# Quantitative Games under Failures[*]

**Thomas Brihaye[1], Gilles Geeraerts[2], Axel Haddad[1], Benjamin Monmege[3], Guillermo A. Pérez[†2], and Gabriel Renault[1]**

1 **Université de Mons, Belgium**
`{thomas.brihaye,axel.haddad,gabriel.renault}@umons.ac.be`
2 **Université libre de Bruxelles, Belgium**
`{gigeerae,gperezme}@ulb.ac.be`
3 **LIF, Aix-Marseille Université, CNRS, France**
`benjamin.monmege@lif.univ-mrs.fr`

## Abstract

We study a generalisation of sabotage games, a model of dynamic network games introduced by van Benthem [16]. The original definition of the game is inherently finite and therefore does not allow one to model infinite processes. We propose an extension of the sabotage games in which the first player (Runner) traverses an arena with dynamic weights determined by the second player (Saboteur). In our model of *quantitative sabotage games*, Saboteur is now given a budget that he can distribute amongst the edges of the graph, whilst Runner attempts to minimise the quantity of budget witnessed while completing his task. We show that, on the one hand, for most of the classical cost functions considered in the literature, the problem of determining if Runner has a strategy to ensure a cost below some threshold is EXPTIME-complete. On the other hand, if the budget of Saboteur is fixed a priori, then the problem is in PTIME for most cost functions. Finally, we show that restricting the dynamics of the game also leads to better complexity.

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

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

**Digital Object Identifier** 10.4230/LIPIcs.FSTTCS.2015.293

## 1 Introduction

Two-player games played on graphs are nowadays a well-established model for systems where two antagonistic agents interact. In particular, they allow one to perform controller synthesis [1], when one of the players models the controller, and the second plays the role of an evil environment. Quantitative generalisations (played on weighted graphs) of these models have attracted much attention in the last decades [5, 8, 3] as they allow for a finer analysis of those systems.

In this setting, most results assume that the arena (i.e., the graph) on which the game is played does not change during the game. There are however many situations where this restriction is not natural, at least from a modelling point of view. For instance, Grüner *et al.* [7] model connectivity problems in *dynamic* networks (i.e., subject to failure and restoration) using a variant of *sabotage games* – a model originally proposed by van Benthem [16] – to model *reachability problems* in a network prone to errors. A sabotage game is played on a directed graph, and starts with a token in an initial vertex. Then, Runner and Saboteur (the

* The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under Grant Agreement n° 601148 (CASSTING).
† Author supported by F.R.S.-FNRS fellowship.

two players of the game) play in alternation: Runner moves the token along one edge and Saboteur is allowed to remove one edge. Runner wins the game if he reaches a target set of vertices. In [12], it is shown that deciding the existence of a winning strategy for Runner is PSPACE-complete.

In those sabotage games, errors are regarded as unrecoverable failures. In practice, this hypothesis might be too strong. Instead, one might want to model the fact that certain uncontrollable events incur additional costs (modelling delays, resource usage. . . ), and look for strategies that allow one to fulfil the game objective *at a minimal cost*, whatever the occurrence of uncontrollable events. For instance, if the graph models a railway network, the failure of a track will eventually be fixed, and, in the meantime, trains might be slowed down on the faulty portion or diverted, creating delays in the journeys. It is thus natural to consider *quantitative* extensions of sabotage games, where Saboteur controls the price of the actions in the game. This is the aim of the present paper.

More precisely, we extend sabotage games in two directions. First, we consider games played on *weighted* graphs. Saboteur is allotted an integral budget $B$ that he can distribute (dividing it into integral parts) on the edges of the graph, thereby setting their weights. At each turn, Saboteur can change this distribution by moving $k$ units of budget from an edge to another edge (for simplicity, we restrict ourselves to $k = 1$ but our results hold for any $k$). Second, we relax the inherent finiteness of sabotage games (all edges will eventually be deleted), and consider infinite horizon games (i.e., plays are now infinite). In this setting, the goal of Runner is to minimise the cost defined by the sequence of weights of edges visited, with respect to some fixed cost function (Inf, Sup, LimInf, LimSup, average or discounted-sum), while Saboteur attempts to maximise the same cost. We call these games *quantitative sabotage games* (QSG, for short).

Let us briefly sketch one potential application of our model, showing that they are useful to perform synthesis in a dynamic environment. Our application is borrowed from Suzuki and Yamashita [17] who have considered the problem of *motion planning* of multiple mobile robots that interact in a finite space. In essence, each robot executes a "Look-Compute-Move" cycle and should realise some specification (that we could specify using LTL, for instance). For simplicity, assume that at every observation (Look) phase, at most one other robot has moved. Clearly every motion phase (Move) will require different amounts of time and energy depending on the location of the other robots. We can model the interaction of each individual robot against all others using a QSG where Runner is one robot, Saboteur is the coalition of all other robots, and the budget is equal to the number of robots minus 1. This model allows one to answer meaningful questions such as '*what is, in the worst case, the average delay the robot incurs because of the dynamics of the system?*', or '*what is the average amount of additional energy required because of the movements of the other robots?*' using appropriate cost functions.

As a second motivational example, let us recall the motivation of the original Sabotage Game: consider a situation in which you need to find your way between two cities within a railway network where a malevolent demon starts cancelling connections? This is called the *real Travelling Salesman Problem* by van Benthem [16]. However, in real life, railway companies have contracts with infrastructure companies which ensure that failures in the railway network are repaired withing a given amount of time (e.g. a service-level agreement). In this case, it is better to consider delays instead of absolute failures in the network. Further, salesmen do not usually have one single trip in their whole carriers. For modelling purposes, one can in fact assume they never stop travelling. In this setting, QSGs can be used to answer the question: '*what is, in the worst case, the average delay time incurred by the salesman*'?

**Table 1** Complexity results for quantitative sabotage games.

| | QSG | static QSG | fixed budget QSG |
|---|---|---|---|
| Inf, LimInf | $\in$ EXPTIME | $\in$ PTIME | $\in$ PTIME |
| Sup, LimSup, Avg | EXPTIME-c | coNP-c | $\in$ PTIME |
| DS | EXPTIME-c | coNP-c | $\in$ NP $\cap$ coNP |

Our model can be used to treat the same questions for other networks and not just railway networks.

**Related Works & Contributions.** Variations of the original sabotage games have been considered by students of van Benthem. In [11], the authors have considered changing the *reachability objective* of Runner to a *safety objective*, and proved it is PSPACE-complete as well. They also consider a co-operative variation of the game which, not surprisingly, leads to a lower complexity: NL-complete. In [14], an asymmetric imperfect information version of the game is studied—albeit, under the guise of the well-known parlor game *Scotland Yard*—and shown to be PSPACE-complete. We remark that although the latter version of sabotage games already includes some sort of dynamicity in the form of the *Scotland Yard* team moving their pawns on the board, both of these studies still focus on inherently finite versions of the game.

We establish that QSGs are EXPTIME-complete in general. Our approach is to prove the result for a very weak problem on QSGs, called the *safety problem*, that asks whether Runner can avoid *ad vitam æternam* edges with non-zero budget on it. We remark that although the safety problem is related to cops and robbers games [1, 6], we were not able to find EXPTIME-hard variants that reduce easily into our formalism. The general problem being EXPTIME-complete, we consider the case where the budget is fixed instead of left as an input of the problem (see Corollary 2). We also consider restricting the behaviour of Saboteur and define a variation of our QSGs in which Saboteur is only allowed to choose an initial distribution of weights but has to commit to it once he has fixed it. We call this the *static* version of the game. For both restrictions, we show that tractable algorithms exist for some of the cost functions we consider. A summary of the complexity results we establish in this work is shown in Table 1. In Section 6, we comment on several implications of the complexity bounds proved in this work.

Some proofs and technical details may be found in the long version [2].

## 2 Quantitative sabotage games

Let us now formally define quantitative sabotage games (QSG). We start with the definition of the cost functions we will consider, then give the syntax and semantics of QSG.

**Cost functions.** A *cost function* $f\colon \mathbb{Q}^\omega \to \mathbb{R}$ associates a real number to a sequence of rationals $u = (u_i)_{i\geqslant 0} \in \mathbb{Q}^\omega$. The six classical cost functions that we consider are

- $\mathsf{Inf}(u) = \inf\{u_i \mid i \geqslant 0\}$;
- $\mathsf{Sup}(u) = \sup\{u_i \mid i \geqslant 0\}$;
- $\mathsf{LimInf}(u) = \liminf_{n\to\infty}\{u_i \mid i \geqslant n\}$;
- $\mathsf{LimSup}(u) = \limsup_{n\to\infty}\{u_i \mid i \geqslant n\}$;
- $\mathsf{Avg}(u) = \liminf_{n\to\infty} \frac{1}{n}\sum_{i=0}^{n} u_i$, which stands for the average cost (also called *mean-payoff* in the literature); and
- $\mathsf{DS}_\lambda(u) = \sum_{i=0}^{\infty} \lambda^i \cdot u_i$, (with $0 < \lambda < 1$), stands for discounted-sum.

In the following, we let $\mathsf{DS} = \{\mathsf{DS}_\lambda \mid 0 < \lambda < 1\}$.

**Syntax.** As sketched in the introduction, *quantitative sabotage games* are played by Runner and Saboteur on a directed weighted graph, called the *arena*. A play alternates between Runner moving the token along the edges and Saboteur modifying the weights. We consider that Saboteur has a fixed integer budget $B$ that he can distribute on edges, thereby setting their weights (which must be integer values). Formally, for a finite set $E$ and a budget $B \in \mathbb{N}$, $\Delta(E, B)$ denotes the set of all *distributions* of budget $B$ on $E$, where a distribution is a function $\delta\colon E \to \{0, 1, \ldots, B\}$ such that $\sum_{e \in E} \delta(e) \leqslant B$ (the last constraint is an inequality since the whole budget need not be distributed on $E$). Then, a *quantitative sabotage game* is a tuple $\mathcal{G} = (V, E, B, v_I, \delta_I, f)$, where $(V, E)$ is a directed graph, $B \in \mathbb{N}$ is the budget of the game, $v_I \in V$ is the initial vertex, $\delta_I \in \Delta(E, B)$ is the initial distribution of the budget, and $f$ is a cost function. We assume, without loss of generality, that there are no deadlocks in $(V, E)$, i.e., for all $v \in V$, there is $v' \in V$ such that $(v, v') \in E$. In the following, we may alternatively write $\Delta(\mathcal{G})$ for $\Delta(E, B)$ when $\mathcal{G}$ is a $\mathsf{QSG}$ with set of edges $E$ and budget $B$.

**Semantics.** To define the semantics of a $\mathsf{QSG}$ $\mathcal{G}$, we first formalise the possible redistributions of the budget by Saboteur. We choose to restrict them, reflecting some physical constraints: Saboteur can move at most one unit of weight in-between two edges. For $\delta, \delta' \in \Delta(\mathcal{G})$, we say that $\delta'$ is a *valid redistribution* from $\delta$, noted $\delta \triangleright \delta'$, if and only if there are $e_1, e_2 \in E$ such that $\delta'(e_1) \in \{\delta(e_1), \delta(e_1) - 1\}$, $\delta'(e_2) \in \{\delta(e_2), \delta(e_2) + 1\}$, and for all other edges $e \notin \{e_1, e_2\}$, $\delta'(e) = \delta(e)$. Then, a *play* in a $\mathsf{QSG}$ $\mathcal{G} = (V, E, B, v_I, \delta_I, f)$ is an infinite sequence $\pi = v_0\delta_0v_1\delta_1 \cdots$ alternating vertices $v_i \in V$ and budget distributions $\delta_i \in \Delta(\mathcal{G})$ such that

**(i)** $v_0 = v_I$;
**(ii)** $\delta_0 = \delta_I$; and
**(iii)** for all $i \geqslant 0$: $(v_i, v_{i+1}) \in E$, and $\delta_i \triangleright \delta_{i+1}$.

Let $\mathrm{Prefs}_\Delta(\mathcal{G})$ denote the set of prefixes of plays ending in a budget distribution, and $\mathrm{Prefs}_V(\mathcal{G})$ the set of prefixes of length at least 2 ending in a vertex. We abuse notations and lift cost functions $f$ to plays letting $f(v_0\delta_0v_1\delta_1 \cdots) = f(\delta_0(v_0, v_1)\delta_1(v_1, v_2) \cdots)$. A *strategy* of Runner is a mapping $\rho\colon \mathrm{Prefs}_\Delta(\mathcal{G}) \to V$ such that $(v_n, \rho(\pi)) \in E$ for all $\pi = v_0\delta_0 \cdots v_n\delta_n \in \mathrm{Prefs}_\Delta(\mathcal{G})$. A strategy of Saboteur is a mapping $\sigma\colon \mathrm{Prefs}_V(\mathcal{G}) \to \Delta(\mathcal{G})$ such that $\delta_{n-1} \triangleright \sigma(\pi)$ for all $\pi = v_0\delta_0 \cdots v_{n-1}\delta_{n-1}v_n \in \mathrm{Prefs}_V(\mathcal{G})$. We denote by $\Sigma_{\mathrm{Run}}(\mathcal{G})$ (respectively, $\Sigma_{\mathrm{Sab}}(\mathcal{G})$) the set of all strategies of Runner (respectively, Saboteur). A pair of strategies $(\rho, \sigma)$ of Runner and Saboteur defines a unique play $\pi_{\rho,\sigma} = v_0\delta_0v_1\delta_1 \cdots$ such that for all $i \geqslant 0$:

**(i)** $v_{i+1} = \rho(v_0\delta_0 \cdots v_i\delta_i)$; and
**(ii)** $\delta_{i+1} = \sigma(v_0\delta_0 \cdots v_i\delta_iv_{i+1})$.

**Values and determinacy.** We are interested in computing the best value that each player can guarantee no matter how the other player plays. To reflect this, we define two values of a $\mathsf{QSG}$ $\mathcal{G}$: the superior value (modelling the best value for Runner)as $\overline{\mathbf{Val}}(\mathcal{G}) := \sup_{\sigma \in \Sigma_{\mathrm{Sab}}(\mathcal{G})} \inf_{\rho \in \Sigma_{\mathrm{Run}}(\mathcal{G})} f(\pi_{\rho,\sigma})$, and the inferior value (modelling the best value for Saboteur) as $\underline{\mathbf{Val}}(\mathcal{G}) := \inf_{\rho \in \Sigma_{\mathrm{Run}}(\mathcal{G})} \sup_{\sigma \in \Sigma_{\mathrm{Sab}}(\mathcal{G})} f(\pi_{\rho,\sigma})$. It is folklore to prove that $\underline{\mathbf{Val}}(\mathcal{G}) \leqslant \overline{\mathbf{Val}}(\mathcal{G})$. Indeed, for the previously mentioned cost functions, we can prove that $\mathsf{QSG}$s are determined, i.e., that $\underline{\mathbf{Val}}(\mathcal{G}) = \overline{\mathbf{Val}}(\mathcal{G})$ for all $\mathsf{QSG}$s $\mathcal{G}$. This can be formally proved by encoding a $\mathsf{QSG}$ $\mathcal{G}$ into a quantitative two-player game $[\![\mathcal{G}]\!]$ (whose vertices contain both vertices of $\mathcal{G}$ and budget distributions), and then using classical Martin's determinacy theorem [13]. $\underline{\mathbf{Val}}(\mathcal{G}) = \overline{\mathbf{Val}}(\mathcal{G})$ is henceforth called the *value of* $\mathcal{G}$, and denoted by $\mathbf{Val}(\mathcal{G})$.

**Example.** Consider the simple QSG $\mathcal{G}$ in Figure 1, where the budget of Saboteur is $B = 4$, and the cost function is Avg. We claim that whatever the initial configuration, $\mathbf{Val}(\mathcal{G}) = 2$. Indeed, consider the strategy of Saboteur that consists in eventually putting all the budget on the edge $(\boxed{1},\boxed{2})$ (i.e., letting $\delta(\boxed{1},\boxed{2}) = 4$ and $\delta(e) = 0$ for all other edges $e$), and then playing as follows: whenever Runner reaches $\boxed{2}$, move one unit of budget from $(\boxed{1},\boxed{2})$ to $(\boxed{2},\boxed{3})$; if Runner moves to $\boxed{3}$, move the unit of budget from $(\boxed{2},\boxed{3})$ to $(\boxed{3},\boxed{1})$; and when Runner moves back to $\boxed{1}$, move all the budget back on $(\boxed{1},\boxed{2})$, by consuming one unit either from $(\boxed{2},\boxed{3})$ or from $(\boxed{3},\boxed{1})$. Let us call this strategy $\overline{\sigma}$. Since we consider the average cost, only the long-term behaviour of Runner is relevant to compute the cost of a play. So, as soon as Saboteur has managed to reach a distribution $\delta$ such that $\delta(\boxed{1},\boxed{2}) = 4$, the only choices for Runner each time he visits $\boxed{1}$ are either to visit the $\boxed{1}\text{-}\boxed{2}\text{-}\boxed{3}\text{-}\boxed{1}$ cycle, or the $\boxed{1}\text{-}\boxed{2}\text{-}\boxed{1}$ cycle. In the former case, Runner traverses 3 edges and pays $4 + 1 + 1 = 6$, hence an average cost of $\frac{6}{3} = 2$ for this cycle. In the latter, he pays an average of $\frac{4+0}{2} = 2$ for the cycle. Hence, whatever the strategy $\rho$ of Runner, we have $\mathsf{Avg}(\pi_{\overline{\sigma},\rho}) = 2$, which proves that $\underline{\mathbf{Val}}(\mathcal{G}) \geqslant 2$. One can check that the strategy $\overline{\rho}$ of Runner consisting in always playing the $\boxed{1}\text{-}\boxed{2}\text{-}\boxed{3}\text{-}\boxed{1}$ cycle indeed guarantees cost 2, proving that $\overline{\mathbf{Val}}(\mathcal{G}) \leqslant 2$. This proves that the value $\mathbf{Val}(\mathcal{G})$ of the game is 2.

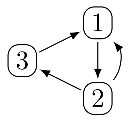


**Figure 1** A QSG.

## 3 Solving quantitative sabotage games

Given a QSG, our main objective is to determine whether Runner can play in such a way that he will ensure a cost at most $T$, no matter how Saboteur plays, and where $T$ is a given threshold. This amounts to determining whether $\mathbf{Val}(\mathcal{G}) \leqslant T$. Thus, for a cost function $f$, the Threshold problem with cost function $f$ consists in determining whether $\mathbf{Val}(\mathcal{G}) \leqslant T$, given a QSG $\mathcal{G}$ with cost function $f$ and a non-negative threshold $T$. When $f = \mathsf{DS}$, we assume that the discount factor $\lambda$ is part of the input. If we want it to be a parameter of the problem (and not a part of the input), we consider $f = \mathsf{DS}_\lambda$. Our main contribution is to characterise the complexity of the threshold problem for all the cost functions introduced before, as summarised in the following theorem:

▶ **Theorem 1.** *For cost functions* Sup, LimSup, Avg, DS *and* $\mathsf{DS}_\lambda$*, the threshold problem over* QSG*s is* EXPTIME*-complete; for* Inf *and* LimInf*, it is in* EXPTIME.

For all cost functions, the EXPTIME membership is established by using the encoding of a QSG $\mathcal{G}$ into a classical quantitative two-player game $[\![\mathcal{G}]\!]$ which is played on a weighted graph, whose vertices are the configurations of the sabotage game, i.e., a tuple containing the current vertex, the last crossed edge and the current weight distribution, and whose weights are in $\{0, \dots, B\}$ (describing how much runner pays by moving from one configuration to another). Notice that $\Delta(\mathcal{G})$ has size at most $(B+1)^{|E|}$, since every distribution is a mapping of $E \to \{0, 1, \dots, B\}$. Hence, we see that the game $[\![\mathcal{G}]\!]$ has a number of vertices at most exponential with respect to $|V|$, and polynomial with respect to $B$ (which, being given in binary, can be exponential in the size of the input of the problem). Using results from [18, 3, 1], we know that we can compute in pseudo-polynomial time the value of the quantitative game $[\![\mathcal{G}]\!]$ for all the cost functions cited in the theorem: here, pseudo-polynomial means polynomial with respect to the number of vertices and edges of $[\![\mathcal{G}]\!]$ (which is exponential with respect to $|V|$), and polynomial with respect to the greatest weight in absolute value, here $B$ (which is also exponential with respect to $|V|$). Thus we obtain the exponential time upper bound

$$\begin{tikzcd}
\mathit{ThPr}_{\mathsf{DS}_\lambda} & \mathit{ThPr}_{\mathsf{Sup}} & \mathit{ThPr}_{\mathsf{LimSup}} & \mathit{ThPr}_{\mathsf{MP}} \\
\mathit{ThPr}_{\mathsf{DS}_\lambda}(0) \arrow[u] & \mathit{ThPr}_{\mathsf{Sup}}(0) \arrow[u] \arrow[l, "\text{Lem. 6}"'] & \mathit{ThPr}_{\mathsf{LimSup}}(0) \arrow[u] \arrow[r, "\text{Lem. 8}"] & \mathit{ThPr}_{\mathsf{MP}}(0) \arrow[u] \\
\text{ABF} \arrow[r, "\text{Lem. 4}"'] & \mathit{ESPr} \arrow[r, "\text{Lem. 5}"'] & \mathit{SPr} \arrow[u, "\text{Lem. 7}"'] \arrow[ul]
\end{tikzcd}$$

**Figure 2** Reductions used in this section. We denote by $\mathit{ThPr}_f$ (respectively, $\mathit{ThPr}_f(0)$) the threshold problem (respectively, the sub-problem of the threshold problem where threshold is 0) for QSGs with cost function $f$. Non-trivial reductions are labelled with the corresponding lemma stated in this section.

announced in the theorem. Note that for $\mathsf{DS}_\lambda$, pseudo-polynomial also means polynomial in the value of the denominator of $\lambda$.[1]

When the budget $B$ is fixed, i.e., when it is a parameter of the problem and not one of the inputs, the explanation above can be adapted to prove that the problem is solvable in polynomial time for all but the $\mathsf{DS}_\lambda$ cost functions. Indeed, we can refine our analysis of the size of $\Delta(\mathcal{G})$. A budget distribution can also be encoded as a mapping $\gamma\colon \{1,\dots,B\} \to E$ where we consider the budget as a set of indexed pebbles: such a mapping represents the distribution $\delta$ defined by $\delta(e) = |\gamma^{-1}(e)|$. This encoding shows that $\Delta(\mathcal{G})$ has size at most $|E|^B$, which is polynomial in $|E|$. For the discounted sum, the role of $\lambda$ in the complexity stays the same, causing an NP ∩ coNP and pseudo-polynomial complexity: this blow-up disappears if $\lambda$ is a parameter of the problem. In the overall, we obtain:

▶ **Corollary 2.** *For cost functions* Inf*,* Sup*,* LimInf*,* LimSup*,* Avg*,* $\mathsf{DS}_\lambda$*, and for fixed budget $B$, the threshold problem for* QSG*s is in* PTIME*; for* DS *(where $\lambda$ is an input), it is in* NP ∩ coNP *and can be solved in pseudo-polynomial time.*

The rest of this section is devoted to the proof of EXPTIME-hardness in Theorem 1 for cost functions Sup, LimSup, Avg and $\mathsf{DS}_\lambda$ (this implies EXPTIME-hardness for DS too). Our gold-standard problem for EXPTIME-hardness is the *alternating Boolean formula* (ABF) problem, introduced by Stockmeyer and Chandra in [15]. Our proof consists of a sequence of reductions from this problem, as depicted in Figure 2. First, we show a reduction to the threshold problem for Sup cost function when the threshold is 0 and **the initial distribution is empty** (i.e., no budget on any edge), on QSGs extended with *safe edges* and *final vertices* (in order to make the reduction more readable). Notice that this problem amounts to determining whether Runner has a strategy to avoid crossing an edge with non-zero budget, therefore we refer to this problem as the *extended safety problem* (*ESPr*). Our next step is to encode safe edges and final vertices into (non-extended) QSGs with gadgets of polynomial size, therefore proving that the *safety problem* (*SPr*) is itself EXPTIME-hard: *SPr* is a special case of the threshold problem $\mathit{ThPr}_{\mathsf{Sup}}(0)$ with Sup cost function and threshold 0, for empty initial distributions. Reductions to threshold problems with other cost functions close our discussion to prove their EXPTIME-hardness.

**Alternating Boolean Formula.** We first recall the alternating Boolean formula problem (ABF) introduced as game $G_6$ in [15], which is the EXPTIME-hard problem from which

[1] In case of discounted-sum, we design $[\![\mathcal{G}]\!]$ with a discount factor $\sqrt{\lambda}$ (not necessarily rational), but we ensure that only one turn over two has a non-zero weight, so that we may indeed apply the reasoning of [18] and their pseudo-polynomial algorithm.

we perform our reductions. Intuitively, an ABF is an (infinite) game played on a Boolean formula whose variables are partitioned into two sets. Each player controls the values of one of the sets of variables. Players take turns changing the value of one of the variables they control. The objective of the first player (Prover) is to eventually make the formula true, while the second player (Disprover) tries to avoid this. We note that this game closely resembles an infinite horizon version of the more classical QBF PROBLEM.

More formally, an ABF instance is given by two finite disjoint sets of Boolean variables, $X$ and $Y$, and a CNF formula over $X \cup Y$. The game is played by two players called Prover and Disprover. They take turns changing the value of at most one of the variables they own ($X$ are the variables of Prover, and $Y$ those of Disprover). Prover wins if and only if the formula is eventually true. A configuration of this game is thus a pair $(\mathrm{val}, \mathrm{Player})$ where val is the current valuation of the variables and Player indicates which player should play next. The ABF PROBLEM consists in, given an ABF game and an initial configuration, determining whether Disprover has a winning strategy from the initial configuration. It is shown EXPTIME-complete in [15].

▶ **Example 3.** Consider the formula $\Phi = Cl_1 \wedge Cl_2 \wedge Cl_3 \wedge Cl_4$ where $Cl_1 = A \vee \neg C$, $Cl_2 = C \vee D$, $Cl_3 = C \vee \neg D$ and $Cl_4 = B \vee \neg B$. Let us further consider the partition of the variables into the sets $X = \{A, B\}$ of Prover, and $Y = \{C, D\}$ of Disprover; and the initial configuration $(\mathrm{val}, \mathrm{Prover})$, where $\mathrm{val} = \{B, C, D\}$ (we denote a valuation by the set of all variables it sets to true). Clearly, in this initial configuration, $\Phi$ is false since $Cl_1$ is false. From that configuration, Prover can either set $A$ to true, or $B$ to false. In the former case, one obtains the configuration $(\{A, B, C, D\}, \mathrm{Disprover})$, where Prover wins, as $\Phi$ now evaluates to true. In the latter case, one obtains the configuration $(\{C, D\}, \mathrm{Disprover})$. We claim that, from this configuration, Prover cannot win the game anymore, i.e., Disprover has a winning strategy that consists in first setting $C$ to false, and in, all subsequent rounds, always flipping the value of $D$, whatever Prover does. Playing according to this strategy ensures Disprover to force visiting only configurations where either $Cl_2$ or $Cl_3$ is false.

**Extended QSG.** To make the encoding of ABF instances into QSG easier, we introduce *extended quantitative sabotage games* (with Sup cost function). Those games are QSG with Sup cost function, a designated subset $F \subseteq V$ of *final vertices* and a designated subset $S \subseteq E$ of *safe edges* (those special vertices and edges are henceforth depicted with double lines). $F$ and $S$ influence the semantics of the game: Saboteur can place some budget on final vertices (which is accounted for in the cost when Runner visits those vertices), but cannot put budget on safe edges; and the game stops as soon as Runner visits a final vertex. We consider the *extended safety problem* (*ESPr*), which is to determine whether an extended QSG $\mathcal{G}$ *with empty initial distribution* has value $\mathbf{Val}(\mathcal{G}) \leqslant 0$.

Since the cost function is Sup, this amounts to checking that Runner has a strategy to reach a final vertex, with no budget assigned to it, without crossing any edge with non-null budget. From now on, we assume $B < |E|$, as the problem is trivial otherwise. Then:

▶ **Lemma 4.** *The ABF problem is polynomial-time reducible to ESPr.*

**Proof Sketch.** We consider an instance of the ABF problem given by Boolean variable sets $X$ and $Y$ (owned by Prover and Disprover, respectively) and a CNF formula $\Phi$ over $X \cup Y$. We construct an extended QSG $\mathcal{E}$ such that Saboteur wins in $\mathcal{E}$ if and only if Prover wins in the ABF problem. Valuations of the variables in $X \cup Y$ are encoded by budget distributions in $\mathcal{E}$. For each variable $x \in X \cup Y$, $\mathcal{E}$ has 4 *final* vertices associated with $x$,

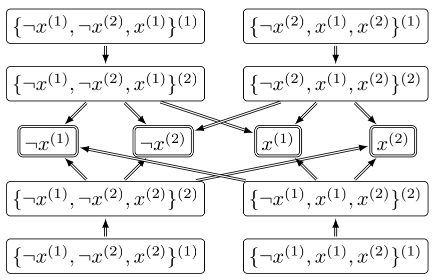

$\{\neg x^{(1)}, \neg x^{(2)}, x^{(1)}\}^{(1)}$
$\{\neg x^{(2)}, x^{(1)}, x^{(2)}\}^{(1)}$
$\{\neg x^{(1)}, \neg x^{(2)}, x^{(1)}\}^{(2)}$
$\{\neg x^{(2)}, x^{(1)}, x^{(2)}\}^{(2)}$
$\neg x^{(1)}$
$\neg x^{(2)}$
$x^{(1)}$
$x^{(2)}$
$\{\neg x^{(1)}, \neg x^{(2)}, x^{(2)}\}^{(2)}$
$\{\neg x^{(1)}, x^{(1)}, x^{(2)}\}^{(2)}$
$\{\neg x^{(1)}, \neg x^{(2)}, x^{(2)}\}^{(1)}$
$\{\neg x^{(1)}, x^{(1)}, x^{(2)}\}^{(1)}$

**Figure 3** Verifying condition (i).

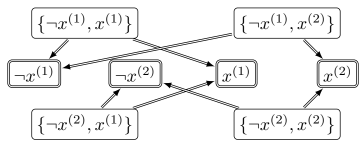

$\{\neg x^{(1)}, x^{(1)}\}$
$\{\neg x^{(1)}, x^{(2)}\}$
$\neg x^{(1)}$
$\neg x^{(2)}$
$x^{(1)}$
$x^{(2)}$
$\{\neg x^{(2)}, x^{(1)}\}$
$\{\neg x^{(2)}, x^{(2)}\}$

**Figure 4** Verifying condition (ii).

$Ver(x) = \{\neg x^{(1)}, \neg x^{(2)}, x^{(1)}, x^{(2)}\}$. A budget distribution $\delta$ encodes a valuation in which variable $x \in X \cup Y$ is **true** if and only if $\delta(x^{(1)}) = \delta(x^{(2)}) = 1$ and $\delta(\neg x^{(1)}) = \delta(\neg x^{(2)}) = 0$.

Then, $\mathcal{E}$ simulates the ABF game as follows. The duty of Saboteur is to move the budget distribution in such a way that he respects the encoding of the valuations explained above. To enforce this, we rely on the two gadgets, depicted in Figure 3 and 4. They allow Runner to check that Saboteur respects the encoding and let him lose if he does not. More precisely, the gadget in Figure 3 allows one to check that $(i)$ there is a non-zero budget on at least two vertices from $Ver(x)$; and the one in Figure 4 that $(ii)$ there is a non-zero budget on exactly $\{\neg x^{(1)}, \neg x^{(2)}\}$ or $\{x^{(1)}, x^{(2)}\}$. To allow Runner to check one of these conditions, we allow him to move to one of the four corner vertices of the corresponding gadget, from where one can easily check Runner can win if and only if the condition is not respected. In our reduction, Runner will be allowed to check condition $(i)$, for all variables, from all vertices but will be able to check $(ii)$ only on some of them, as we will see later.

The remaining of the construction is done in a way to allow Saboteur and Runner to choose valid re-configurations of $Ver(x)$ for all variables $x$, and make sure that if a player cheats, it allows the other player to win the safety game. If at some point, the formula $\Phi$ becomes true, then we allow Saboteur to enter a final gadget which verifies that the current budget distribution to $Ver(X) = \bigcup_{x \in X \cup Y} Ver(x)$ satisfies $\Phi$. This last gadget lets Runner choose a clause and then allows Saboteur to choose a literal, within this clause, which should be true. It is easy to see that the choice of clause $Cl$ can be done by way of safe edges. The choice of literal, done by Saboteur, consists in choosing a suffix of $Cl$ for which the left-most literal holds. Figure 5 shows the *ESPr* which results from applying our construction to the ABF formula from Example 3. ◀

We now explain how to encode safe edges and final vertices into usual QSGs, therefore showing the EXPTIME-hardness of the safety problem for QSGs.

▶ **Lemma 5.** *The extended safety problem ESPr is polynomial-time reducible to a safety problem SPr with budget* 2.

**Proof Sketch.** Each final vertex $v$ in an extended QSG $\mathcal{E}$ is replaced by the gadget in Figure 6a, where $\{\alpha_i \mid 1 \leqslant i \leqslant B+1\}$ is a clique of size $B+1$, hence bigger than the budget of Saboteur. To encode $\delta(v) = 1$ in $\mathcal{E}$, Saboteur now puts one unit of budget on $(\boxed{A}, \boxed{C_1})$. If Runner reaches the gadget (through $\boxed{A}$), Saboteur puts one unit of budget on $(\boxed{A}, \boxed{C_2})$. Clearly, Runner loses if and only if there was already one unit on $(\boxed{A}, \boxed{C_1})$ (i.e., $v$ was marked in $\mathcal{E}$). Each safe edge $(\boxed{A}, \boxed{C})$ is replaced by the gadget in Figure 6b. Here, we make use of final vertices and disjoint paths so that Saboteur cannot block all paths from $\boxed{A}$ to $\boxed{C}$ without letting Runner win by visiting a final vertex with zero budget. Both gadgets have polynomial size since we assume that $B < |E|$. ◀

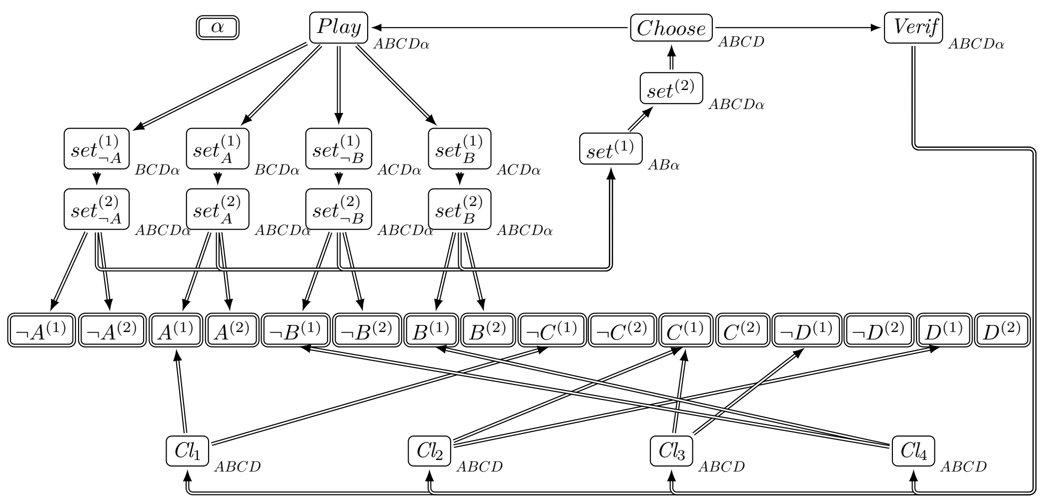


**Figure 5** Excerpt of the *ESPr* constructed from the ABF of Example 3. In addition to these nodes and edges, the full *ESPr* contains: an initialisation gadget; a *safe* edge from a node $n$ to all four corner nodes of gadget $(i)$ in Figure 3 iff $n$ is labeled by $\alpha$; and a *safe* edge from a node $n$ to all four corner nodes of gadget $(ii)$ in Figure 4 testing variable $x \in \{A, B, C, D\}$ iff $n$ is labeled by $x$. These parts have been omitted for the sake of clarity.

As the safety problem is a specific case of the threshold problem for Sup QSGs (where the initial distribution is empty, and threshold is fixed to 0), it follows that $\mathit{ThPr}_{\mathsf{Sup}}(0)$ and $\mathit{ThPr}_{\mathsf{Sup}}$ are EXPTIME-hard too.

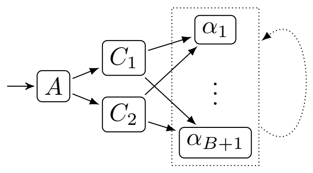
We note that given a QSG $\mathcal{G}$, for all plays $\pi$ in $\mathcal{G}$, for all $0 < \lambda < 1$, and for all $\delta \in \Delta(\mathcal{G})$, $\mathsf{Sup}(\overline{\pi}) = 0$ if and only if $\mathsf{DS}_\lambda(\overline{\pi}) = 0$. This implies the following result, showing that $\mathit{ThPr}_{\mathsf{DS}_\lambda}(0)$ and $\mathit{ThPr}_{\mathsf{DS}_\lambda}$ are also EXPTIME-hard.

▶ **Lemma 6.** *For any $\lambda \in (0,1)$, the threshold problem for* $\mathsf{DS}_\lambda$ *and threshold* 0 *is equivalent to the threshold problem for* Sup *and threshold* 0*.*

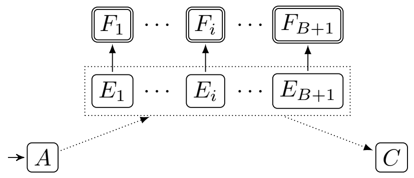
Let us now focus on LimSup. To show that $\mathit{ThPr}_{\mathsf{LimSup}}$ is EXPTIME-hard, we describe a reduction from *SPr* to $\mathit{ThPr}_{\mathsf{LimSup}}(0)$ as stated in the following lemma.

▶ **Lemma 7.** *The safety problem SPr is polynomial-time reducible to the threshold problem for* LimSup *and threshold* 0*.*

**Proof Sketch.** Let $\mathcal{I} = (V, E, B, v_I, \delta_I, \mathsf{Sup})$ be an instance of *SPr* (with $G(\mathcal{I})$ its underlying graph $(V, E)$). We build a QSG $\mathcal{G}$ with cost function LimSup such that $\mathbf{Val}(\mathcal{G}) = 0$ if and only if Runner wins in $\mathcal{I}$. The idea of the construction is that a play of $\mathcal{G}$ consists in simulating a potentially infinite sequence of plays of $\mathcal{I}$, using appropriate gadgets to 'reset' the safety game between two successive simulations. Then, repeatedly playing a winning strategy for $\mathcal{I}$ allows Runner to ensure a LimSup of 0 in $\mathcal{G}$; and one can extract a winning strategy for the safety game $\mathcal{I}$ from any strategy ensuring a LimSup of 0 in $\mathcal{G}$. The QSG $\mathcal{G}$ has budget $B' = |E|$ and is obtained by extending $G(\mathcal{I})$ with two gadgets. Note that we are giving Saboteur more budget than he had in $\mathcal{I}$. However, as we will see in the sequel, at the beginning of every faithful simulation of $\mathcal{I}$ (i.e. when Runner moves to $G(\mathcal{I})$) there will be $B' - B$ of it in the second gadget and $B$ in the first and during any faithful simulation of $\mathcal{I}$ only budget from the initial gadget is redistribtued into $G(\mathcal{I})$.

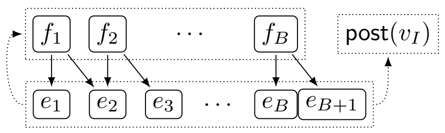


**(a)** A gadget for final vertices.



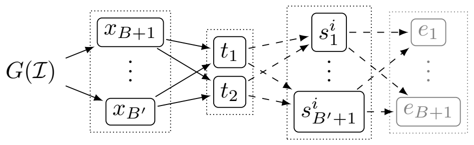
$F_1$ $\cdots$ $F_i$ $\cdots$ $F_{B+1}$
$E_1$ $\cdots$ $E_i$ $\cdots$ $E_{B+1}$
$A$
$C$

**(b)** A gadget for safe edges.



$f_1$ $f_2$ $\cdots$ $f_B$
post$(v_I)$
$e_1$ $e_2$ $e_3$ $\cdots$ $e_B$ $e_{B+1}$

**(c)** Initial gadget for Sup to LimSup reduction.



**(d)** Exit gadget for Sup to LimSup reduction. Dashed arrows represent a (safe) path traversing $B'$ sets $s^i$ of vertices.

**Figure 6** Dotted arrows represent edges from all sources to all targets.

The first gadget is an initial gadget which is visited every time the safety game is 'reset'. It allows Runner to stay safe from any weighted edges (and avoid reaching $G(\mathcal{I})$) until Saboteur has placed $B$ units of budget on it (and thus removed them from the $G(\mathcal{I})$. It is depicted in Figure 6c, where all $e_i$ are intuitively copies of $v_I$, and post$(v_I)$ corresponds to the set of all successors of $v_I$ in $G(\mathcal{I})$.

The second gadget allows Runner to leave $G(\mathcal{I})$ if Saboteur ever places more than $B$ units of budget on $G(\mathcal{I})$ (and thus removes this budget from the gadgets), thereby triggering a 'reset' of the simulation. This gadget, depicted in Figure 6d, also allows Runner to come back to the initial gadget visiting only edges with zero budget. The figure shows a sequence of safe transitions (i.e. several vertices with high out-degree) which leads back to the copies $e_i$ of the initial vertex. Further, this 'safe path' takes long enough for Saboteur to redistribute the budget from $G(\mathcal{I})$ to both gadgets. In order for Saboteur to stop Runner from always taking this 'safe exit' from $G(\mathcal{I})$ he can place $B' - B$ budget in specific edges of this second gadget. More specifically, he can place a unit of budget on one outgoing edge from each $x_j$, for $B + 1 \leqslant j \leqslant B'$, before forcing Runner to enter $G(\mathcal{I})$.

**Intuition behind the global construction.** Assume that Saboteur has a winning strategy in $\mathcal{I}$. Then, when Runner is in the initial gadget, Saboteur will play as expected and remove all weights from $G(\mathcal{I})$. Critically, the weights he removes from $G(\mathcal{I})$ will go to specific edges in both gadgets described above. Runner is now forced to play into $G(\mathcal{I})$, and Saboteur can follow his winning strategy to hit Runner at some point without using more than $B$ weights. If Runner attempts to bail out of $G$ through the alternative exit, and to head back to the initial gadget, then we make sure he is also hit by Saboteur. Clearly, this ensures that the LimSup value of the game is strictly greater than 0. Now assume that Runner has a winning strategy in $\mathcal{I}$. In this case, if Saboteur does not remove all weights from $G(\mathcal{I})$, then Runner is allowed to stay in the initial gadget forever or jump to $G(\mathcal{I})$ and immediately bail out using the exit gadget. In both cases he avoids getting hit by Saboteur. Let us assume Saboteur plays as expected and thus Runner enters $G(\mathcal{I})$ eventually. In this case, Runner can play his winning strategy, hence avoiding edges with non-zero budget (with Saboteur using budget $B$). Either he dodges weighted edges forever, or Saboteur cheats and uses some of his additional budget. However, in this case he creates an exit for Runner back to the initial gadget, and the same analysis as above applies. This implies that the value of the game is exactly 0. ◀

Proving the EXPTIME-hardness result for cost function Avg is done by noticing that, for threshold 0, both problems are equivalent.

▶ **Lemma 8.** *The threshold problem for* LimSup *and threshold* $0$ *is polynomial-time reducible to the threshold problem for* Avg *and threshold* $0$*.*

## 4 Static quantitative sabotage games

In light of the EXPTIME-completeness of QSGs, we study in this section a restriction of the problem, that might be sufficient to model some interesting cases. The restriction concerns the dynamics of the behaviour of Saboteur. In a *static* QSG, Saboteur chooses at the beginning a budget distribution (hence, changing the initial budget distribution), and then commits to this distribution during the whole game. The situation is no longer a reactive two-player game, but rather we ask whether for every possible initial (and static) budget distribution, Runner has a nicely behaved strategy.

Formally, for a QSG $\mathcal{G} = (V, E, B, v_I, f)$ (we remove the initial budget distribution from the tuple in this section, since it is useless) and a budget distribution $\delta \in \Delta(\mathcal{G})$, we denote by $\mathcal{G}_\delta$ the QSG obtained from $\mathcal{G}$ by taking $\delta$ as initial budget distribution. Furthermore, we define the *identity strategy* $\iota$ of Saboteur in $\mathcal{G}$, as the strategy mapping every prefix $\pi \in \mathrm{Prefs}_{\mathrm{Sab}}(\mathcal{G})$ to the last budget distribution appearing in prefix $\pi$. We let $\mathbf{Val}_{\mathrm{stat}}(\mathcal{G}) = \sup_{\delta\in\Delta(\mathcal{G})} \inf_{\rho\in\Sigma_{\mathrm{Run}}(\mathcal{G})} f(\pi^{\delta}_{\rho,\iota})$, where $\pi^{\delta}_{\rho,\iota}$ denotes the unique play defined by the profile $(\rho, \iota)$ in QSG $\mathcal{G}_\delta$. Notice that this value is equal to $\inf_{\rho\in\Sigma_{\mathrm{Run}}(\mathcal{G})} \sup_{\delta\in\Delta(\mathcal{G})} f(\pi^{\delta}_{\rho,\iota})$, since in $\mathcal{G}$, when Saboteur follows strategy $\iota$, the quantitative game $[\![\mathcal{G}]\!]$ is split into independent games, one for each initial distribution $\delta$, that Runner knows as soon as it starts playing. The Static Threshold problem with cost function $f$ consists in, given as input a QSG $\mathcal{G}$ with cost function $f$ and a non-negative threshold $T$, determining whether the inequality $\mathbf{Val}_{\mathrm{stat}}(\mathcal{G}) \leqslant T$ holds. We now state the complexity of this new problem.

▶ **Theorem 9.** *For cost functions* Inf *and* LimInf*, the static threshold problem over* QSG*s is in* PTIME*; for* Sup*,* LimSup*,* Avg*, and* DS*, it is* coNP*-complete.*

First, we give the intuition behind our polynomial-time algorithm to decide the static threshold problem for cost functions Inf and LimInf.

▶ **Lemma 10.** *For cost functions* Inf *and* LimInf*, the static threshold problem over* QSG*s is in* PTIME*.*

**Proof Sketch.** For Inf, we claim that $\mathbf{Val}_{\mathrm{stat}}(\mathcal{G}) = \lfloor|\overline{E}|/B\rfloor$, where $\overline{E}$ is the set of edges reachable from $v_I$. Indeed once a distribution $\delta$ is chosen, any optimal strategy of Runner will make him reach an edge of $\overline{E}$ that has the minimum weight, thus Saboteur must distribute evenly its budget over $\overline{E}$. A similar argument works for LimInf, showing that $\mathbf{Val}_{\mathrm{stat}}(\mathcal{G}) = \lfloor|\widetilde{E}|/B\rfloor$, where $\widetilde{E}$ is the set of edges reachable from $v_I$ and contained in a strongly connected component. ◀

Then, let us turn to the coNP-completeness of the problem for cost functions Sup, LimSup, Avg, and DS. Notice that, because of the two possible definitions of $\mathbf{Val}_{\mathrm{stat}}(\mathcal{G})$ explained in the beginning of the section, the complement of the static threshold problem asks whether there exists a budget distribution $\delta$ such that $f(\pi^{\delta}_{\rho,\iota}) > T$ for every strategy $\rho \in \Sigma_{\mathrm{Run}}(\mathcal{G})$

of Runner. Thus we show the NP-completeness of the complement of the static threshold problems for the four cost functions.

▶ **Lemma 11.** *For cost functions* Sup*,* LimSup*,* Avg*, and* DS*, the complement of the static threshold problem over* QSG*s is* NP*-complete.*

**Proof Sketch.** For the membership in NP, we can first guess a budget distribution $\delta$ (that is of size polynomial), and then compute the value of the one-player (since player Max has no choices anymore) quantitative game $\mathcal{G}_\delta$, to check if it is greater than $T$: computing the value of such a game can be done in polynomial time for the four cost functions we consider (see [1]).

For the NP-hardness with cost functions LimSup and Avg, we give a reduction from the following problem. The FEEDBACK ARC SET PROBLEM asks, given a directed graph $G = (V, E)$ and a threshold $k \leqslant |E|$, whether there is a set $E'$ of at most $k$ edges of $G$ such that $(V, E \setminus E')$ is acyclic. Karp showed [9] that the feedback arc set problem is NP-complete. Let us consider an instance of the feedback arc set problem, given by a directed graph $G = (V, E)$ and a natural integer $k \leqslant |E|$. Wlog, we can add to the graph a vertex $v_I$, with null in-degree, and, for all vertices $v \neq v_I$, an edge $(v_I, v)$. Observe that this does not change the output of the feedback arc set problem as $v_I$ is not included in any cycle. We then construct a QSG $\mathcal{G} = (V, E, k, v_I, f)$ with $f \in \{\mathsf{LimSup}, \mathsf{Avg}\}$. It is not difficult to show that $\mathbf{Val}_{\mathrm{stat}}(\mathcal{G}) > 0$ if and only if there exists a set $E'$ of $k$ edges of $G$ such that $(V, E \setminus E')$ is acyclic. The result for Sup and DS is then obtained by a slight modification of the previous proof. In particular, we make use of Lemma 6, once more. ◀

## 5 Reactive systems under failure

One can see a sabotage game as a system in which a controller tries to evolve while avoiding as much as possible the failures caused by the environment. The vertices of the graph represent configurations of the system, edges represent the actions, and the budget of the Saboteur may represent a finite amount of failures that can simultaneously occur during the execution. In a quantitative reasoning, a failure may be better represented by a quantity describing how much some elements of the system are overloaded, and then how much it would cost, in terms of time or energy, to use them.

Following this main motivation, we propose to look at sabotage games as a particular semantics of controllable systems. Indeed, while a standard semantics would analyse the feasibility of a requirement in a fully functional system, a *sabotage semantics* allows one to analyse systems subject to errors, and to decide, e.g., whether one can satisfy a Boolean constraint while minimising the average number of failures encountered during the execution. In particular, sabotage games, as introduced in this work, would correspond to the sabotage semantics of a system where the controller must walk in a graph with no particular objective, other than minimising the failures.

From a modelling point of view, graphs—which can be viewed as one-player games with trivial winning conditions—are quite limited. In more realistic models, we may be interested in modelling systems with uncontrollable actions (i.e., two-player games), and where the controller has a specific Boolean goal to achieve, instead of simply staying in the graph *ad vitam æternam*. A more realistic goal is usually expressed via a parity condition or LTL formulas. When a reactive system is modelled by a two-player parity game, deciding whether one can ensure the parity condition, while maintaining a cost associated with the sabotage semantics below a given threshold, can be shown to be not harder than solving sabotage games. That is, the problem is EXPTIME-complete. This result is obtained by a reduction to quantitative parity games [4] (see [2] for a formal proof). When the requirement is expressed with an LTL formula instead of a parity condition, the problem becomes 2-EXPTIME-complete, due to an additional exponential blow-up in the size of the input formula. Note, however, that the LTL-reactive synthesis problem itself (with the standard non-sabotage semantics) is already 2-EXPTIME-complete. In this case, the sabotage semantics does not add to the complexity of the problem, which further shows that our present contributions might have practical applications, albeit the high complexity.

## 6 Conclusion

We have conducted a study of systems subject to failure, using the model of *quantitative sabotage games*. We have shown that under *dynamic sabotage*, the threshold problem is EXPTIME-complete for most objective functions, and coNP-complete under *static sabotage*, for the same functions (see table 1 for a summary of these results). We have also shown the applicability of our framework to deal with the more general problem of reactive synthesis in systems under failures. The QSGs we have introduced open many questions related to evolving structures. Here we have studied the worst-case scenario, i.e., where the environment is modelled by an antagonistic adversary, but, as considered in [10] for reachability Boolean objectives, one could also look at a probabilistic model, where failures, i.e., redistributions of weights, are random variables. Another natural extension of this work would be to consider a more realistic setting where the controller (Runner) has partial information regarding the weights of Saboteur.

Although the synthesis problem has been widely studied in theory, there are not many tools which implement the known theoretical solutions to decide it. The is is particularly true for quantitative objectives. Recently, however, competitions have been organised to encourage the development of such tools and the standardisation of an input format (see, e.g., SYNTCOMP and SyGuS).[2] Motivated by the similarities between the ABF problem (solving a safety game described by a logical formula) and the synthesis problem as solved in those competition (solving a safety game described by a logical circuit), one of our future projects is to show that quantitative extensions of some of the practical tools implemented for the reactive synthesis problem could be used to solve sabotage games.

## References

1 Krzysztof R. Apt and Erich Grädel. *Lectures in game theory for computer scientists*. Cambridge University Press, 2011.

2 Thomas Brihaye, Gilles Geeraerts, Axel Haddad, Benjamin Monmege, Guillermo A. Pérez, and Gabriel Renault. Quantitative games under failures. Research Report 1504.06744, arXiv, April 2015.

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

4 Krishnendu Chatterjee, Thomas A. Henzinger, and Marcin Jurdzinski. Mean-payoff parity games. In *LICS*, pages 178–187. IEEE, 2005.

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

[2] Links to both competitions' websites: `http://www.syntcomp.org` and `http://www.sygus.org/`.

6 Arthur S. Goldstein and Edward M. Reingold. The complexity of pursuit on a graph. *Theor. Comput. Sci.*, 143(1):93 – 112, 1995.

7 Sten Grüner, Frank G. Radmacher, and Wolfgang Thomas. Connectivity games over dynamic networks. *Theor. Comput. Sci.*, 493:46–65, 2013.

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

9 Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a Symposium on the Complexity of Computer Computations*, pages 85–103, 1972.

10 Dominik Klein, Frank G. Radmacher, and Wolfgang Thomas. Moving in a network under random failures: A complexity analysis. *Science of Comp. Prog.*, 77(7-8):940–954, 2012.

11 Lena Maria Kurzen. *Complexity in interaction.* PhD thesis, Institute for Logic, Language and Computation, 2011.

12 Christof Löding and Philipp Rohde. Solving the sabotage game is PSPACE-hard. In *MFCS*, volume 2747 of *LNCS*, pages 531–540. Springer, 2003.

13 Donald A Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975.

14 Merlijn Sevenster. *Branches of imperfect information: logic, games, and computation.* PhD thesis, Institute for Logic, Language and Computation, 2006.

15 Larry J. Stockmeyer and Ashok K. Chandra. Provably difficult combinatorial games. *SIAM J. Comput.*, 8(2):151–174, 1979.

16 Johan van Benthem. An essay on sabotage and obstruction. In *Mechanizing Mathematical Reasoning*, volume 2605 of *LNAI*, pages 268–276. Springer, 2005.

17 Masafumi Yamashita and Ichiro Suzuki. Characterizing geometric patterns formable by oblivious anonymous mobile robots. *Theor. Comput. Sci.*, 411(26-28):2433–2453, 2010.

18 Uri Zwick and Michael S. Paterson. The complexity of mean payoff games. *Theor. Comput. Sci.*, 158:343–359, 1996.