of penalties over a smaller set we get $\mathsf{penalty}(\sigma_1, v) \leq \mathsf{penalty}(\sigma, v)$. On the other hand the path ϱ_v that achieved the maximal penalty is in $\mathsf{Out}^{\max}_{\mathsf{G}}(\sigma_1, v)$. Since its penalty has been preserved, we have $\mathsf{penalty}(\sigma_1, v) \geq \mathsf{penalty}(\sigma, v)$ and thus the equality.

The last step is to define a winning strategy σ' in G' based on σ_1 and preserving the cost/penalty. By construction, for every edge $e = (v, v') \in E'$ there exist two edges (v, v_e) and (v_e, v') in E, thus we can associate to every run $\varrho' = v_0 v_1 ... v_k$ in $\mathsf{Runs}_{\mathsf{G}'}$ a unique run $\iota(\varrho') = v_0 v_{e_0} v_1 v_{e_1} ... v_{e_{k-1}} v_k$ in $\mathsf{Runs}_{\mathsf{G}}$ with $e_i = (v_i, v_{i+1})$ for every $0 \leq i < k$. We now inductively define the strategy σ:

– if $v \in V'_{\Diamond}$, we set $\sigma'(v) = e$ where $\sigma_1(v) = \{(v, v_e)\}$;
– if $\varrho' \in \mathsf{Out}_{\mathsf{G}'}(\sigma', v)$ with $\mathsf{last}(\varrho') = v' \in V'_{\Diamond}$ we set $\sigma'(\varrho') = e$, where $\sigma_1(\iota(\varrho)) = \{(v', v_e)\}$.

By construction of G, from a state v in $V'_{\Diamond} \subseteq V_{\Diamond}$, the choice of an outgoing edge (v, v_e) corresponds to the choice of the edge e in G', and the cost of this edge in G' is precisely the weight of the losing transition from v_e in G. Furthermore in G, all choices made from $V_{\Diamond} \setminus V'_{\Diamond}$ induce no extra penalties (because the weight of all edges leaving those states is always 0). This implies that the cost of strategy σ' coincides with the penalty of multi-strategy σ_1. ⌟

B Proof of Lemma 5

Lemma 5. *Assume that we are given an optimal winning multi-strategy σ. We can effectively construct in polynomial time a memoryless multi-strategy σ', which is winning and optimal.*

Proof. Assume that W is the set of winning states. Let $\mathcal{T}$ be the forest representing the multi-strategy σ (its set of roots is W). Since σ is winning from every state in W, all branches of the forest are finite. For every node t of $\mathcal{T}$, we define $\gamma_{\mathcal{T}}(t)$ as the *residual penalty* of σ after prefix t. Formally, $\gamma_{\mathcal{T}}(t) = \mathsf{penalty}(\sigma, t)$. Obviously, for all $v \in V$, we have $\mathsf{penalty}(\sigma, v) = \gamma_{\mathcal{T}}(v)$.

We will consider a measure $\mu_{\mathcal{T}}$ on the set of nodes of the forest $\mathcal{T}$ as follows: if t is a node of $\mathcal{T}$, we let $\mu_{\mathcal{T}}(t) = (\gamma_{\mathcal{T}}(t), \mathit{height}_{\mathcal{T}}(t))$.

We say that *no memory is required* for state v in $\mathcal{T}$ if, for every two nodes t and t' that are occurrences of v, the subtree of $\mathcal{T}$ rooted at t and the subtree of $\mathcal{T}$ rooted at t' are identical. Note that in that case, $\mu_{\mathcal{T}}(t) = \mu_{\mathcal{T}}(t')$.

For every $0 \leq i \leq |V|$, we (inductively) build in polynomial time a forest $\mathcal{T}^i$ and a set $M_i \subseteq V$ containing i elements, such that:

(a) $\mathcal{T}^i$ represents an optimal winning multi-strategy from all the states of V;
(b) for every $v \in M_i$, no memory is required for v in $\mathcal{T}^i$, and for every node t' which is a descendant of some node that is an occurrence of v, letting $v' = \mathsf{last}(t')$, it holds $v' \in M_i$.

Intuitively, each $\mathcal{T}^i$ will be the forest of a winning optimal multi-strategy σ_i, and each M_i will be a set of states from which σ_i is memoryless (*i.e.*, σ_i is memoryless from the states in M_i, and from the states that occur in the outcomes from these states).

For $i = 0$, it suffices to define $\mathcal{T}^0$ as the forest $\mathcal{T}$, and $M_0 = \varnothing$. We assume we have already constructed the forest $\mathcal{T}^i = (T^i, V)$, and a corresponding set M_i. We pick a state $v_i \in V \setminus M_i$ such that there is an occurrence $t_i \in T^i$ of v_i with

$$\mu_{\mathcal{T}^i}(t_i) = \min\{\mu_{\mathcal{T}^i}(t) \mid t \text{ is an occurrence of some } v \in V \setminus M_i \text{ in } \mathcal{T}^i\}.$$

Notice that each node t in the subtree rooted at t_i (in the forest $\mathcal{T}^i$) is an occurrence of a state of M_i. Indeed, if this were not the case, there would be a node t'_i, which is a descendant of t_i, and which would then be such that $\gamma_{\mathcal{T}^i}(t'_i) \leq \gamma_{\mathcal{T}^i}(t_i)$ and $\mathit{height}_{\mathcal{T}^i}(t'_i) < \mathit{height}_{\mathcal{T}^i}(t_i)$, contradicting the choice of t_i. In particular, there is no occurrence of v_i in the subtree of $\mathcal{T}^i$ rooted at t_i.

The forest $\mathcal{T}^{i+1}$ is defined from the forest $\mathcal{T}^i$ by replacing every subtree rooted at an occurrence of v_i with the subtree of $\mathcal{T}^i$ rooted at t_i. We define $M_{i+1} = M_i \cup \{v_i\}$ (which then has $i+1$ elements). It remains to check that all required conditions are satisfied. Clearly enough, since (by induction hypothesis) $\mathcal{T}^i$ represents a multi-strategy from all the states of V, this is also the case of $\mathcal{T}^{i+1}$. Pick an occurrence $t \in T^i$ of v_i such that v_i does not occur in any prefix of t. Then $t \in \mathcal{T}^{i+1}$. By definition of t_i, we obviously have that $\gamma_{\mathcal{T}^i}(t) \geq \gamma_{\mathcal{T}^i}(t_i)$. On the other hand, we also have that $\gamma_{\mathcal{T}^{i+1}}(t) = \gamma_{\mathcal{T}^i}(t_i) \leq \gamma_{\mathcal{T}^i}(t)$. These constraints propagate to all prefixes of t in $\mathcal{T}^{i+1}$, hence the multi-strategy represented by $\mathcal{T}^{i+1}$ is optimal (from every state of V). This concludes the proof of condition (a). By construction, no memory is required for state v_i in $\mathcal{T}^{i+1}$. Assume that for some $v \in M_i$, there are two different subtrees of $\mathcal{T}^{i+1}$ rooted at an occurrence of v. It means that at least one of the subtrees has been changed by the substitution, hence that an occurrence of v_i was in the subtree rooted at t. This contradicts condition (b) satisfied by $\mathcal{T}^i$ since $v_i \notin M_i$. Hence, the condition (b) is satisfied by $\mathcal{T}^{i+1}$.

The forest $\mathcal{T}^{|V|}$ represents a multi-strategy σ' which is memoryless, optimal and winning from every state of the game. ⌟

C Proof of Lemmas 6 and 7

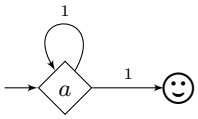

Fig. 6. No optimal discounted multi-strategy

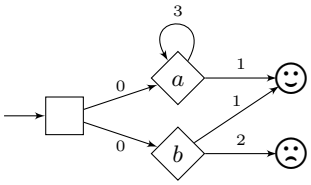

Fig. 7. No memoryless optimal discounted multi-strategy

Lemma 6. *There exists weighted games for which there is no optimal winning multi-strategy under discounted penalties.*

Proof. Figure 6 shows such a game. There is only one non terminal state and a multi-strategy for this game consists in choosing at each step whether to block the loop, the transition to the terminal state or none, with respective cost one, one and zero. In this game there is no multi-strategy with cost zero, since blocking nothing allows an infinite run staying in state a forever. On the other hand a multi-strategy that blocks nothing the n first steps and then blocks the loop has cost λ^n. Hence, for every $\varepsilon > 0$, it is easy to design a ε-optimal discounted multi-strategy. ⌟

Lemma 7. *There are weighted games under discounted penalties for which there exist a memoryful optimal winning multi-strategy but no memoryless one.*

Proof. Figure 7 shows such a game. The only memoryless winning multi-strategy for this game blocks the losing transition and the loop and has penalty 3λ. Any winning multi-strategy has a penalty of at least 2λ (the penalty of blocking the losing transition), and if we take a multi-strategy that blocks the loop only after n steps with $3\lambda^n \leq 2\lambda$ (such a finite n exists since $\lambda < 1$) we get an optimal winning multi-strategy (which is obviously not memoryless). ⌟

D Proof of Theorem 8

Theorem 8. *The optimal discounted-penalty problem is in* $\mathsf{NP} \cap \mathsf{coNP}$*, and is* PTIME*-hard.*

Proof. We let $\mathsf{G} = (V_\Box, V_\Diamond, E, \mathsf{weight})$ be a finite weighted game, and let $c \in \mathbb{Q}$. Applying the transformation of Lemma 2 to the discounted case, we get an exponential-size weighted game $\mathsf{G}' = (V'_\Box, V'_\Diamond, E', \mathsf{weight}')$ with $V_\Box \subseteq V'_\Box$ and $V_\Diamond = V'_\Diamond$ such that for every $v \in V_\Box \cup V_\Diamond$, Player $\Diamond$ has a winning multi-strategy from v in G with discounted penalty no more than c (for discount λ) iff Player $\Diamond$

has a winning strategy from v in G' with discounted cost no more than c (for discount $\sqrt{\lambda}$). The game G' is memoryless determined [ZP96], which means that for every $c \in \mathbb{Q}$, for every state $v \in V'_\Box \cup V'_\Diamond$, either Player $\Diamond$ has a memoryless strategy $\sigma_\Diamond$ with $\mathsf{cost}^{\sqrt{\lambda}}(\sigma_\Diamond, v) \leq c$, or Player $\Box$ has a memoryless strategy $\sigma_\Box$ with $\mathsf{cost}^{\sqrt{\lambda}}(\sigma_\Box, v) > c$.

The NP algorithm is then as follows: guess a memoryless strategy $\sigma_\Diamond$ for Player $\Diamond$, *i.e.* for every $v \in V_\Diamond$ guess a subset $F \subseteq vE$ and set $\sigma_\Diamond(v) = (v, F)$. The game G' where we have removed parts not allowed by $\sigma_\Diamond$ has polynomial size and can be constructed from G without first constructing G'. In the resulting graph, from v, it is easy to compute in polynomial time the maximal discounted cost, which corresponds to $\mathsf{cost}^{\sqrt{\lambda}}(\sigma_\Diamond, v)$. Thus we can decide in polynomial time whether $\mathsf{cost}^{\sqrt{\lambda}}(\sigma_\Diamond, v) \leq c$.

Conversely, the coNP algorithm consists in guessing a strategy for Player $\Box$ with cost larger than c. However, Player $\Box$ controls exponentially many states in G', and her strategy cannot be guessed in polynomial time. We have to first define a succinct encoding of memoryless optimal strategies of Player $\Box$.

Applying [Jur98], we know that an optimal memoryless strategy $\sigma_\Box$ can be computed, that is is characterized by values $val^{\sqrt{\lambda}}(v) = \mathsf{cost}^{\sqrt{\lambda}}(\sigma_\Box, v)$ (for $v \in V'$) satisfying the following optimality equations:

$$\begin{cases} val^{\sqrt{\lambda}}(v) = \min\limits_{(v,F)\in V'_\Box} \left(\mathsf{weight}'(v,(v,F)) + \sqrt{\lambda}\cdot val^{\sqrt{\lambda}}(v,F)\right) & \text{if } v \in V_\Diamond \\ val^{\sqrt{\lambda}}(v) = \mathsf{weight}'(v,(v,vE)) + \sqrt{\lambda}\cdot val^{\sqrt{\lambda}}(v,vE) & \text{if } v \in V_\Box \\ val^{\sqrt{\lambda}}(\text{☺}) = 0 & \\ val^{\sqrt{\lambda}}(v,F) = \max\limits_{e=(v,v')\in F} \left(\mathsf{weight}'((v,F),v') + \sqrt{\lambda}\cdot val^{\sqrt{\lambda}}(v')\right) & \text{if } (v,F) \in V'_\Box \setminus V_\Box \end{cases}$$

By construction of G', we can rewrite these equations as follows:

$$\begin{cases} val^{\sqrt{\lambda}}(v) = \min\limits_{F\subseteq vE} \left(\sum\limits_{e\in vE\setminus F} \mathsf{weight}(e) + \sqrt{\lambda}\cdot val^{\sqrt{\lambda}}(v,F)\right) & \text{if } v \in V_\Diamond \\ val^{\sqrt{\lambda}}(v) = \sqrt{\lambda}\cdot val^{\sqrt{\lambda}}(v,vE) & \text{if } v \in V_\Box \\ val^{\sqrt{\lambda}}(\text{☺}) = 0 & \\ val^{\sqrt{\lambda}}(v,F) = \max\limits_{e=(v,v')\in F} \sqrt{\lambda}\cdot val^{\sqrt{\lambda}}(v') & \text{if } (v,F) \in V'_\Box \setminus V_\Box \end{cases}$$

Furthermore the strategy $\sigma_\Box$ can be defined by:

$$\begin{cases} \sigma_\Box(v) = (v, vE) & \text{if } v \in V_\Box \\ \sigma_\Box((v,F)) = ((v,F),v') & \text{if } (v,F) \in V'_\Box \setminus V_\Box \text{ and} \\ & val^{\sqrt{\lambda}}(v') = \max\limits_{e=(v,v'')\in F} val^{\sqrt{\lambda}}(v'') \end{cases}$$

In particular, we can define a total order $\prec$ (which we call a *preference order*) on states such that $v \prec v'$ if $val^{\sqrt{\lambda}}(v) \geq val^{\sqrt{\lambda}}(v')$. For any $F \subseteq vE$, we define $v^F_\prec$

as the smallest element w.r.t. order $\prec$ such that $(v, v^F_\prec) \in F$. We can then fix the strategy $\sigma^\prec_\Box$ as follows:

$$\begin{cases} \sigma^\prec_\Box(v) = (v, vE) & \text{if } v \in V_\Box \\ \sigma^\prec_\Box((v,F)) = ((v,F), v^F_\prec) & \text{if } v \in V'_\Box \smallsetminus V_\Box \end{cases}$$

The strategy $\sigma^\prec_\Box$ is memoryless optimal for Player $\Box$ and follows the preference order $\prec$, which means that if $v^F_\prec = v^{F'}_\prec$, then $\mathsf{target}(\sigma^\prec((v,F))) = \mathsf{target}(\sigma^\prec((v,F')))$. And this proves that we can restrict strategies for Player $\Box$ in G' to memoryless strategies with a preference order on states. The main advantage of these strategies is that they have polynomial size.

The coNP algorithm proceeds as follows: guess a preference order $\prec$ on states of G. We want to compute $\mathsf{cost}^\lambda(\sigma^\prec_\Box, v)$ in G' in polynomial time (thus without explicitely constructing G'). To that aim, we construct a weighted game G'' whose discounted cost coincides with $\mathsf{cost}^\lambda(\sigma^\prec_\Box, v)$. The game $\mathsf{G}'' = (V''_\Box, V''_\Diamond, E'', \mathsf{weight}'')$ is defined as:

- $V''_\Box = V_\Box \cup \{(v, v^F_\prec) \mid F \subseteq vE \text{ and } (v,F) \in V'_\Box\}$ and $V''_\Diamond = V_\Diamond$; notice that, from the definition of $v^F_\prec$, there are only polynomially many states.
- the set E'' is made of:
  - $(v, (v, v^F_\prec)) \in E''$ if $(v, (v,F)) \in E'$;
  - $((v, v^F_\prec), v^F_\prec) \in E''$.
- the weight function is defined as:
  - $\mathsf{weight}''(v, (v, v^F_\prec)) = \min\limits_{F' \subseteq vE \text{ s.t. } v^{F'}_\prec = v^F_\prec} \mathsf{weight}'(v, (v,F))$
  - $\mathsf{weight}''((v, v^F_\prec), v^F_\prec) = 0$

Then it is not difficult to realise that

$$\mathsf{weight}''(v, (v, v^F_\prec)) = \sum_{v' \prec v^F_\prec \text{ s.t. } (v,v') \in vE} \mathsf{weight}(v, v')$$

Indeed, we restrict the order $\prec$ to $\{v' \mid (v,v') \in vE\} = \{v_1, \cdots, v_p\}$, assuming $v_1 \prec v_2 \cdots \prec v_p$. Now, we can write:

$$\begin{aligned} \mathsf{weight}''(v, (v, v_i)) &= \min_{F \subseteq vE \text{ s.t. } v^F_\prec = v_i} \mathsf{weight}'(v, (v,F)) \\ &= \min_{F \subseteq vE \text{ s.t. } v_1, \cdots, v_{i-1} \notin F \text{ and } v_i \in F} \mathsf{weight}'(v, (v,F)) \\ &= \min_{F \subseteq vE \text{ s.t. } v_1, \cdots, v_{i-1} \notin F \text{ and } v_i \in F} \left(\sum_{e \in vE \smallsetminus F} \mathsf{weight}(e) \right) \\ &= \sum_{j=1}^{i-1} \mathsf{weight}(v, v_j) \qquad (\text{taking } F = vE \smallsetminus \{(v, v_j) \mid 1 \leq j < i\}) \end{aligned}$$

Thus the game G'' can be computed in polynomial time. Furthermore all states in $V''_\Box$ have at most one successor, this is thus not a game, but a graph in

which we can easily compute in polynomial time the discounted cost. It remains to prove that the discounted cost in G'' from v (denoted $\mathsf{cost}^{\sqrt{\lambda}}_{\mathsf{G}''}(v)$) coincides with $\mathsf{cost}^{\sqrt{\lambda}}_{\mathsf{G}'}(\sigma^{\prec}_{\Box}, v)$. We first notice that any run in G'' can be seen as a run in G' under strategy $\sigma^{\prec}_{\Box}$, and the (discounted) costs coincide. Thus, $\mathsf{cost}^{\sqrt{\lambda}}_{\mathsf{G}''}(v) \geq \mathsf{cost}^{\sqrt{\lambda}}_{\mathsf{G}'}(\sigma^{\prec}_{\Box}, v)$.

Let ϱ be a run generated by $\sigma^{\prec}_{\Box}$ from v in G': we have $\varrho = v_0(v_0, F_0)v_1(v_1, F_1)\cdots$ with $(v_i, F_i) = \sigma^{\prec}_{\Box}(v_i)$ if $v_i \in V_{\Box}$ and $v_{i+1} = v_i{}^{\prec}_{F_i}$ in any case. The run $\widetilde{\varrho}$ defined as $v_0(v_0, \widetilde{F_0})v_1(v_1, \widetilde{F_1})\cdots$ where $\widetilde{F}_i = F_i$ if $v_i \in V_{\Box}$, and $\widetilde{F}_i$ is the largest subset of v_iE so that $v_i{}^{\widetilde{F_i}}_{\prec} = v_i{}^{F_i}_{\prec}$. The run $\widetilde{\varrho}$ is also generated by $\sigma^{\prec}_{\Box}$, and its discounted is no more than that of ϱ. Furthermore this run can be played in G'', hence we get the converse inequality: $\mathsf{cost}^{\sqrt{\lambda}}_{\mathsf{G}''}(v) \leq \mathsf{cost}^{\sqrt{\lambda}}_{\mathsf{G}'}(\sigma^{\prec}_{\Box}, v)$, hence equality holds.

Thus, $\mathsf{cost}^{\sqrt{\lambda}}_{\mathsf{G}'}(\sigma^{\prec}_{\Box}, v)$ can be computed in polynomial time, and we can decide whether $\mathsf{cost}^{\sqrt{\lambda}}_{\mathsf{G}'}(\sigma^{\prec}_{\Box}, v) > c$ in polynomial time. ⌟