Reactive Synthesis Without Regret

Paul Hunter*, **Guillermo A. Pérez**†, **and Jean-François Raskin***

Département d'Informatique, Université Libre de Bruxelles (U.L.B.), Belgium
{phunter,gperezme,jraskin}@ulb.ac.be

Abstract

Two-player zero-sum games of infinite duration and their quantitative versions are used in verification to model the interaction between a controller (Eve) and its environment (Adam). The question usually addressed is that of the existence (and computability) of a strategy for Eve that can maximize her payoff against any strategy of Adam. In this work, we are interested in strategies of Eve that minimize her regret, i.e. strategies that minimize the difference between her actual payoff and the payoff she could have achieved if she had known the strategy of Adam in advance. We give algorithms to compute the strategies of Eve that ensure minimal regret against an adversary whose choice of strategy is (i) unrestricted, (ii) limited to positional strategies, or (iii) limited to word strategies, and show that the two last cases have natural modelling applications. We also show that our notion of regret minimization in which Adam is limited to word strategies generalizes the notion of good for games introduced by Henzinger and Piterman, and is related to the notion of determinization by pruning due to Aminof, Kupferman and Lampert.

1998 ACM Subject Classification F.1.1 Automata, D.2.4 Formal methods

Keywords and phrases Quantitative games, Regret, Verification, Synthesis, Game theory

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2015.114

1 Introduction

The model of two player games played on graphs is an adequate mathematical tool to solve important problems in computer science, and in particular the reactive system synthesis problem [20]. In that context, the game models the non-terminating interaction between the system to synthesize and its environment. Games with quantitative objectives are useful to formalize important quantitative aspects such as mean-response time or energy consumption. They have attracted large attention recently, see e.g. [7, 3]. Most of the contributions in this context are for zero-sum games: the objective of Eve (that models the system) is to maximize the value of the game while the objective of Adam (that models the environment) is to minimize this value. This is a worst-case assumption: because the cooperation of the environment cannot be assumed, we postulate that it is *antagonistic*.

In this antagonistic approach, the main solution concept is that of a *winning strategy*. Given a threshold value, a winning strategy for Eve ensures a minimal value greater than the threshold against any strategy of Adam. However, sometimes there are no winning strategies. What should the behaviour of the system be in such cases? There are several possible answers to this question. One is to consider *non-zero sum* extensions of those games: the environment (Adam) is not completely antagonistic, rather it has its own specification. In such games, a strategy for Eve must be winning only when the outcome satisfies the objectives of Adam, see e.g. [5]. Another option for Eve is to play a strategy which minimizes

* Authors supported by the ERC inVEST (279499) project.
† Author supported by F.R.S.-FNRS fellowship.

26th International Conference on Concurrency Theory (CONCUR 2015).
Editors: Luca Aceto and David de Frutos Escrig; pp. 114–127

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

her *regret*. The regret is informally defined as the difference between what a player actually wins and what she could have won if she had known the strategy chosen by the other player. Minimization of regret is a central concept in decision theory [2]. This notion is important because it usually leads to solutions that agree with common sense.

Let us illustrate the notion of regret minimization on the example of Fig. 1. In this example, Eve owns the squares and Adam owns the circles (we do not use the letters labelling edges for the moment). The game is played for infinitely many rounds and the value of a play for Eve is the long run average of the values of edges traversed during the play (the so-called *mean-payoff*). In this game, Eve is only able to secure a mean-payoff of $\frac{1}{2}$ when Adam is fully antagonistic. Indeed, if Eve (from v_1) plays to v_2 then Adam can force a mean-payoff value of 0, and if she plays to v_3 then the mean-payoff value is at least $\frac{1}{2}$. Note also that if Adam is not fully antagonistic, then the mean-payoff could be as high as 2. Now, assume that Eve does not try to force the highest value in the worst-case but tries to minimize her regret. If she plays $v_1 \mapsto v_2$ then the regret is equal to 1. This is because Adam can play the following strategy: if Eve plays to v_2 (from v_1) then he plays $v_2 \mapsto v_1$ (giving a mean-payoff of 0), and if Eve plays to v_3 then he plays to v_5 (giving a mean-payoff of 1). If she plays $v_1 \mapsto v_3$ then her regret is $1\frac{1}{2}$ since Adam can play the symmetric strategy. It should thus be clear that the strategy of Eve which always chooses $v_1 \mapsto v_2$ is indeed minimizing her regret.

In this paper, we will study three variants of *regret minimization*, each corresponding to a different set of strategies we allow Adam to choose from. The first variant is when Adam can play any possible strategy (as in the example above), the second variant is when Adam is restricted to playing *memoryless strategies*, and the third variant is when Adam is restricted to playing *word strategies*. To illustrate the last two variants, let us consider again the example of Fig. 1. Assume now that Adam is playing memoryless strategies only. Then in this case, we claim that there is a strategy of Eve that ensures regret 0. The strategy is as follows: first play to v_2, if Adam chooses to go back to v_1, then Eve should henceforth play $v_1 \mapsto v_3$. We claim that this strategy has regret 0. Indeed, when v_2 is visited, either Adam chooses $v_2 \mapsto v_4$, and then Eve secures a mean-payoff of 2 (which is the maximal possible value), or Adam chooses $v_2 \mapsto v_1$ and then we know that $v_1 \mapsto v_2$ is not a good option for Eve as cycling between v_1 and v_2 yields a payoff of only 0. In this case, the mean-payoff is either 1, if Adam plays $v_3 \mapsto v_5$, or a payoff of $\frac{1}{2}$, if he plays $v_3 \mapsto v_1$. In all the cases, the regret is 0. Let us now turn to the restriction to word strategies for Adam. When considering this restriction, we use the letters that label the edges of the graph. A word strategy for Adam is a function $w : \mathbb{N} \to \{a, b\}$. In this setting Adam plays a sequence of letters and this sequence is independent of the current state of the game. When Adam plays word strategies, the strategy that minimizes regret for Eve is to always play $v_1 \mapsto v_2$. Indeed, for any word in which the letter a appears, the mean-payoff is equal to 2, and the regret is 0, and for any word in which the letter a does not appear, the mean-payoff is 0 while it would have been equal to $\frac{1}{2}$ when playing $v_1 \mapsto v_3$. So the regret of this strategy is $\frac{1}{2}$ and it is the minimal regret that Eve can secure. Note that the three different strategies give three different values in our example. This is in contrast with the worst-case analysis of the same problem (memoryless strategies suffice for both players).

We claim that at least the two last variants are useful for modelling purposes. For example, the memoryless restriction is useful when designing a system that needs to perform well in an environment which is only partially known. In practical situations, a controller may discover the environment with which it is interacting at run time. Such a situation can be modelled by an arena in which choices in nodes of the environment model an entire family of environments and each memoryless strategy models a specific environment of the family.

Table 1 Complexity of deciding the regret threshold problem.

Payoff type	Any strategy	Memoryless strategies	Word strategies
Sup, Inf, and LimSup	PTIME-c (Thm. 1)	coNP-h (Lem. 8) and in PSPACE (Lem. 6)	EXPTIME-c (Thm. 10)
LimInf	PTIME-c (Thm. 1)	PSPACE-c (Thm. 5)	EXPTIME-c (Thm. 10)
$\underline{\text{MP}}$, $\overline{\text{MP}}$	MP equivalent (Thm. 1)	PSPACE-c (Thm. 5)	Undecidable (Lem. 13)

In such cases, if we want to design a controller that performs reasonably well against all the possible environments, we can consider a controller that minimizes regret: the strategy of the controller will be as close as possible to an optimal strategy if we had known the environment beforehand. This is, for example, the modelling choice done in the famous Canadian traveller's problem [19]: a driver is attempting to reach a specific location while ensuring the traversed distance is *not too far* from the shortest feasible path. The partial knowledge is due to some roads being closed because of snow. The Canadian traveller, when planning his itinerary, is in fact searching for a strategy to minimize his regret for the shortest path measure against a memoryless adversary who determines the roads that are closed. Similar situations naturally arise when synthesizing controllers for *robot motion planning* [21]. We now illustrate the usefulness of the variant in which Adam is restricted to play word strategies. Assume that we need to design a system embedded into an environment that produces disturbances: if the sequence of disturbances produced by the environment is independent of the behavior of the system, then it is natural to model this sequence not as a function of the state of the system but as a temporal sequence of events, i.e. a *word* on the alphabet of the disturbances. Clearly, if the sequences are not the result of an antagonistic process, then minimizing the regret against all disturbance sequences is an adequate solution concept to obtain a reasonable system and may be preferable to a system obtained from a strategy that is optimal under the antagonistic hypothesis.

Contributions. In this paper, we provide algorithms to solve the *regret threshold problem* (strict and non-strict) in the three variants explained above, i.e. given a game and a threshold, does there exist a strategy for Eve with a regret that is (strictly) less than the threshold against all (resp. all memoryless, resp. all word) strategies for Adam. Almost all of our algorithms are reductions to well-known games, therefore synthesizing the corresponding controller amounts to computing the strategy of Eve in the resulting game. We study this problem for six common quantitative measures: Inf, Sup, LimInf, LimSup, $\underline{\text{MP}}$, $\overline{\text{MP}}$. For all measures, but MP, the strict and non-strict threshold problems are equivalent. We state our results for both cases for consistency. In almost all the cases, we provide matching lower bounds showing the worst-case optimality of our algorithms. Our results are summarized in Table 1.

For the variant in which Adam plays word strategies only, we show that we can recover decidability of mean-payoff objectives when the memory of Eve is fixed in advance: in this case, the problem is NP-complete (Theorems 14 and 15).

Related works. The notion of regret minimization is a central one in game theory, see e.g. [22] and references therein. Also, *iterated* regret minimization has been recently proposed by Halpern et al. as a concept for *non-zero* sum games [13]. There, it is applied to matrix games and not to game graphs. In a previous contribution, we have applied the iterated regret

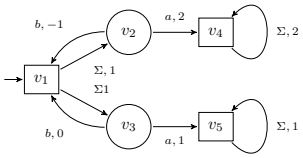

Figure 1 Example weighted arena G_0.

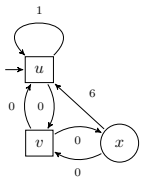

Figure 2 Example weighted arena G_1.

minimization concept to non-zero sum games played on weighted graphs for the shortest path problem [12]. Restrictions on how Adam is allowed to play were not considered there. As we do not consider an explicit objective for Adam, we do not consider iteration of the regret minimization here.

The disturbance-handling embedded system example was first given in [8]. In that work, the authors introduce *remorsefree strategies*, which correspond to strategies which minimize regret in games with ω-regular objectives. They do not establish lower bounds on the complexity of realizability or synthesis of remorsefree strategies and they focus on word strategies of Adam only.

In [14], Henzinger and Piterman introduce the notion of *good for games automata*. A non-deterministic automaton is good for solving games if it fairly simulates the equivalent deterministic automaton. We show that our notion of regret minimization for word strategies extends this notion to the quantitative setting (Proposition 17). Our definitions give rise to a natural notion of approximate determinisation for weighted automata on infinite words.

In [1], Aminof et al. introduce the notion of *approximate determinisation by pruning* for weighted sum automata over finite words. For $\alpha \in (0, 1]$, a weighted sum automaton is *α-determinisable by pruning* if there exists a finite state strategy to resolve non-determinism and that constructs a run whose value is at least α times the value of the maximal run of the given word. So, they consider a notion of approximation which is a *ratio*. We will show that our concept of regret, when Adam plays word strategies only, defines instead a notion of approximation with respect to the *difference* metric for weighted automata (Proposition 16). There are other differences with their work. First, we consider infinite words while they consider finite words. Second, we study a general notion of regret minimization problem in which Eve can use any strategy while they restrict their study to fixed memory strategies only and leave the problem open when the memory is not fixed a priori.

Finally, the main difference between these related works and this paper is that we study the Inf, Sup, LimInf, LimSup, $\underline{\mathsf{MP}}$, $\overline{\mathsf{MP}}$ measures while they consider the total sum measure or qualitative objectives.

2 Preliminaries

A *weighted arena* is a tuple $G = (V, V_\exists, E, w, v_I)$ where (V, E, w) is a finite edge-weighted graph[1] with integer weights, $V_\exists \subseteq V$, and $v_I \in V$ is the initial vertex. In the sequel we depict vertices owned by Eve (i.e. $V_\exists$) with squares and vertices owned by Adam (i.e. $V \setminus V_\exists$) with circles. We denote the maximum absolute value of a weight in a weighted arena by W.

A *play* in a weighted arena is an infinite sequence of vertices $\pi = v_0 v_1 \dots$ where $v_0 = v_I$ and $(v_i, v_{i+1}) \in E$ for all i. We extend the weight function to partial plays by setting $w(\langle v_i \rangle_{i=k}^{l}) = \sum_{i=k}^{l-1} w(v_i, v_{i+1})$.

[1] W.l.o.g. G is assumed to be total: for each $v \in V$, there exists $v' \in V$ such that $(v, v') \in E$.

A *strategy for* Eve (Adam) is a function σ that maps partial plays ending with a vertex v in $V_\exists$ ($V \setminus V_\exists$) to a successor of v. A strategy has memory m if it can be realized as the output of a finite state machine with m states (see e.g. [15] for a formal definition). A *memoryless (or positional) strategy* is a strategy with memory 1, that is, a function that only depends on the last element of the given partial play. A play $\pi = v_0 v_1 \dots$ is *consistent with a strategy* σ for Eve (Adam) if whenever $v_i \in V_\exists$ ($v_i \in V \setminus V_\exists$), $\sigma(\langle v_j \rangle_{j \le i}) = v_{i+1}$. We denote by $\mathfrak{S}_\exists(G)$ ($\mathfrak{S}_\forall(G)$) the set of all strategies for Eve (Adam) and by $\Sigma^m_\exists(G)$ ($\Sigma^m_\forall(G)$) the set of all strategies for Eve (Adam) in G that require memory of size at most m, in particular $\Sigma^1_\exists(G)$ ($\Sigma^1_\forall(G)$) is the set of all memoryless strategies of Eve (Adam) in G. We omit G if the context is clear.

Payoff functions. A play in a weighted arena defines an infinite sequence of weights. We define below several classical *payoff functions* that map such sequences to real numbers.[2] Formally, for a play $\pi = v_0 v_1 \dots$ we define:

- the Inf (Sup) payoff, is the minimum (maximum) weight seen along a play: $\mathsf{Inf}(\pi) = \inf\{w(v_i, v_{i+1}) : i \ge 0\}$ and $\mathsf{Sup}(\pi) = \sup\{w(v_i, v_{i+1}) : i \ge 0\}$;
- the LimInf (LimSup) payoff, is the minimum (maximum) weight seen infinitely often: $\mathsf{LimInf}(\pi) = \liminf_{i\to\infty} w(v_i, v_{i+1})$ and $\mathsf{LimSup}(\pi) = \limsup_{i\to\infty} w(v_i, v_{i+1})$;
- the *mean-payoff* value of a play, i.e. the limiting average weight, defined using lim inf or lim sup since the running averages might not converge: $\underline{\mathsf{MP}}(\pi) = \liminf_{k\to\infty} \frac{1}{k} w(\langle v_i \rangle_{i<k})$ and $\overline{\mathsf{MP}}(\pi) = \limsup_{k\to\infty} \frac{1}{k} w(\langle v_i \rangle_{i<k})$.

A payoff function $\mathbf{Val}$ is *prefix-independent* if for all plays $\pi = v_0 v_1 \dots$, for all $j \ge 0$, $\mathbf{Val}(\pi) = \mathbf{Val}(\langle v_j \rangle_{j \ge i})$. It is well-known that LimInf, LimSup, $\underline{\mathsf{MP}}$, and $\overline{\mathsf{MP}}$ are prefix-independent. Often, the arguments that we develop work uniformly for these four measures because of their prefix-independent property. Inf and Sup are not prefix-independent but often in the sequel we apply a simple transformation to the game and encode Inf into a LimInf objective, and Sup into a LimSup objective. The transformation consists of encoding in the vertices of the arena the minimal (maximal) weight that has been witnessed by a play, and label the edges of the new graph with this same recorded weight. When this simple transformation does not suffice, we mention it explicitly.

Regret. Consider a fixed weighted arena G, and payoff function $\mathbf{Val}$. Given strategies σ, τ, for Eve and Adam respectively, and $v \in V$, we denote by $\pi^v_{\sigma\tau}$ the unique play starting from v that is consistent with σ and τ and denote its value by: $\mathbf{Val}^v_G(\sigma, \tau) := \mathbf{Val}(\pi^v_{\sigma\tau})$. We omit G if it is clear from the context. If v is omitted, it is assumed to be v_I.

Let $\Sigma_\exists \subseteq \mathfrak{S}_\exists$ and $\Sigma_\forall \subseteq \mathfrak{S}_\forall$ be sets of strategies for Eve and Adam respectively. Given $\sigma \in \Sigma_\exists$ we define the *regret of* σ *in* G *w.r.t.* $\Sigma_\exists$ *and* $\Sigma_\forall$ as:

$$\mathbf{reg}^\sigma_{\Sigma_\exists, \Sigma_\forall}(G) := \sup_{\tau \in \Sigma_\forall} (\sup_{\sigma' \in \Sigma_\exists} \mathbf{Val}(\sigma', \tau) - \mathbf{Val}(\sigma, \tau)).$$

We define the *regret of* G *w.r.t.* $\Sigma_\exists$ *and* $\Sigma_\forall$ as:

$$\mathbf{Reg}_{\Sigma_\exists, \Sigma_\forall}(G) := \inf_{\sigma \in \Sigma_\exists} \mathbf{reg}^\sigma_{\Sigma_\exists, \Sigma_\forall}(G).$$

When $\Sigma_\exists$ or $\Sigma_\forall$ are omitted from $\mathbf{reg}(\cdot)$ and $\mathbf{Reg}(\cdot)$ they are assumed to be the set of all strategies for Eve and Adam.

[2] The values of all functions are not infinite, and therefore in $\mathbb{R}$ since we deal with finite graphs only.

We will make use of two other values associated with the vertices of an arena: the *antagonistic* and *cooperative* values, defined for plays from a vertex $v \in V$ as

$$\mathbf{aVal}^v(G) := \sup_{\sigma \in \mathfrak{S}_\exists} \inf_{\tau \in \mathfrak{S}_\forall} \mathbf{Val}^v(\sigma, \tau) \qquad \mathbf{cVal}^v(G) := \sup_{\sigma \in \mathfrak{S}_\exists} \sup_{\tau \in \mathfrak{S}_\forall} \mathbf{Val}^v(\sigma, \tau).$$

When clear from context G will be omitted, and if v is omitted it is assumed to be v_I.

▶ Remark. It is well-known that **cVal** and **aVal** can be computed in polynomial time, w.r.t. the underlying graph of the given arena, for all payoff functions but MP [4, 6]. For MP, **cVal** is known to be computable in polynomial time for **aVal** it can be done in UP ∩ coUP [17] and in *pseudo-polynomial time* [23, 3].

3 Variant I: Adam plays any strategy

For this variant, we establish that for all the payoff functions that we consider, the problem of computing the antagonistic value and the problem of computing the regret value are *inter-reducible* in polynomial time. As a direct consequence, we obtain the following theorem:

▶ **Theorem 1.** *Deciding if the regret value is less than a given threshold (strictly or non-strictly) is* PTIME*-complete (under log-space reductions) for* Inf*,* Sup*,* LimInf*, and* LimSup*, and equivalent to mean-payoff games (under polynomial-time reductions) for* $\underline{\mathsf{MP}}$ *and* $\overline{\mathsf{MP}}$*.*

Upper bounds. We now describe an algorithm to compute regret for all payoff functions.

▶ **Lemma 2.** *For payoff functions* Inf*,* Sup*,* LimInf*,* LimSup*,* $\underline{\mathsf{MP}}$*, and* $\overline{\mathsf{MP}}$ *computing the regret of a game is at most as hard as computing the antagonistic value of a (polynomial-size) game with the same payoff function.*

Sketch. We describe how the algorithm works for the $\underline{\mathsf{MP}}$ function, the algorithm is similar for all other payoff functions and details are given in the technical report [16]. Let us fix a weighted arena G. We define a new weight function w' as follows. For any edge $e = (u, v)$ let $w'(e) = -\infty$ if $u \in V \setminus V_\exists$, and if $u \in V_\exists$ then $w'(e) = \max\{\mathbf{cVal}^{v'} : (u, v') \in E \setminus \{e\}\}$. Intuitively, w' represents the best value obtainable for a strategy of Eve that differs at the given edge. It is not difficult to see that in order to minimize regret, Eve is trying to simultaneously maximize the value given by the original weight function w, and minimize the maximum w'-weighted edge seen. For $b \in \mathrm{Range}(w')$ we define G^b to be the graph obtained by restricting G to edges e with $w'(e) \leq b$.

Next, we will construct a new weighted arena $\hat{G}$ such that the regret of G is a function of the *antagonistic* value of $\hat{G}$. Figure 3 depicts the general form of the arena we construct. We have three vertices $v_0 \in \hat{V} \setminus \hat{V}_\exists$ and $v_1, v_\perp \in \hat{V}_\exists$ and a "copy" of G as G^b for each $b \in \mathrm{Range}(w') \setminus \{-\infty\}$. We have a self-loop of weight 0 on v_0 which is the initial vertex of $\hat{G}$, a self-loop of weight $-2W - 1$ on $v_\perp$, and weight 0 edges from v_0 to v_1 and from v_1 to the initial vertices of G^b for all b. Recall that G^b might not be total. To fix this we add, for all vertices without a successor, a weight 0 edge to $v_\perp$. The remainder of the weight function $\hat{w}$, is defined for each edge e^b in G^b as $\hat{w}(e^b) = w(e) - b$.

Intuitively, in $\hat{G}$ Adam first decides whether he can ensure a non-zero regret. If this is the case, then he moves to v_1. Next, Eve chooses a maximal value she will allow for strategies which differ from the one she will play (this is the choice of b). The play then moves to the corresponding copy of G, i.e. G^b. She can now play to maximize her mean-payoff value. However, if her choice of b was not correct then the play will end in $v_\perp$. We claim this construction ensures that $\mathbf{Reg}(G) = -\mathbf{aVal}(\hat{G})$. ◀

Lower bounds. For all the payoff functions, from G we can construct in logarithmic space G' such that the antagonistic value of G is equal to the regret value of G', and so we have:

▶ **Lemma 3.** *For payoff functions* Inf, Sup, LimInf, LimSup, $\underline{\mathsf{MP}}$, *and* $\overline{\mathsf{MP}}$ *computing the regret of a game is at least as hard as computing the antagonistic value of a (polynomial-size) game with the same payoff function.*

Sketch. Suppose G is a weighted arena with initial vertex v_I. Consider the weighted arena G' obtained by adding to G the gadget of Figure 5. The initial vertex of G' is set to be v'_I. We claim that the right choice of values for the parameters L, M_1, M_2, N_1, N_2 makes it so that the antagonistic value of G is a function of the regret of the game G'.

For concreteness, let us consider the payoff function $\underline{\mathsf{MP}}$, and let $L = M_1 = M_2 = 0$, $N_1 = W + 1$, and $N_2 = -3W - 2$. At v'_I, Eve has a choice: she can choose to remain in the gadget or she can move to the original game G. If she chooses to remain in the gadget, her payoff will be $-3W - 2$, meanwhile Adam could choose a strategy that would have achieved a payoff of $\mathbf{cVal}(G)$ if she had chosen to play to G. Hence her regret in this case is $\mathbf{cVal}(G) + 3W + 2 \geq 2W + 2$. Otherwise, if she chooses to play to G, she can achieve a payoff of at most $\mathbf{aVal}(G)$ if Adam is adversarial. As $\mathbf{cVal}(G) \leq W$ and W is the maximum possible payoff achievable in G, the strategy of Adam which now maximizes Eve's regret is the one which remains in the gadget – giving a payoff of $W + 1$. Her regret in this case is $K + 1 - \mathbf{aVal}(G) \leq 2W + 1$. Therefore, to minimize her regret she will play this strategy. It follows that $\mathbf{Reg}(G') = W + 1 - \mathbf{aVal}(G)$, and thus the adversarial value of G can be deduced from the regret value of G'. ◀

Memory requirements for Eve and Adam. It follows from the reductions underlying the proof of Lemma 2 that Eve only requires positional strategies to minimize regret when there is no restriction on Adam's strategies. On the other hand, Adam's strategy for maximizing regret consists of a combination of three positional strategies: first he moves to the optimal vertex for deviating, then he plays his optimal (positional) strategy in the antagonistic game. His strategy for the alternative scenario, assuming Eve had deviated, is his optimal strategy in the co-operative game which is also positional. This combined strategy is clearly realizable as a strategy with three memory states, giving us:

▶ **Corollary 4.** *For payoff functions* LimInf, LimSup, $\underline{\mathsf{MP}}$ *and* $\overline{\mathsf{MP}}$*:* $\mathbf{Reg}(G) = \mathbf{Reg}_{\Sigma^1_\exists, \Sigma^3_\forall}(G)$.

The algorithm we give relies on the prefix-independence of the payoff function. As the transformation from Inf and Sup to equivalent prefix-independent ones is polynomial it follows that polynomial memory (w.r.t. the size of the underlying graph of the arena) suffices for both players.

4 Variant II: Adam plays memoryless strategies

For this variant, we provide a polynomial space algorithm to solve the problem for all the payoff functions, we then provide lower bounds.

▶ **Theorem 5.** *Deciding if the regret value is less than a given threshold (strictly or non-strictly) playing against memoryless strategies of Adam is* PSPACE*-complete for* LimInf, $\underline{\mathsf{MP}}$ *and* $\overline{\mathsf{MP}}$*; in* PSPACE *and* coNP*-hard for* Inf, Sup *and* LimSup.

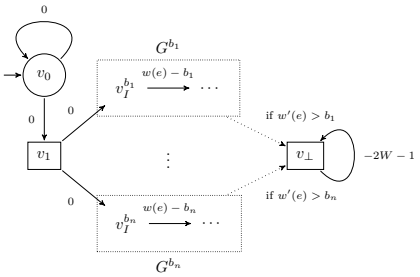

Figure 3 Weighted arena $\hat{G}$, constructed from G. Dotted lines represent several edges added when the condition labelling it is met.

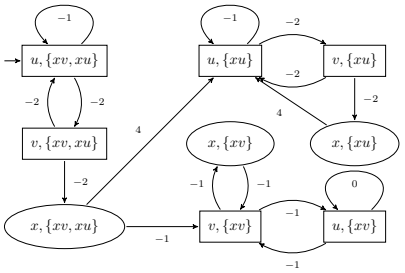

Figure 4 Weighted arena $\hat{G}_1$, constructed from G_1. In the edge set component only edges leaving Adam nodes are depicted.

Upper bounds. Let us now show how to compute regret against positional adversaries.

▶ **Lemma 6.** *For payoff functions* Inf, Sup, LimInf, LimSup, $\underline{\text{MP}}$ *and* $\overline{\text{MP}}$*, the regret of a game played against a positional adversary can be computed in polynomial space.*

Sketch. Once again, we describe how the algorithm works for the $\underline{\text{MP}}$. Given a weighted arena G, we construct a new weighted arena $\hat{G}$ such that we have that $-\mathbf{aVal}(\hat{G})$ is equivalent to the regret of G. The argument works for all prefix independent payoff functions and the details are given in the technical report [16] for Inf, Sup.

The vertices of $\hat{G}$ encode the choices made by Adam. For a subset of edges $D \subseteq E$, let $G{\restriction}D$ denote the weighted arena $(V, V_\exists, E \cap D, w, v_I)$. The new weighted arena $\hat{G}$ is the tuple $(\hat{V}, \hat{V_\exists}, \hat{E}, \hat{w}, \hat{v_I})$ where

(i) $\hat{V} = V \times \mathcal{P}(E)$;
(ii) $\hat{V_\exists} = \{(v, e) \in \hat{V} : v \in V_\exists\}$;
(iii) $\hat{v_I} = (v_I, E)$;
(iv) $\hat{E}$ contains the edge $\big((u, C), (v, D)\big)$ if and only if $(u, v) \in C$ and, either $u \in V_\exists$ and $D = C$, or $u \in V \setminus V_\exists$ and $D = C \setminus \{(u, x) \in E : x \neq v\}$;
(v) $\hat{w}\big((u, C), (v, D)\big) = w(u, v) - \mathbf{cVal}(G{\restriction}D)$.

The application of this transformation for the graph of Fig. 2 is given in Fig. 4.

Finally, we recall that the value of a mean-payoff game is equivalent to the value of its *cycle forming game* [10]. This finite cycle forming game is identical to the mean-payoff game except that it is stopped as soon as a cycle is formed and the value of the game is given by the mean-payoff value of the cycle. It follows that one can use an Alternating Turing Machine to compute the value of $\hat{G}$ in time bounded by the length of the longest simple path in $\hat{G}$: $|V|(|E| + 1)$. Since APTIME = PSPACE, the result follows. ◀

Lower bounds. We give a reduction from the QSAT Problem to the problem of determining whether, given $r \in \mathbb{Q}$, $\mathbf{Reg}_{\mathfrak{S}_\exists, \Sigma^1_\forall}(G) \lhd r$ for the payoff functions LimInf, $\underline{\text{MP}}$, and $\overline{\text{MP}}$ (for $\lhd \in \{<, \leq\}$). Then we provide a reduction from the complement of the 2-disjoint-paths Problem for LimSup, Sup, and Inf.

▶ **Lemma 7.** *For $r \in \mathbb{Q}$, weighted arena G and payoff function* LimInf, $\underline{\text{MP}}$, *or* $\overline{\text{MP}}$*, determining whether $\mathbf{Reg}_{\mathfrak{S}_\exists, \Sigma^1_\forall}(G) \lhd r$, for $\lhd \in \{<, \leq\}$, is* PSPACE*-hard.*

Sketch. The crux of the reduction from QSAT is a gadget for each clause of the QSAT formula. Visiting this gadget allows Eve to gain information about the highest payoff obtainable in the gadget, each entry point corresponds to a literal from the clause, and the

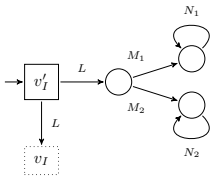

Figure 5 Gadget to reduce a game to its regret game.

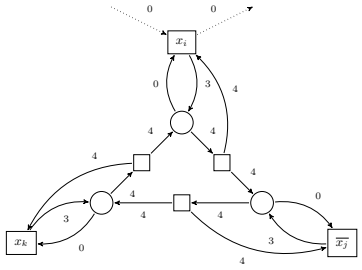

Figure 6 Clause gadget for the QBF reduction for clause $x_i \vee \neg x_j \vee x_k$.

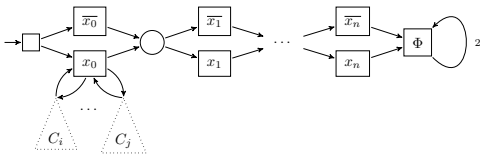

Figure 7 Depiction of the reduction from QBF.

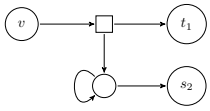

Figure 8 Regret gadget for 2-disjoint-paths reduction.

literal is visited when it is made true by the valuation of variables chosen by Eve and Adam in the reduction described below. Figure 6 depicts an instance of the gadget for a particular clause. Let us focus on the mean-payoff function. Note that staying in the inner 6-vertex triangle would yield a mean-payoff value of 4. However, in order to do so, Adam needs to cooperate with Eve at all three corner vertices. Also note that if he does cooperate in at least one of these vertices then Eve can secure a payoff value of at least $\frac{11}{3}$.

The complete reduction for MP consists in describing how to construct, from an input QBF Φ, a weighted arena G of linear size w.r.t. Φ. In G, Eve can ensure $\mathbf{Reg}_{\mathfrak{S}_\exists, \Sigma^1_\forall}(G) < 2$ if and only if the QBF true. We assume Φ is in 3-CNF and w.l.o.g. we assume that each clause contains at least one existentially quantified variable.

It is common to consider a QBF as a game between an existential and a universal player. The game we construct mimics the choices of the existential and universal player and makes sure that the regret of the game is smaller than 2 if and only if Φ is true. Figure 7 depicts the general structure of the game. Eve and Adam choose valuations for the variables. Depending on these choices, clause gadgets and literals that are made true by the constructed valuation can be visited by Eve.

Assume the QBF is true, then Eve has a value-choosing strategy s.t. for any strategy of Adam, all clauses have at least one literal which holds. Hence, Eve can ensure to visit every clause gadget. If Adam helps Eve in any of these gadgets then she can ensure a payoff of $\frac{11}{3}$ as explained above. Otherwise, she arrives at Φ, gets mean-payoff 2 and, since Adam did not help her in any of the clause gadgets, she is sure that no alternative strategy can achieve a payoff value of 4. Hence, the regret of the game is less than 2.

Conversely, if the QBF is false then Adam can make sure Eve does not visit at least one clause gadget that corresponds to a clause that is false in the constructed valuation. Additionally, in every clause gadget she does visit, he does not help her. She can now ensure a mean-payoff value of 2 by going to Φ but an alternative strategy of Eve can force a visit to the gadget not visited (as each clause contains at least one existentially quantified variable) and there Adam can now fully cooperate and ensures a payoff of 4 to Eve. ◀

▶ **Lemma 8.** *For $r \in \mathbb{Q}$, weighted arena G and payoff function* Inf, Sup, *or* LimSup, *determining whether* $\mathbf{Reg}_{\mathfrak{S}_\exists,\Sigma^1_\forall}(G) \lhd r$, *for* $\lhd \in \{<, \leq\}$, *is* coNP*-hard.*

Proof. We provide a reduction from the complement of the 2-DISJOINT-PATHS PROBLEM on directed graphs [11]. As the problem is known to be NP-complete, the result follows. In other words, we sketch how to translate a given instance of the 2-DISJOINT-PATHS PROBLEM into a weighted arena in which Eve can ensure regret value strictly less than 1 if and only if the answer to the 2-DISJOINT-PATHS PROBLEM is negative.

Consider a directed graph G and distinct vertex pairs (s_1, t_1) and (s_2, t_2). W.l.o.g. we assume that for all $i \in \{1, 2\}$:

(i) t_i is reachable from s_i, and
(ii) t_i is a sink (i.e. has no outgoing edges)

in G. We now describe the changes we apply to G in order to get the underlying graph structure of the weighted arena and then comment on the weight function. Let all vertices from G be Adam vertices and s_1 be the initial vertex. We replace all edges (v, t_1) incident on t_1 by a copy of the gadget shown in Figure 8. Next, we add self-loops on t_1 and t_2 with weights 1 and 2, respectively. Finally, the weights of all remaining edges are 0.

We claim that, in this weighted arena, Eve can ensure regret strictly less than 1 – for payoff functions Sup and LimSup – if and only if in G the vertex pairs (s_1, t_1) and (s_2, t_2) cannot be joined by vertex-disjoint paths. Indeed, we claim that the strategy that minimizes the regret of Eve is the strategy that, in states where she has a choice, tells her to go to t_1.

First, let us prove that this strategy has regret strictly less than 1 if and only if no two disjoint paths in the graph exist between the pairs of states (s_1, t_1) and (s_2, t_2). Assume the latter is the case. Then if Adam chooses to always avoid t_1, then clearly the regret is 0. If t_1 is eventually reached, then the choice of Eve secures a value of 1 (for all payoff functions). Note that if she had chosen to go towards s_2 instead, as there are no two disjoint paths, we know that either the path constructed from s_2 by Adam never reaches t_2, and then the value of the path is 0 – and the regret is 0 for Eve– or the path constructed from s_2 reaches t_1 again – and, again, the regret is 0 for Eve. Now assume that two disjoint paths between the source-target pairs exist. If Eve changed her strategy to go towards s_2 (instead of choosing t_1) then Adam has a strategy to reach t_2 and achieve a payoff of 2. Thus, her regret would be equal to 1.

Second, we claim that any other strategy of Eve has a regret greater than or equal to 1. Indeed, if Eve decides to go towards s_2 (instead of choosing to go to t_1) then Adam can choose to loop on the state before s_2 and the payoff in this case is 0. Hence, the regret of Eve is at least 1.

Note that minimal changes are required for the same construction to imply the result for Inf. Further, the weight function and threshold r can be accommodated so that Eve wins for the non-strict regret threshold. Hence, the general result follows. ◀

Memory requirements for Eve. It follows from our algorithms for computing regret in this variant that Eve only requires strategies with exponential memory. Examples where exponential memory is necessary can be easily constructed.

▶ **Corollary 9.** *For all payoff functions* Sup, Inf, LimSup, LimInf, $\underline{\text{MP}}$ *and* $\overline{\text{MP}}$, *for all game graphs G, there exists m which is $2^{\mathcal{O}(|G|)}$ such that:* $\mathbf{Reg}_{\mathfrak{S}_\exists,\Sigma^1_\forall}(G) = \mathbf{Reg}_{\Sigma^m_\exists,\Sigma^1_\forall}(G)$.

5 Variant III: Adam plays word strategies

For this variant, we provide tight upper and lower bounds for all the payoff functions: the regret threshold problem is EXPTIME-complete for Sup, Inf, LimSup, and LimInf, and undecidable for $\underline{\text{MP}}$ and $\overline{\text{MP}}$. For the later case, the decidability can be recovered when we fix a priori the size of the memory that Eve can use to play, the decision problem is then NP-complete. Finally, we show that our notion of regret minimization for word strategies generalizes the notion of *good for games* introduced by Henzinger and Piterman in [14], and we also formalize the relation that exists with the notion of determinisation by pruning for weighted automata introduced by Aminof et al. in [1].

Additional definitions. We say that a strategy of Adam is a *word strategy* if his strategy can be expressed as a function $\tau : \mathbb{N} \to [\max\{deg^+(v) : v \in V\}]$, where $[n] = \{i : 1 \leq i \leq n\}$. Intuitively, we consider an order on the successors of each Adam vertex. On every turn, the strategy τ of Adam will tell him to move to the i-th successor of the vertex according to the fixed order. We denote by $\mathfrak{W}_\forall$ the set of all such strategies for Adam. When considering word strategies, it is more natural to see the arena as a (weighted) automaton.

A *weighted automaton* is a tuple $\Gamma = (Q, q_I, A, \Delta, w)$ where A is a finite alphabet, Q is a finite set of states, q_I is the initial state, $\Delta \subseteq Q \times A \times Q$ is the transition relation, $w : \Delta \to \mathbb{Z}$ assigns weights to transitions. A *run* of Γ on a word $a_0 a_1 \ldots \in A^\omega$ is a sequence $\rho = q_0 a_0 q_1 a_1 \ldots \in (Q \times A)^\omega$ such that $(q_i, a_i, q_{i+1}) \in \Delta$, for all $i \geq 0$, and has *value* $\mathbf{Val}(\rho)$ determined by the sequence of weights of the transitions of the run and the payoff function. The value Γ assigns to a word is the supremum of the values of all its runs on the word. We say the automaton is deterministic if Δ is functional.

A game in which Adam plays word strategies can be reformulated as a game played on a weighted automaton $\Gamma = (Q, q_I, A, \Delta, w)$ and strategies of Adam – of the form $\tau : \mathbb{N} \to A$ – determine a sequence of input symbols to which Eve has to react by choosing Δ-successor states starting from q_I. In this setting a strategy of Eve which minimizes regret defines a run by resolving the non-determinism of Δ in Γ, and ensures the difference of value given by the constructed run is minimal w.r.t. the value of the best run on the word spelled out by Adam. The following result summarizes the results of this section:

▶ **Theorem 10.** *Deciding if the regret value is less than a given threshold (strictly or non-strictly) playing against word strategies of Adam is* EXPTIME*-complete for* Inf*,* Sup*,* LimInf*, and* LimSup*; it is undecidable for* $\underline{\text{MP}}$ *and* $\overline{\text{MP}}$*.*

Upper bounds. There is an EXPTIME algorithm for solving the regret threshold problem for Inf, Sup, LimInf, and LimSup. This algorithm is obtained by a reduction to parity games.

▶ **Lemma 11.** *For $r \in \mathbb{Q}$, weighted automaton Γ and payoff function* Inf*,* Sup*,* LimInf*, or* LimSup*, determining whether $\mathbf{Reg}_{\mathfrak{S}_\exists, \mathfrak{W}_\forall}(\Gamma) \lhd r$, for $\lhd \in \{<, \leq\}$, can be done in exponential time.*

Sketch. We focus, for this sketch, on the LimInf payoff function. Our decision algorithm consists in first building a deterministic automaton for $\Gamma = (Q_1, q_I, A, \Delta_1, w_1)$ using the construction provided in [6]. We denote by $D_\Gamma = (Q_2, s_I, A, \Delta_2, w_2)$ this deterministic automaton and we know that it is at most exponentially larger than Γ. Then we shift the weights of the automaton D_Γ by $-r$, and we denote it by D_Γ^{-r}. Finally, we need to decide if Eve is able to simulate D_Γ^{-r} on Γ in the sense of [6]: she must be able to resolve non-determinism in Γ on letters given by Adam in a way that the run constructed in Γ has a

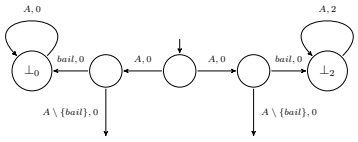

Figure 9 Initial gadget used in reduction from countdown games.

value greater than or equal to the value of the unique run of D_Γ^{-r} on the word constructed by Adam. This simulation problem can be reduced to determining the winner in a parity game. In our case, the parity game that we need is linear in the size of the product between D_Γ^{-r} and Γ, and so exponential in the size of Γ, but uses a polynomial number of priorities. Solving this parity game can be done in exponential time in the size of Γ, giving us the required upper bound for the regret threshold problem. Details for LimInf and the other measures are given in the technical report [16]. ◀

Lower bounds. We first establish EXPTIME-hardness for the payoff functions Inf, Sup, LimInf, and LimSup by giving a reduction from countdown games [18]. That is, we show that given a countdown game, we can construct a game where Eve ensures regret less than 2 if and only if Counter wins in the original countdown game.

▶ **Lemma 12.** *For $r \in \mathbb{Q}$, weighted automaton Γ and payoff function* Inf, Sup, LimInf, *or* LimSup, *determining whether $\mathbf{Reg}_{\mathfrak{S}_\exists,\mathfrak{W}_\forall}(\Gamma) \lhd r$, for $\lhd \in \{<, \leq\}$, is* EXPTIME*-hard.*

To show undecidability of the problem for the mean-payoff function we give a reduction from the threshold problem in *mean-payoff games with partial-observation*. This problem was shown to be undecidable in [9, 15].

▶ **Lemma 13.** *For $r \in \mathbb{Q}$, weighted automaton Γ and payoff function* $\underline{\mathsf{MP}}$ *or* $\overline{\mathsf{MP}}$*, determining whether $\mathbf{Reg}_{\mathfrak{S}_\exists,\mathfrak{W}_\forall}(\Gamma) \lhd r$, for $\lhd \in \{<, \leq\}$, is undecidable even if Eve is only allowed to play finite memory strategies.*

Fixed memory for Eve. Since the problem is EXPTIME-hard for most payoff functions and already undecidable for $\underline{\mathsf{MP}}$ and $\overline{\mathsf{MP}}$, we now fix the memory Eve can use.

▶ **Theorem 14.** *For $r \in \mathbb{Q}$, weighted automaton Γ and payoff function* Inf, Sup, LimInf, LimSup, $\underline{\mathsf{MP}}$, *or* $\overline{\mathsf{MP}}$*, determining whether $\mathbf{Reg}_{\Sigma_\exists^m,\mathfrak{W}_\forall}(\Gamma) \lhd r$, for $\lhd \in \{<, \leq\}$, can be done in* NTIME$(m^2|\Gamma|^2)$.

Sketch. Guess the strategy σ of Eve. Consider the non-deterministic automaton constructed from the synchronous product of the original machine and the deterministic automaton defined by the automaton restricted to transitions allowed by σ, this clearly has size at most $m|\Gamma|$. The weights of the transitions of the new automaton are set to the difference of the values of the functions of the two original automata. The language of the new machine is empty (for accepting threshold r) if and only if the desired property holds. As emptiness of a weighted automaton $\mathcal{A}$ can be decided in $O(|\mathcal{A}|^2)$ time [6], the result follows. ◀

We provide a matching lower bound. The proof is an adaptation of the NP-hardness proof from [1] (all the details are provided in the technical report [16]).

▶ **Theorem 15.** *For $r \in \mathbb{Q}$, weighted automaton Γ and payoff function* Inf, Sup, LimInf, LimSup, $\underline{\mathsf{MP}}$, *or* $\overline{\mathsf{MP}}$*, determining whether $\mathbf{Reg}_{\Sigma_\exists^1,\mathfrak{W}_\forall}(\Gamma) \lhd r$, for $\lhd \in \{<, \leq\}$, is* NP*-hard.*

Relation to other works. Let us first extend the definitions of *approximation, embodiment* and *refinement* from [1] to the setting of ω-words. Consider two weighted automata $\mathcal{A} = (Q_\mathcal{A}, q_I, A, \Delta_\mathcal{A}, w_\mathcal{A})$ and $\mathcal{B} = (Q_\mathcal{B}, q_I, A, \Delta_\mathcal{B}, w_\mathcal{B})$ and let $d : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be a *metric*.[3] We say $\mathcal{B}$ *(strictly) α-approximates* $\mathcal{A}$ *(with respect to d)* if $d(\mathcal{B}(w), \mathcal{A}(w)) \leq \alpha$ (resp. $d(\mathcal{B}(w), \mathcal{A}(w)) < \alpha$) for all words $w \in \Sigma^\omega$. We say $\mathcal{B}$ *embodies* $\mathcal{A}$ if $Q_\mathcal{A} \subseteq Q_\mathcal{B}$, $\Delta_\mathcal{A} \subseteq \Delta_\mathcal{B}$ and $w_\mathcal{A}$ agrees with $w_\mathcal{B}$ on $\Delta_\mathcal{A}$. For an automaton $\mathcal{A} = (Q, q_I, A, \Delta, w)$ and an integer $k \geq 0$, the k-refinement of $\mathcal{A}$ is the automaton obtained by refining the state space of $\mathcal{A}$ using k boolean variables. The automaton $\mathcal{A}$ is said to be *(strictly) (α, k)-determinisable by pruning* if the k-refinement of $\mathcal{A}$ embodies a deterministic automaton which (strictly) α-approximates $\mathcal{A}$. The next result follows directly from the above definitions.

▶ **Proposition 16.** *For non-negative $\alpha \in \mathbb{Q}$, $k \in \mathbb{N}$, a weighted automaton Γ is (strictly) (α, k)-DBP (w.r.t. the difference metric) iff $\mathbf{Reg}_{\Sigma^{2^k}_\exists, \mathfrak{W}_\forall}(\Gamma) \leq \alpha$ (resp. $\mathbf{Reg}_{\Sigma^{2^k}_\exists, \mathfrak{W}_\forall}(\Gamma) < \alpha$).*

In [14] the authors define *good for games automata*. Their definition is based on a game which is played on an ω-automaton by Spoiler and Simulator. We propose the following generalization of the notion of good for games automata for weighted automata. A weighted automaton $\mathcal{A}$ is *(strictly) α-good for games* if Simulator, against any word $w \in A^\omega$ spelled by Spoiler, can resolve non-determinism in $\mathcal{A}$ so that the resulting run has value v and $d(v, \mathcal{A}(w)) \leq \alpha$ (resp. $d(v, \mathcal{A}(w)) < \alpha$), for some metric d. We summarize the relationship that follows from the definition in the following result:

▶ **Proposition 17.** *For non-negative $\alpha \in \mathbb{Q}$, a weighted automaton Γ is (strictly) α-good for games (w.r.t. the difference metric) iff $\mathbf{Reg}_{\mathfrak{S}_\exists, \mathfrak{W}_\forall}(\Gamma) \leq \alpha$ (resp. $\mathbf{Reg}_{\mathfrak{S}_\exists, \mathfrak{W}_\forall}(\Gamma) < \alpha$).*

Acknowledgements. We thank Udi Boker for his comments on how to determinise LimSup automata.

References

1 Benjamin Aminof, Orna Kupferman, and Robby Lampert. Reasoning about online algorithms with weighted automata. *ACM Transactions on Algorithms*, 2010.

2 David E. Bell. Regret in decision making under uncertainty. *Operations Research*, 30(5):961–981, 1982.

3 Lubos Brim, Jakub Chaloupka, Laurent Doyen, Raffaella Gentilini, and Jean-François Raskin. Faster algorithms for mean-payoff games. *Formal Methods in System Design*, 38(2):97–118, 2011.

4 Arindam Chakrabarti, Luca de Alfaro, Thomas A. Henzinger, and Mariëlle Stoelinga. Resource interfaces. In *EMSOFT*, volume 2855 of *LNCS*, pages 117–133. Springer, 2003.

5 Krishnendu Chatterjee, Laurent Doyen, Emmanuel Filiot, and Jean-François Raskin. Doomsday equilibria for omega-regular games. In *VMCAI*, volume 8318, pages 78–97. Springer, 2014.

6 Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM Trans. Comput. Log.*, 11(4), 2010.

7 Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. Generalized mean-payoff and energy games. In *FSTTCS*, pages 505–516, 2010.

8 Werner Damm and Bernd Finkbeiner. Does it pay to extend the perimeter of a world model? In *FM*, volume 6664 of *LNCS*, pages 12–26. Springer, 2011.

[3] The metric used in [1] is the ratio measure.

9 Aldric Degorre, Laurent Doyen, Raffaella Gentilini, Jean-François Raskin, and Szymon Toruńczyk. Energy and mean-payoff games with imperfect information. In *CSL*, pages 260–274, 2010.

10 A. Ehrenfeucht and J. Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8:109–113, 1979.

11 Tali Eilam-Tzoreff. The disjoint shortest paths problem. *Discrete Applied Mathematics*, 85(2):113–138, 1998.

12 Emmanuel Filiot, Tristan Le Gall, and Jean-François Raskin. Iterated regret minimization in game graphs. In *MFCS*, volume 6281 of *LNCS*, pages 342–354. Springer, 2010.

13 Joseph Y. Halpern and Rafael Pass. Iterated regret minimization: A new solution concept. *Games and Economic Behavior*, 74(1):184–207, 2012.

14 Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In *CSL*, pages 395–410, 2006.

15 Paul Hunter, Guillermo A. Pérez, and Jean-François Raskin. Mean-payoff games with partial-observation (extended abstract). In *Reachability Problems*, pages 163–175, 2014.

16 Paul Hunter, Guillermo A. Pérez, and Jean-François Raskin. Reactive synthesis without regret. *CoRR*, abs/1504.01708, 2015.

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

18 Marcin Jurdzinski, Jeremy Sproston, and François Laroussinie. Model checking probabilistic timed automata with one or two clocks. *LMCS*, 4(3), 2008.

19 Christos H. Papadimitriou and Mihalis Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84(1):127–150, 1991.

20 A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190. ACM, ACM Press, 1989.

21 Min Wen, Ruediger Ehlers, and Ufuk Topcu. Correct-by-synthesis reinforcement learning with temporal logic constraints. *arXiv preprint arXiv:1503.01793*, 2015.

22 Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *NIPS*, pages 905–912, 2008.

23 Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *TCS*, 158(1):343–359, 1996.