# Quantitative Games under Failures

Thomas Brihaye[1], Gilles Geeraerts[2], Axel Haddad[1], Benjamin Monmege[2], 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,benjamin.monmege,gperezme}@ulb.ac.be

**Abstract.** We study a generalisation of sabotage games, a model of dynamic network games introduced by van Benthem [17]. 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.

## 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 [5,8,2] 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 [17] – 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 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 [11], 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

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 |

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 [18] 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.

*Related Works & Contributions.* 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.[3] 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 1). 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 5, we comment on several implications of the complexity bounds proved in this work.

## 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. The six classical cost functions that we consider are Inf, Sup, LimInf, LimSup, Avg and $\mathsf{DS}_\lambda$ (with $0 < \lambda < 1$): Avg stands for the average cost (also called *mean-payoff* in the literature) and maps every sequence $u = (u_i)_{i \geqslant 0} \in \mathbb{Q}^\omega$ to $\mathsf{Avg}(u) = \liminf_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n} u_i$, whereas $\mathsf{DS}_\lambda$ stands for discounted-sum, defined by $\mathsf{DS}_\lambda(\pi) = \sum_{i=0}^{\infty} \lambda^i \cdot u_i$ and the other functions have their usual meaning. In the following, we let $\mathsf{DS} = \{\mathsf{DS}_\lambda \mid 0 < \lambda < 1\}$.

---

[3] We compare to related works on cops and robbers games in Appendix A.

*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 QSG with set of edges $E$ and budget $B$.

*Semantics.* To define the semantics of a 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 QSG $\mathcal{G} = (V, E, B, v_I, \delta_I, f)$ is an infinite sequence $\pi = v_0 \delta_0 v_1 \delta_1 \cdots$ 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 $f(v_0 \delta_0 v_1 \delta_1 \cdots) = f(\delta_0(v_0, v_1) \delta_1(v_1, v_2) \cdots)$. 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 $\pi_{\rho,\sigma} = v_0 \delta_0 v_1 \delta_1 \cdots$ such that for all $i \geqslant 0$: $(i)$ $v_{i+1} = \rho(v_0 \delta_0 \cdots v_i \delta_i)$; and $(ii)$ $\delta_{i+1} = \sigma(v_0 \delta_0 \cdots v_i \delta_i v_{i+1})$.

*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 QSG $\mathcal{G}$: the superior value (modelling the best value for Runner) as $\overline{\mathbf{Val}}(\mathcal{G}) := \inf_{\rho \in \Sigma_{\mathrm{Run}}(\mathcal{G})} \sup_{\sigma \in \Sigma_{\mathrm{Sab}}(\mathcal{G})} f(\pi_{\rho,\sigma})$, 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 QSGs are determined, i.e., that $\underline{\mathbf{Val}}(\mathcal{G}) = \overline{\mathbf{Val}}(\mathcal{G})$ for all QSGs $\mathcal{G}$. This can be formally proved by encoding a 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 [12], as formally done in Appendix B. $\underline{\mathbf{Val}}(\mathcal{G}) = \overline{\mathbf{Val}}(\mathcal{G})$ is henceforth called the *value of $\mathcal{G}$*, and denoted by $\mathbf{Val}(\mathcal{G})$.

*Example.* Consider the simple QSG $\mathcal{G}$ in Fig. 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 $(\textcircled{1}, \textcircled{2})$ (i.e., letting $\delta(\textcircled{1}, \textcircled{2}) = 4$ and $\delta(e) = 0$ for all other edges $e$), and then playing as follows: whenever Runner reaches $\textcircled{2}$, move one unit of budget from $(\textcircled{1}, \textcircled{2})$ to $(\textcircled{2}, \textcircled{3})$; if Runner moves to $\textcircled{3}$, move the unit of budget from $(\textcircled{2}, \textcircled{3})$ to $(\textcircled{3}, \textcircled{1})$; and when Runner moves back to $\textcircled{1}$, move all the



Fig. 1: A QSG

budget back on $(\textcircled{1}, \textcircled{2})$, by consuming one unit either from $(\textcircled{2}, \textcircled{3})$ or from $(\textcircled{3}, \textcircled{1})$. 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 $\delta(\textcircled{1}, \textcircled{2}) = 4$, the only choices for Runner each time he visits $\textcircled{1}$ are either to visit the $\textcircled{1}$–$\textcircled{2}$–$\textcircled{3}$–$\textcircled{1}$ cycle, or the $\textcircled{1}$–$\textcircled{2}$–$\textcircled{1}$ 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
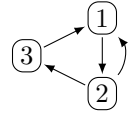
3

consisting in always playing the (1)–(2)–(3)–(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.

## 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 (explained in Appendix B) 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 $\{0, \ldots, B\}$ (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 $E \to \{0, 1, \ldots, B\}$. 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 [20,2,1], 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 announced in the theorem. Note that for $\mathsf{DS}_\lambda$, pseudo-polynomial also means polynomial in the value of the denominator of $\lambda$.[4]

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 $\gamma \colon \{1, \ldots, B\} \to E$ 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 $\mathsf{NP} \cap \mathsf{coNP}$ and pseudo-polynomial complexity: this blow-up disappears if $\lambda$ is a parameter of the problem. In the overall, we obtain:

**Corollary 1.** *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* $\mathsf{NP} \cap \mathsf{coNP}$ *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 [16]. Our proof consists of a sequence of reductions from this problem, as depicted in Fig. 2. First, we show a reduction to the threshold problem for Sup cost function when the threshold is 0 and **the initial distribution**

---

[4] 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 [20] and their pseudo-polynomial algorithm.
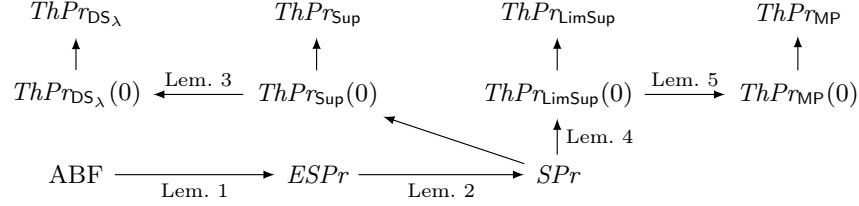
Fig. 2: Reductions used in this section. We denote by $ThPr_f$ (respectively, $ThPr_f(0)$) the threshold problem (respectively, the sub-problem of the threshold problem where threshold is 0) for QSGs with cost function $f$. Non-trivial reductions are labelled with the corresponding lemma stated in this section.

**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 $ThPr_{\mathsf{Sup}}(0)$ 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 [16]), which is the EXPTIME-hard problem from which we perform our reductions. 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 (val, 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 [16].

*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 an *unmarked* final vertex without crossing any edge with non-null budget. From now on, we assume $B < |E|$, as the problem is trivial otherwise. Then:

**Lemma 1.** *ESPr is at least as hard as the ABF problem.*

*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 $x$, $Ver(x) = \{\neg x^{(1)}, \neg x^{(2)}, x^{(1)}, x^{(2)}\}$. 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
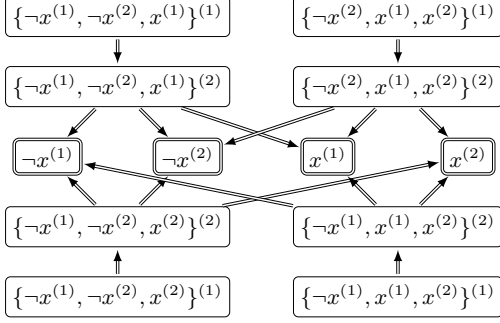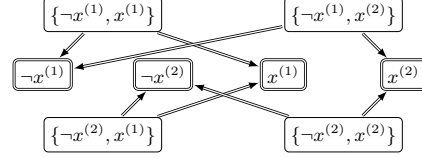
Fig. 3: Verifying condition $(i)$
Fig. 4: Verifying condition $(ii)$

two gadgets, depicted in Fig. 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 Fig. 3 allows one to check that there is a non-zero budget on at least two vertices from $Ver(x)$; and the one in Fig. 3 that 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.

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 is true. It is easy to see that the choice of clause $\chi$ can be done by way of safe edges. The choice of literal, done by Saboteur, consists in choosing a suffix of $\chi$ for which the left-most literal holds. We refer the reader to Appendix C.1 for the full reduction (as well as an example), in particular how we can force, before the beginning of the actual game, to start in the initial valuation of the ABF game. □

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 2.** *When the budget is at least 2, the safety problem SPr is at least as hard as the extended safety problem ESPr.*

*Proof (Sketch).* Each final vertex $v$ in an extended QSG $\mathcal{E}$ is replaced by the gadget in Fig. 5a, 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 Fig. 5b. 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 $B < |E|$. □

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 $ThPr_{\mathsf{Sup}}(0)$ and $ThPr_{\mathsf{Sup}}$ 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 $ThPr_{\mathsf{DS}_\lambda}(0)$ and $ThPr_{\mathsf{DS}_\lambda}$ are also EXPTIME-hard.

(a) A gadget for final vertices



(b) A gadget for safe edges



(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.
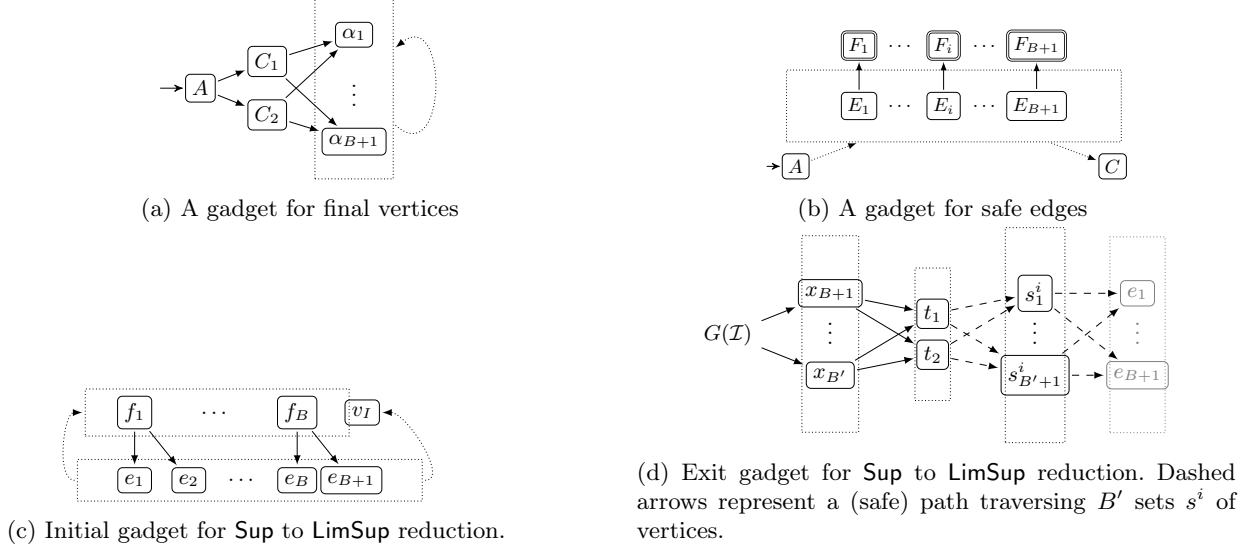
Fig. 5: Dotted arrows represent edges from all sources to all targets.

**Lemma 3.** *For any $\lambda \in (0, 1)$, the threshold problem for $\mathsf{DS}_\lambda$ and threshold $0$ is equivalent to the threshold problem for $\mathsf{Sup}$ 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 4.** *The threshold problem for $\mathsf{LimSup}$ and threshold 0 is at least as hard as the safety problem $SPr$.*

*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. The former is an initial gadget that ensures Runner does not reach $G(\mathcal{I})$ until Saboteur has placed $B$ units of budget on $G(\mathcal{I})$ (and thus removed them from the gadgets). It is depicted in Fig. 5c. The latter 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 Fig. 5d, also makes sure Runner comes back to the initial gadget visiting only edges with zero budget. Further, this 'safe path' takes long enough for Saboteur to redistribute the budget from $G(\mathcal{I})$ to both gadgets. □

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

**Lemma 5.** *The threshold problem for $\mathsf{Avg}$ and threshold 0 is at least as hard as the threshold problem for $\mathsf{LimSup}$ 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 $\mathbf{Val}_{\mathrm{stat}}(\mathcal{G}) = \sup_{\delta \in \Delta(\mathcal{G})} \inf_{\rho \in \Sigma_{\mathrm{Run}}(\mathcal{G})} f(\pi_{\rho,\iota}^\delta)$, where $\pi_{\rho,\iota}^\delta$ denotes the unique play defined by the profile $(\rho, \iota)$ in QSG $\mathcal{G}_\delta$. Notice that this value is equal to $\inf_{\rho \in \Sigma_{\mathrm{Run}}(\mathcal{G})} \sup_{\delta \in \Delta(\mathcal{G})} f(\pi_{\rho,\iota}^\delta)$, since in $\mathcal{G}$, when Saboteur follows strategy $\iota$, the quantitative game $[\![\mathcal{G}]\!]$ (see Appendix B) 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 2.** *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 6.** *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. $\square$

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 $f(\pi_{\rho,\iota}^\delta) > T$ 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 7.** *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 3, once more. We refer the reader to Appendix D.2 for the details. $\square$

# 5  Conclusion

In this work, we have allowed Runner to traverse the given graph while trying to minimise the cost, computed from the sequence of budget units crossed during the play. As an extension, we could also consider QSGs in which Runner is given some specific task specified, e.g., by a parity objective over the given graph. It is also possible to add uncontrollable vertices to the graph, in which the movement of the token is determined by the environment. The game then becomes multi-objective, since Runner wants to satisfy a given Boolean objective while minimising a cost function. It is not hard to show that deciding whether Runner wins such games is also EXPTIME-complete. If the parity condition is given succinctly in the form of an LTL formula then the corresponding synthesis problems, even for dynamic arenas, is still 2-EXPTIME-complete.[5]

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.

# References

1. K. R. Apt and E. Grädel. *Lectures in game theory for computer scientists*. Cambridge University Press, 2011.
2. K. Chatterjee, L. Doyen, and T. A. Henzinger. Quantitative languages. *ACM Trans. Comput. Log.*, 11(4), 2010.
3. K. Chatterjee, T. A. Henzinger, and M. Jurdzinski. Mean-payoff parity games. In *LICS*, pages 178–187. IEEE, 2005.
4. K. Chatterjee, T. A. Henzinger, and N. Piterman. Generalized parity games. In *FoSSaCS*, pages 153–167. Springer, 2007.
5. A. Ehrenfeucht and J. Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8:109–113, 1979.
6. A. S. Goldstein and E. M. Reingold. The complexity of pursuit on a graph. *Theor. Comput. Sci.*, 143(1):93 – 112, 1995.
7. S. Grüner, F. G. Radmacher, and W. Thomas. Connectivity games over dynamic networks. *Theor. Comput. Sci.*, 493:46–65, 2013.
8. M. Jurdziński. Deciding the winner in parity games is in UP ∩ coUP. *Information Processing Letters*, 68(3):119–124, 1998.
9. R. M. Karp. Reducibility among combinatorial problems. In *Proceedings of a Symposium on the Complexity of Computer Computations*, pages 85–103, 1972.
10. D. Klein, F. G. Radmacher, and W. Thomas. Moving in a network under random failures: A complexity analysis. *Science of Comp. Prog.*, 77(7-8):940–954, 2012.
11. C. Löding and P. Rohde. Solving the sabotage game is PSPACE-hard. In *MFCS*, volume 2747 of *LNCS*, pages 531–540. Springer, 2003.
12. D. A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975.
13. D. A. Martin. The determinacy of blackwell games. *J. Symb. Log.*, 63(4):1565–1581, 1998.
14. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th Symp. Principles of Programming Languages*, pages 179–190. ACM, 1989.
15. P. D. Seymour and R. Thomas. Graph searching and a min-max theorem for tree-width. *Journal of Combinatorial Theory, Series B*, 1:22–33, 1993.
16. L. J. Stockmeyer and A. K. Chandra. Provably difficult combinatorial games. *SIAM J. Comput.*, 8(2):151–174, 1979.
17. J. van Benthem. An essay on sabotage and obstruction. In *Mechanizing Mathematical Reasoning*, volume 2605 of *LNAI*, pages 268–276. Springer, 2005.
18. M. Yamashita and I. Suzuki. Characterizing geometric patterns formable by oblivious anonymous mobile robots. *Theor. Comput. Sci.*, 411(26-28):2433–2453, 2010.
19. W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1–2):135 – 183, 1998.
20. U. Zwick and M. S. Paterson. The complexity of mean payoff games. *Theor. Comput. Sci.*, 158:343–359, 1996.

---

[5] We refer the interested reader to Appendix E for a formal discussion on these claims.

# A Relation with cops and robbers game

We observe that the result on safety games is related to the Cops and Robbers games studied mostly by the graph theoretical community (see, e.g., [1] and references therein for a survey). We remark that Cops and Robbers games are usually defined as played on the vertices of undirected graphs. In [6] it was shown that several variants of the Cops and Robbers game without helicopters and, as usual, played on the vertices of an undirected graph, are EXPTIME-complete. In contrast, our result implies that the Cops and Robbers game played on the edges of a graph with $B$ cops, one *helicopter* and a *slow robber*, i.e., which can traverse at most one edge per turn, is EXPTIME-hard. A similar version is studied in [15], where they consider helicopters and a *fast robber*. However, the game is played on the vertices of an undirected graph and the complexity of solving the game is left open in that paper. It is easy to lift our results to games where weights are placed on vertices and no longer on edges by considering line graphs: in contrast, the other direction from vertices to edges would have been more difficult, and is not currently known for the best of our knowledge.

# B Encoding of quantitative sabotage games in quantitative games

In this section, we give formal definitions of quantitative two-player games, and show an exponential encoding of QSGs into these games. Thereafter, we choose to call Min and Max the two players of our games, to distinguish them from Runner and Saboteur, used in the main part of this article.

*Two-player games.* A weighted arena is a tuple $G = (V_{\mathrm{Min}}, V_{\mathrm{Max}}, E, w, v_I)$ with $V = V_{\mathrm{Min}} \uplus V_{\mathrm{Max}}$ a finite set of vertices partitioned into the set $V_{\mathrm{Min}}$ of vertices of player Min and the set $V_{\mathrm{Max}}$ of vertices of player Max, $E \subseteq V^2$ is a set of edges, $w\colon E \to \mathbb{N}$ is a weight function assigning an integer weight to each edge of the arena, and $v_I \in V$ is an initial vertex. Given a weight function $w\colon E \to \mathbb{R}$, we write $|w|$ for the greatest weight in $w$, i.e., $|w| = \max_{e \in E} w(e)$.

Intuitively, the two players Min and Max move a token along the edges of the graph $(V, E)$, starting on vertex $v_I$. When the token is on a vertex of $V_{\mathrm{Min}}$, it is Min that chooses the next vertex, and when on $V_{\mathrm{Max}}$, it is Max. To allow them to play infinitely, we make the assumption that every vertex $v$ has an outgoing edge, i.e., that there exists $(v, v') \in E$. A strategy for a player is simply a mapping telling him what to play depending on the past. Formally, given an arena $G = (V_{\mathrm{Min}}, V_{\mathrm{Max}}, E)$, a play is an infinite sequence of vertices $v_0 v_1 v_2 \cdots \in V^\omega$ such that $v_0 = v_I$, and $(v_i, v_{i+1}) \in E$ for all $i \geqslant 0$. We say that a prefix $v_0 \cdots v_k$ of a play belongs to Min (respectively, Max) if $v_k \in V_{\mathrm{Min}}$ (respectively, $v_k \in V_{\mathrm{Max}}$). A strategy for player $p$ is a mapping $\sigma$ from prefixes of plays belonging to $p$ to vertices such that $(v_k, \sigma(v_0 \cdots v_k)) \in E$ for all prefix $v_0 \cdots v_k$ belonging to $p$. The outcomes of a strategy $\sigma$ of player $p$ are all plays $v_0 v_2 \cdots$ such that for all $v_0 \cdots v_k$ with $v_k \in V_p$, $v_{k+1} = \sigma(v_0 \cdots v_k)$. We write $Play(G)$ the set of plays in $G$ (we omit $G$ when it is clear from the context), $Play(\sigma)$ the set of outcomes of a strategy $\sigma$, and $Play(\sigma_{\mathrm{Min}}, \sigma_{\mathrm{Max}})$ the only play contained in $Play(\sigma_{\mathrm{Min}}) \cap Play(\sigma_{\mathrm{Max}})$.

Since we are dealing with quantitative game, we use a *value function* to map plays to values in $\overline{\mathbb{R}} = \mathbb{R} \uplus \{+\infty\}$. A quantitative game is a pair $(G, f)$ consisting of an arena $G$ and such a value function $f$. Most standard value functions are defined by using the weights in the weighted arena: equipped of one of the cost functions $f\colon \mathbb{R}^\omega \to \overline{\mathbb{R}}$ described in the main part of the article (Inf, LimInf, Sup, LimSup, Avg, or $\mathrm{DS}_\lambda$ for instance), we may define a value function $f_w$ by setting $f_w(v_0 v_2 \cdots) = f(w(v_0, v_1) w(v_1, v_2) \cdots)$ for all plays $v_0 v_1 \cdots$.

In a quantitative game $(G, f)$, the value of a strategy $\sigma_{\mathrm{Min}}$ (respectively, $\sigma_{\mathrm{Max}}$) of Min (respectively, Max) is:

$$\mathbf{Val}((G, f), \sigma_{\mathrm{Min}}) = \sup_{\pi \in Play(\sigma_{\mathrm{Min}})} f(\pi), \quad \mathbf{Val}((G, f), \sigma_{\mathrm{Max}}) = \inf_{\pi \in Play(\sigma_{\mathrm{Max}})} f(\pi).$$

To characterise the best value that each player can guarantee no matter what the opponent is doing, we consider the upper value $\overline{\mathbf{Val}}$ (the best Min can hope for) and lower value $\underline{\mathbf{Val}}$ (the best Max can hope for), defined by:

$$\overline{\mathbf{Val}}(G, f) = \inf_{\sigma_{\mathrm{Min}}} \mathbf{Val}((G, f), \sigma_{\mathrm{Min}}), \quad \underline{\mathbf{Val}}(G, f) = \sup_{\sigma_{\mathrm{Max}}} \mathbf{Val}((G, f), \sigma_{\mathrm{Max}}).$$

**Proposition 1.** *In quantitative games, for all the value functions $f_w$ obtained by considering the cost functions $f$ used above, upper and lower values coincide: we then let $\mathbf{Val}(G, f) = \overline{\mathbf{Val}}(G, f) = \underline{\mathbf{Val}}(G, f)$ be the value of the game.*

*Proof.* We rely on Martin's determinacy theorem for Blackwell games [13], since all the cost functions considered are Borel measurable. □

*Encoding of quantitative sabotage games.* Starting from a QSG $\mathcal{G} = (V, E, B, v_I, \delta_I, f)$, we encode it in the quantitative two-player game $[\![\mathcal{G}]\!] = ((V_{\mathrm{Min}}^c, V_{\mathrm{Max}}^c, E^c, w, v_I^c), f_w')$ as follows:

- $V_{\mathrm{Min}}^c = V \times \Delta(\mathcal{G})$, $V_{\mathrm{Max}}^c = E \times \Delta(\mathcal{G})$: Min vertices represent configurations of $\mathcal{G}$ (i.e., the vertex of $\mathcal{G}$ currently occupied by Runner, together with the current budget distribution), and Max vertices encode the last edge played by Runner in $\mathcal{G}$ and again the current budget distribution;
- $E^c = \{((v, \delta), (e, \delta)) \mid e = (v, v') \in E\} \cup \{((e, \delta), (v', \delta')) \mid e = (v, v') \wedge \delta \triangleright \delta'\}$;
- for all $e = (v, v') \in E$ and $\delta, \delta' \in \Delta(\mathcal{G})$ we let $w((v, \delta), (e, \delta)) = \delta(e)$ and
  - $w((e, \delta), (v', \delta')) = 0$ if $f = \mathsf{DS}_\lambda$,
  - $w((e, \delta), (v', \delta')) = \delta(e)$ otherwise;
- $v_I^c = (v_I, \delta_I)$ is the initial configuration;
- if $f = \mathsf{DS}_\lambda$, we let $f' = \mathsf{DS}_{\sqrt{\lambda}}$, otherwise $f' = f$.

We claim that $\mathcal{G}$ and $[\![\mathcal{G}]\!]$ are equivalent, meaning that they have the same value. The main difference lies in the way costs are computed. Indeed, consider a pair of consecutive moves from both players in the original QSG $\mathcal{G}$, i.e., the traversal of an edge $e = (v, v')$ by Runner, followed by a budget redistribution $\delta \triangleright \delta'$ by Saboteur. Observe that this pair of moves incurs a cost of $\delta(e)$ in the original QSG, but is encoded by the traversal of *two consecutive edges* $((v, \delta), (e, \delta))$ and $((e, \delta), (v', \delta'))$ in $[\![\mathcal{G}]\!]$ that have *both* weight $\delta(e)$ (or weight $\delta(e)$ and then weight 0 for the discounted sum case). Observe however that this is not a problem for the cost functions that we are considering. Indeed, Sup, Inf, LimSup, and LimInf are resistant to stuttering. The value of the average cost is also consistent, since both the sum of the visited weights *and* the length of the paths are doubled in $[\![\mathcal{G}]\!]$ with respect to $\mathcal{G}$. For the discounted sum, this is taken care of by replacing the original discount factor $\lambda$ in $\mathcal{G}$ by $\sqrt{\lambda}$ in $[\![\mathcal{G}]\!]$.

## C Proofs of Section 3

### C.1 Reduction from ABF to *ESPr*: proof of Lemma 1

In this section, we fix an instance of the ABF problem, i.e., a CNF formula $\Phi$ and an initial configuration $(\mathrm{val}_0, \mathrm{Player}_0)$. We let $N$ be the number of variables in $\Phi$. We construct an extended QSG $\mathcal{G}$ such that Saboteur wins in the extended safety problem over $\mathcal{G}$ if and only if Prover wins the ABF game. Therefore, Saboteur will act as Prover while Runner will act as Disprover. As an example of our construction, we consider the following CNF formula:

$$\Phi = Cl_1 \wedge Cl_2 \wedge Cl_3,$$
$$Cl_1 = A \vee B \vee \neg C$$
$$Cl_2 = \neg A \vee \neg C$$
$$Cl_3 = D \vee B \vee \neg C$$

where $\{A, B\}$ are Disprover's (Runner's) variables, while $\{C, D\}$ are Prover's (Saboteur's) variables. Recall that in extended QSGs, we allow for the use of safe edges, i.e., edges where Saboteur cannot put budget, and final vertices, i.e., vertices where the play ends, with the budget placed on this vertex taken into account to compute the cost. We present step by step the vertices contained in $\mathcal{G}$. For every variable $X$, we create 4 final vertices $Ver(X) = \{\neg X^{(1)}, \neg X^{(2)}, X^{(1)}, X^{(2)}\}$. We also create another final vertex called $\alpha$. In the following, we always assume that edges are safe, unless explicitly stated.

*Forcing to have at least budget 2 on $Ver(X)$.* For each variable $X$, and each triplet $t = \{v_1, v_2, v_3\} \subset Ver(X)$, we create two vertices, $t^{(1)}$ and $t^{(2)}$ such that in the graph, $(t^{(1)}, t^{(2)}), (t^{(2)}, v_i) \in E$ for all $i \in \{1, 2, 3\}$. Note that if there is zero budget on the triplet $t$ when Runner arrives in $t^{(1)}$, then Runner is sure to reach one of the $v_i$ without visiting edges with non-zero budget (and hence win the game). For all vertices $v$ in the graph (except those in the initialisation gadget, as we see later), all variables $X$ and all triplets $t$ as described above, we create an edge $(v, t)$. If at some point in the game there is a variable $X$ such that there are less than 2 vertices in $Ver(X)$ with a budget on them, then Runner is sure to win the game. This gadget is depicted in Fig. 3. In the following, we assume that Saboteur always place at least 2 units of budget on each $Ver(X)$. We also assume that if it is the case, then Runner does not go on a $t^{(1)}$ vertex (indeed he will be sure to lose the play if he does so). Saboteur always places at least 2 units of budget on each $Ver(X)$. We also assume that if it is the case, then Runner does not go on a $t^{(1)}$ vertex (indeed he will be sure to lose the play if he does so).

The budget in the game is $B = 2N + 1$. Let $v$ be a vertex that has an outgoing edge towards $\alpha$. When Runner leaves $v$, in order for Saboteur not to lose, there must be one unit of budget on $\alpha$ and exactly 2 units of budget on each $Ver(X)$.

*Forcing the budget to be well distributed.* Now we present another gadget that allows Runner to force, on some vertices, that ($\star$) either $\neg X^{(1)}$ and $\neg X^{(2)}$ have one unit of budget each, or $X^{(1)}$ and $X^{(2)}$ have one unit of budget each. To do so, for each pair $p = \{\neg X^{(i)}, X^{(j)}\}$, we construct a vertex $p$ that has two outgoing edges, $(p, \neg X^{(i)})$ and $(p, X^{(j)})$. Let $v$ be a vertex that has outgoing edges toward each of those $p$ vertices. When Runner leaves $v$, if property ($\star$) is not fulfilled, then there is a pair $p = \{\neg X^{(i)}, X^{(j)}\}$ with zero budget on it. By going on $p$, Runner ensures to reach one of those vertices without budget, hence to win the game. We let $Check(X)$ be the set of all those vertices associated with $X$, and we will describe later which vertices have outgoing edges toward $Check(X)$. This gadget is depicted in Fig. 4. When Runner leaves a vertex $v$ with an outgoing edge toward $Check(X)$, we now assume that Saboteur has made true property ($\star$) for $X$, so that Runner never goes to $Check(X)$ (indeed he will be sure to lose if he does so when ($\star$) is fulfilled).

Let $v$ be a vertex that is connected to $\alpha$ and to all vertices of $Check(X)$ for all $X$ (we let $Check = \bigcup_{X \in Var} Check(X)$). When Runner leaves $v$, in order for Saboteur not to lose, there must be one unit of budget on $\alpha$, and one unit of budget either on $\neg X^{(1)}$ and $\neg X^{(2)}$, or on $X^{(1)}$ and $X^{(2)}$, for all $X$. We call such a configuration a valid one, and remark that there is an immediate bijection from valid configurations and valuations of the variables of the CNF formula. We call a *valid vertex* a vertex connected to $\alpha$ and to $Check$.

*Initialising the game.* We add a gadget, at the beginning of the game, forcing Saboteur to distribute the budget accordingly to the initial valuation $val_0$ of the ABF game. The gadget works as follows: Runner crosses $2N + 1$ safe edges successively, and then goes on a vertex $v$ that has edges towards each vertex of the required initial configuration. Saboteur has the time to put the required units of budget on this configuration, and is forced to do so, otherwise Runner would be able to reach a final vertex. Therefore, we are sure that once this gadget is left to start playing the game, the configuration is indeed the required one. From vertex $v$, there is also another safe edge going either to the vertex $Play$ or to the vertex $set^{(1)}$ (the role of both vertices is explained later), depending on whether $Player_0$ is Disprover or Prover, respectively.

*Structure of the graph.* From the CNF formula presented above as an example, we construct the graph depicted in Fig. 6. For the sake of clarity, we omit the gadgets introduced above. Double bordered vertices represent final vertices, and double arrows represent safe edges. As stated above, from all vertices depicted here, gadget of Fig. 3 is used to check that $Ver(X)$ contains at least budget 2, for all variables $X$. The subscript in $\{A, B, C, D, \alpha\}$ on vertices depicts an edge from the vertex to the corresponding gadget $Check$, or vertex $\alpha$.

*Saboteur modifies Prover's variables.* The two safe vertices $set^{(1)}$ and $set^{(2)}$ describe Prover's turn to modify one of its variables. Both vertices have an outgoing edge towards $\alpha$ ensuring that one pebble is left on it.
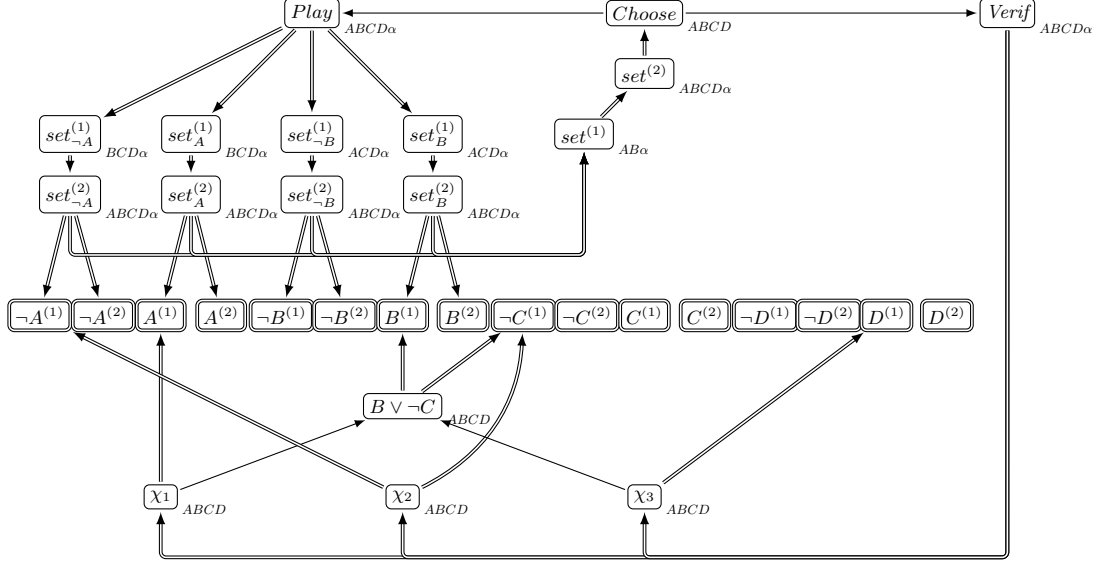
Fig. 6: The game of the example (the legend is given in the text)

$set^{(2)}$ is a valid vertex, and $set^{(1)}$ is connected to $Check(X)$ for all variables $X$ belonging to Disprover. Finally, there is an edge $(set^{(1)}, set^{(2)})$ connecting those two vertices.

Let $v$ be a valid vertex with an outgoing edge $set^{(1)}$ and let $\text{val}_1$ be the valuation of variables induced by a valid configuration at the moment Runner leaves $v$. If Runner goes to the vertex $set^{(1)}$ and then to $set^{(2)}$, let $\text{val}_2$ be the valuation induced by the valid configuration at the moment Runner leaves $set^{(2)}$. We claim that between $\text{val}_2$ and $\text{val}_1$, at most one variable of Prover has been modified. Indeed after Runner has arrived in $set^{(1)}$, Saboteur cannot remove the budget on $\alpha$, and he cannot take the budget on some $Ver(X)$ to put it on another $Ver(X')$, with $X' \neq X$, as there would be only budget 1 on $Ver(X)$ and Runner would win. Therefore, the only possible move for Saboteur is to redistribute the budget inside some $Ver(X)$. Moreover, if $X$ belongs to Disprover after a move, $Ver(X)$ will not satisfy the property $(\star)$ and, since $set^{(1)}$ is connected to $Check(X)$, Runner would win. Therefore, either Saboteur does nothing, or he redistributes the budget inside some $Ver(X)$ where $X$ belongs to Prover. If he has done nothing then after Runner has gone to $set^{(2)}$, by the same reasoning, and by the necessity that at this moment the configuration is valid, one can ensure that again Saboteur does nothing, in which case we would have $\text{val}_2 = \text{val}_1$. Let us focus on the case where Prover has performed some redistribution in $Ver(X)$. Without loss of generality, assume that when leaving $v$, the budget was placed on $X^{(1)}$ and $X^{(2)}$, and after leaving $set^{(1)}$ the budget is on $\neg X^{(1)}$ and $X^{(2)}$. By the same reasoning, we know that after reaching $set^{(2)}$, Saboteur can only redistribute the budget inside a $Ver(X')$ where $X'$ belongs to Prover. Furthermore, if $X' \neq X$ then, when leaving $set^{(2)}$, $Ver(X)$ would not satisfy $(\star)$ and the configuration would not be valid. Therefore Saboteur can either choose to have the budget on $X^{(1)}$ and $X^{(2)}$, or on $\neg X^{(1)}$ and $\neg X^{(2)}$, therefore between $\text{val}_2$ and $\text{val}_1$ only the valuation of $X$ may have change.

*Runner modifies Disprover's variables.* From the vertex $Play$, Runner chooses a variable $X$ of Disprover, and goes either to $set^{(1)}_{\neg X}$ or to $set^{(1)}_X$: assume without loss of generality that he goes to $set^{(1)}_X$. Those two vertices have outgoing edges toward $\alpha$ and toward $Check(X')$ for all $X' \neq X$. Let $\text{val}_1$ be the valuation associated with the valid configuration when Runner leaves $Play$. After arriving in $set^{(1)}_X$, Saboteur can only redistribute the budget inside $Ver(X)$. After arriving in $set^{(2)}_X$, Saboteur is forced to reach a valid valuation, therefore if he has modified the budget distribution in $Ver(X)$, he must do it again in order for $Ver(X)$ to satisfy $(\star)$. Furthermore, as $set^{(2)}_X$ has outgoing edges to the two final vertices $X^{(1)}$ and $X^{(2)}$, there must be

a unit of budget on each of those vertices. Therefore, if we let $\text{val}_2$ be the valuation induced by the valid configuration when Runner leaves $set_X^{(2)}$, $\text{val}_2$ must be equal to $\text{val}_1$ except possibly for $X$ that must now be true.

*Verifying a valuation.* Before explaining the whole behaviour of the game, let us describe the verification process. As *Verif* is a valid vertex, when Runner leaves this vertex, the configuration is valid: we therefore let val be the valuation induced by this configuration. We show here that, from the moment Runner leaves *Verif*, Saboteur has a winning strategy if and only if val satisfies the CNF formula. Let us first describe this part of the arena.

*Verif* has one outgoing safe edge toward each vertex $\chi_i$ associated with the eponym clause. Those vertices are connected to *Check*. Take a clause $\chi_i = at_1 \wedge \cdots \wedge at_\ell$. For each strict suffix of this clause containing at least two atoms, i.e., for each sub-clause of the form $at_j \wedge \cdots \wedge at_\ell$ with $1 < j < \ell$, create an eponym vertex. Then $\chi_i$ has a safe edge toward $at_1^{(1)}$ and a (non safe) edge toward the rest of the clause, i.e., nothing if $\ell = 1$, $at_2^{(1)}$ if $\ell = 2$, and the vertex '$at_2 \wedge \cdots \wedge at_\ell$' if $\ell > 2$. The same principle applies to the vertex '$at_2 \wedge \cdots \wedge at_\ell$', etc. For example, take $\chi_1$ in the CNF formula $\Phi$. The vertex $\chi_1$ has edges toward $\neg A^{(1)}$ and toward the vertex '$B \vee \neg C$' which is the rest of the clause. Then the vertex '$B \vee \neg C$' has an edge toward $B^{(1)}$ and an edge toward $\neg C^{(1)}$.

Assume first that val satisfies the formula, and let us see how Saboteur has a winning strategy. When Runner reaches a clause $\chi_i$, we know that it is true in val, i.e., that one of its atom is true. On the game, this is represented by the fact that one of the atoms $at_j$ has non-zero budget on the two associated vertices $at_j^{(1)}$ and $at_j^{(2)}$. For example assume that Runner goes to $\chi_1$ and that $B$ is true, i.e., there is some budget on $B^{(1)}$ and $B^{(2)}$. Saboteur will use the budget on $\alpha$ to guide Runner in direction of this atom. In the example, when Runner reaches $\chi_1$, Saboteur will put the budget on $A^{(1)}$, then when Runner will go to '$B \vee \neg C$', Saboteur will move the same unit of budget on $\neg C^{(1)}$, forcing Runner to go to $B^{(1)}$. However, as there was already some budget on $B^{(1)}$, Runner cannot leave '$B \vee \neg C$' without touching some non-zero budget, and loses the safety game.

On the other hand, assume that val does not satisfy the formula and let us see how Runner has a winning strategy. As the valuation does not satisfy the formula, there exists a clause $\chi_i$ that is false. Runner goes to this clause. As it is false, all the atoms are false, in particular, in the game, for all $at_j$, there is budget 0 on $at_j^{(1)}$. Runner will have the following behaviour. If, after reaching $\chi_i$, Saboteur has not put some budget on $at_1^{(1)}$, then he goes there and wins, otherwise he goes to the vertex representing the rest of the formula. From there, the same reasoning applies: if Saboteur has not put some budget on $at_2^{(1)}$, then Runner goes there and wins, otherwise he reaches the next sub-clause. At the end, Runner reaches the vertex '$at_{\ell-1}^{(1)} \vee at_\ell^{(1)}$', and whatever Saboteur does, Runner reaches a final vertex with budget 0.

*How the game works.* When Runner leaves vertex *Play*, the configuration is valid; once he reaches $set^{(1)}$, the configuration is valid again, and the difference with the previous one is that the valuation may have changed for at most one variable. Once reaching *Choose*, Saboteur may also have changed the valuation of one of its variables. When Runner reaches *Choose*, Saboteur can only redistribute the budget on $\alpha$. One can easily see that he has no interest in changing the valuation by putting some budget in $Ver(X)$ for some variable $X$, as at the next step he must put the budget back on $\alpha$. However, Saboteur can either put the free unit of budget on the edge $(Choose, Play)$, forcing Runner to go to the verification part on the game, or put it on the edge $(Choose, Verif)$, forcing Runner to remain in the part of the game where they change the valuation. If Saboteur has a winning strategy in the ABF game, he will apply it, and once the valuation satisfies the formula, he will force Runner to go to the verification part. On the other hand, if the formula is never true, Saboteur is forced to prevent Runner from going to the verification part (otherwise Runner would reach a final vertex as seen above), and the game will last forever, allowing Runner to win.

## C.2 Reduction from *ESPr* to *SPr*: proof of Lemma 2

We describe how to transform an extended QSG into a regular QSG. The transformation rids the original sabotage game of its safe edges and final vertices, and replaces them with corresponding gadgets with the same properties.

Final vertices are replaced by the gadget shown in Fig. 5a. More formally, all edges incident in a final vertex are replaced by edges incident on a copy of the gadget. $A$ is the entry point of the gadget, i.e., any edge pointing towards the final vertex in the extended QSG would now lead to $A$. Vertices $C_1$ and $C_2$ are both connected to $\alpha_i$, for all $1 \leqslant i \leqslant B+1$, and the $\alpha_i$'s form a clique of size $B+1$. It should be clear that, if Runner reaches one of the $\alpha_i$, then he can ensure that the value of the play, from then onwards, is exactly 0. Indeed, as there are $B+1$ outgoing edges, at least one of them has no budget on it; if Runner crosses this edge, he reaches another $\alpha_j$ where the same property holds. Thus, one can easily see that when Runner reaches $A$, he can win if and only if there is no budget on either one of the edges: $(A, C_1)$, $(A, C_2)$.

Safe edges can be encoded as follows. Assume that we have a safe edge $(A, C)$ in the extended QSG. To encode it in a standard QSG (with final vertices, as we have already seen how to encode them), we add $B+1$ vertices $E_1, \ldots, E_{B+1}$, and $B+1$ final vertices $F_1, \ldots, F_{B+1}$. We remove the edge $(A, C)$, and add the edges $(A, E_i)$, $(E_i, F_i)$ and $(E_i, C)$, for all $i \leqslant B+1$. The gadget is depicted in Fig. 5b. Runner has a strategy to go from $A$ to $C$ without crossing an edge with non-zero budget, and forcing Saboteur to move at most one unit of budget inside the game. That is to say, we have introduced one additional step to get from $A$ to $C$, but we will see that Saboteur cannot move more than one unit of budget on edges outside of the gadget, or he loses. Indeed, when Runner leaves $A$, there must exist $i$ such that there is no budget on edges $(A, E_i)$, $(E_i, F_i)$, $(E_i, C)$ nor on the final vertex $F_i$. If Runner goes to $E_i$, Saboteur must take a unit of budget and put it either on $(E_i, F_i)$ or on $F_i$, otherwise Runner can reach $F_i$ and win. Now, Runner is able to reach $C$, and then Saboteur can redistribute the budget as he wants.

## C.3 Reduction from $ThPr_{lsupfun}(0)$ to *SPr*: proof of Lemma 4

Consider an instance $\mathcal{I}$ of the safety problem with underlying graph $G(\mathcal{I}) = (V, E)$, budget $B$, and a starting vertex $v_I$. We build a QSG $\mathcal{G}'$ with graph $G' = (V', E')$, initial vertex $t_1$, budget $B'$, and cost function LimSup as follows:

$$B' = |E|,$$
$$V' = V \cup \{s_i^j \mid 1 \leqslant i \leqslant B'+1, 1 \leqslant j \leqslant B'\} \cup \{x_m \mid B+1 \leqslant m \leqslant B'\} \cup \{t_1, t_2\}$$
$$\quad \cup \{f_k \mid 1 \leqslant k \leqslant B\} \cup \{e_\ell \mid 1 \leqslant \ell \leqslant B+1\},$$
$$E' = E \cup \{(e_i, f_j) \mid 1 \leqslant i \leqslant B+1, 1 \leqslant j \leqslant B\} \cup \{(f_k, e_k), (f_k, e_{k+1}) \mid 1 \leqslant k \leqslant B\}$$
$$\quad \cup \{(x_\ell, t_m) \mid 1 \leqslant m \leqslant 2, B+1 \leqslant \ell \leqslant B'\} \cup \{(t_m, s_i^{B'}) \mid 1 \leqslant i \leqslant B'+1, 1 \leqslant m \leqslant 2\}$$
$$\quad \cup \{(s_i^j, s_{i'}^{j-1}) \mid 1 \leqslant i, i' \leqslant B'+1, 2 \leqslant j \leqslant B'\} \cup \{(s_i^1, e_m) \mid 1 \leqslant m \leqslant 2, 1 \leqslant i \leqslant B+1\}$$
$$\quad \cup \{(e_m, u) \mid 1 \leqslant m \leqslant B+1, (v_I, u) \in E\} \cup \{(u, x_\ell) \mid u \in V, B+1 \leqslant \ell \leqslant B'\}$$
$$\quad \cup \{(e_m, x_\ell) \mid 1 \leqslant m \leqslant B+1, B+1 \leqslant \ell \leqslant B'\}.$$

Intuitively, the sub-graph of $G'$ defined by the vertices $e_i$ and $f_j$ form an initial gadget which ensures that Runner can stay out of $G(\mathcal{I})$ without paying, as long as there is some weight assigned to edges from $E$. We also add an exit gadget consisting of the sub-graph of $G'$ defined by the $x_k$ vertices. These allow Runner to exit from $G(\mathcal{I})$ if Saboteur "cheats" by assigning more weights to edges from $E$ than the original bound $B$. Both gadgets are linked by a "safe path" formed by the vertices $s_i^j$. Note that we add sufficiently many $s_i^j$ so that, for Runner, getting from any $s_i^{B'}$ to any $s_j^1$ is always possible without traversing a weighted edge.

We prove that Runner wins in $\mathcal{I}$ if and only if $\mathbf{Val}(\mathcal{G}') \leqslant 0$.

Assume first Runner wins $\mathcal{I}$. In $\mathcal{G}'$, he has no trouble following a path from $t_1$ through the $u_i^j$ until he arrives on some $u_i^1$ with budget distribution $w_0$ such that $w_0(u_i^1, e_j) = 0$, for some $1 \leqslant j \leqslant B+1$, since there are $B'+1$ vertices at each level of the safe path. On his next turn, he can then move to such an $e_j$. As

long as the budget distribution has some budget assigned to some edge of $E$, there exists a vertex $e_k$ or $x_\ell$ with no budget on either in-edges or out-edges, respectively. In the first case, Runner can go to such such an $e_k$ via $f_k$ without paying anything. In the second case, Runner can get to $t_1$ or $t_2$ via $x_\ell$ and repeat the process, all without paying. When the budget distribution has no weight assigned to edges of $E$, Runner can follow his strategy from $\mathcal{I}$ – with the exception that he plays his first move from $e_j$ instead of $v_I$ – as long as Saboteur keeps at most $B$ budget units on edges of $E$. When this is no longer the case, say Runner is on a vertex $u$, with budget distribution $w_1$, that means there are at most $B' - B - 1$ budget units on other edges, hence there is a vertex $x_\ell$ such that $w_1(u, x_\ell) = w_1(x_\ell, t_1) = w_1(x_\ell, t_2) = 0$. Runner then moves to $x_\ell$. On his next turn, he can then move to either $t_1$ or $t_2$, following an edge with no weight on it. Then Runner can restart this strategy.

Assume now, that Saboteur wins $\mathcal{I}$. From the start of the game, Runner will have to traverse one $s_i^j$ for all $j$ from $B'$ to 1. When Runner is on a vertex $s_i^j$ for $j$ between $B + 1$ and $B'$, Saboteur puts a budget unit on the edge $(x_j, t_1)$ and leaves it there. Similarly, when Runner is on a vertex $u_i^j$ for $j$ between 1 and $B$, Saboteur puts a unit of budget back on the edge $(f_j, e_j)$ and leaves it there. When Runner finally reaches some $e_k$, Saboteur passes. Then, if Runner goes to $f_\ell$ or $x_m$, Saboteur can assign some budget to $(f_\ell, e_{\ell+1})$ or $(x_m, t_2)$ and put it back where it was after Runner's next move, where he will inevitably cross a weighted edge, then wait until Runner gets back to some $e_k$. Alternatively, from $e_k$, Runner can move to a vertex in $G(\mathcal{I})$. In this case, Saboteur follows his strategy from $\mathcal{I}$, using budget units assigned to edges of the form $(f_k, e_k)$ when needed, until Runner crosses an edge of $E$ with some weight on it or gets to some $x_\ell$. In the latter case, Saboteur can react the same way as if Runner was coming from $e_k$. In the former case, Saboteur can start putting some weights on all edges of $E$ until Runner gets to some $x_\ell$. If Runner never does, he will pay one at each step, which is enough for Saboteur. Otherwise, Runner goes to some $x_\ell$, then to $t_1$ or $t_2$, where Saboteur can restart his strategy.

# D   Proofs of Section 4

## D.1   Static threshold problem for Inf and LimInf is in PTIME: proof of Lemma 6

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, for a given budget distribution $\delta$, Runner simply goes towards the edge reachable from $v_I$ with the least budget possible; therefore, Saboteur must place *equal* budget on each such edge. With a budget $B$, he can ensure $\lfloor |\overline{E}|/B \rfloor$ on every edge (some edges may contain a bigger portion of the budget, but some edges will always have at most $\lfloor |\overline{E}|/B \rfloor$). Hence, deciding the static threshold problem for Inf amounts to computing the set $\overline{E}$ (can be done in linear time with a depth-first-search algorithm), and checking whether $|\overline{E}| \leqslant B \times (T + 1)$.

For LimInf, we must refine the study by considering strongly connected components. Precisely, we claim 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 of the graph. Indeed, for a given budget distribution $\delta$, Runner simply goes towards a cycle reachable from $v_I$ containing an edge with the least budget $b$ possible: he will visit infinitely often this edge, ensuring an inferior limit at most $b$. Such a cycle is included in a strongly connected component, and reciprocally, every edge of a strongly connected component is part of a cycle. Hence, Saboteur must secure *equal* budget on each edge of every strongly connected components. Then, deciding the static threshold problem for LimInf amounts to computing the set $\widetilde{E}$ (can be done in linear time, e.g., with Tarjan's algorithm), and checking whether $|\widetilde{E}| \leqslant B \times (T + 1)$.

## D.2   Static threshold problem for Sup, LimSup, Avg and DS is coNP-complete: proof of Lemma 7

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]).

To prove the NP-hardness for cost functions LimSup and Avg, we give a reduction from the following problem. The FEEDBACK ARC SET PROBLEM consists in, given as input a directed graph $G = (V, E)$ and a threshold $k \leqslant |E|$, determining whether there is a set $E'$ of $k$ edges of $G$ such that $(V, E \setminus E')$ is acyclic. Karp showed in [9] that the feedback arc set problem is NP-complete.

We now use the feedback arc set problem to prove the results of coNP-hardness of the static threshold problem. Let us consider an instance of the feedback arc set problem, given by a directed graph $G = (V, E)$ and a natural integer $k \leqslant |E|$. We suppose, without loss of generality, the existence of a vertex $v_I$, without any in-going edges, and linked with an edge to every other vertex: since $v_I$ is not included in any cycle, the set $E'$ of the output of the problem has no interest at containing any of the edges added in this way.

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. Indeed, $\mathbf{Val}_{\mathrm{stat}}(\mathcal{G}) > 0$ implies that there exists a distribution $\delta \in \Delta(\mathcal{G})$ such that for all strategies $\rho$ of Runner, $f(\overline{\pi_{\rho,\iota}^{\delta}}) > 0$. Noticing that every vertex is reachable from the initial vertex $v_I$, and considering memoryless strategies of Runner (such that $\pi_{\rho,\iota}^{\delta}$ ends with a simple cycle of the graph), we show that every cycle contains at least one edge with a non-zero budget. The set $E' = \{e \in E \mid \delta(e) > 0\}$ is then a valid output for the feedback arc set problem. For the reciprocal implication, we simply assign a budget 1 to each vertex of the set $E'$.

The result for Sup and DS is then obtained by a slight modification of the previous proof. Let $\underline{\mathcal{G}} = (\underline{V}, \underline{E}, k, v_I, \mathsf{Sup})$ be the QSG obtained from $\mathcal{G}$ by transforming every edge $(v_I, v)$ into a safe edge (see Lemma 2). Without loss of generality, we can now assume that $\delta$ never assigns budget to the edges $\underline{E} \setminus E$. We also note that $v_I$ has no in-going edges so that every play in $\underline{\mathcal{G}}$ traverses a safe edge at most once. We claim that $\mathbf{Val}_{\mathrm{stat}}(\underline{\mathcal{G}}) > 0$ if and only if there exists a set $E'$ of $k$ edges of $G$ such that $(V, E \setminus E')$ is acyclic. Indeed, if $\mathbf{Val}_{\mathrm{stat}}(\underline{\mathcal{G}}) > 0$, considering $E' = \{e \in E \mid \delta(e) > 0\}$, it is easy to show that $(V, E \setminus E')$ is acyclic: if not, Runner may simply jump from $v_I$, with a safe edge, to one of the vertices of a cycle of $(V, E \setminus E')$, and then loop in this cycle forever, without visiting any edge with non-zero budget. For the reciprocal implication, again, it suffices to assign a budget 1 to each vertex of $E'$. The result for $\mathsf{DS}_\lambda$ follows from the same reduction together with Lemma 3.

# E   Towards more expressive sabotage games

In this section we increase the expressiveness of the definition of sabotage games, and show that the threshold problem for these new games are still in EXPTIME. The lower bound is immediate since they are extensions of previous problems shown EXPTIME-hard in the rest of the article.

One can see a sabotage game as a system in which a controller tries to evolve while avoiding as much as possible the weights put by Saboteur. The vertices of the graph represent configurations of the system, edges represent the actions, and the budget of the Saboteur may represent several problems that can occur during the execution. For example, it may describe a number of failures that can happen at the same time, or in a much quantitative way, it may represent how much some elements of the systems are overload, and then how much it would cost, in terms of time or energy, to use them.

We propose to look at sabotage as a particular semantics of systems. Based on the observation of Appendix B, remember that one can define the semantics of a QSG $\mathcal{G} = (V, E, B, v_I, \delta_I, f)$ as a quantitative two-player game $[\![\mathcal{G}]\!]$. If we split the model (the graph $G = (V, E)$ with initial vertex $v_I$), from the sabotage parameters (budget $B > 0$, initial distribution $\delta_I \in \Delta(E, B)$, and cost function $f$), we can define:

$$[\![G]\!]_{B, \delta_I, f} = [\![(V, E, B, v_I, \delta_I, f)]\!] \,.$$

We have seen that the value of the QSG $(V, E, B, v_I, \delta_I, f)$ is identical to the value of the quantitative two-player game $[\![G]\!]_{B, \delta_I, f}$.

From a model 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 as modelling systems with uncontrollable actions (i.e., as two-player games), and where the controller has a specific Boolean goal to

achieve, instead of simply visiting the graph *ad vitam æternam*. A more realistic goal is usually expressed via LTL formulas, that can be modelled into qualitative games with parity winning conditions, as we show in the following.

*Qualitative two-player games.* As a complement of the quantitative two-player games defined in Appendix B, we now focus on *qualitative* two-player games games. Consider a weighted arena $G = (V_{\text{Min}}, V_{\text{Max}}, E, w, v_I)$ as before. In the qualitative setting, we are no longer interested in associating a value to each play (in particular, the weight function $w$ is of no use here), but simply stating whether a play is winning or not for a player. Formally, a *winning condition* is a subset of $V^\omega$ containing the set of winning plays. A qualitative game is a pair $(G, C)$ consisting of an arena $G$ and a winning condition $C$. A play $\pi$ is declared winning for Min (respectively, for Max) if $\pi \in C$ (respectively, $\pi \notin C$). A strategy $\sigma_p$ of player $p$ is winning for $p$ if all plays $\pi \in Play(\sigma_p)$ are winning; a play/strategy is losing for player $p$ otherwise. We say that player $p$ wins (respectively, loses) the game if he has (respectively, does not have) a winning strategy. Here are some usual winning conditions considered widely in the literature:

- for all $F \in V$ or $F \in E$, *Reach*$(F)$ is the set of plays that contain an occurrence of $F$.
- for all $F \in V$ or $F \in E$, *Safe*$(F)$ is the set of plays that do not contain any occurrence of $F$.
- for all $F \in V$ or $F \in E$, *Büchi*$(F)$ is the set of plays that contain infinitely many occurrences of $F$.
- for all $F \in V$ or $F \in E$, *coBüchi*$(F)$ is the set of plays that contain only finitely many (possibly none) occurrences of $F$.
- for all *Col*$: V \to \mathbb{N}$ (such mapping is called a *colouring function*), *Parity*$(Col)$ is the set of plays $v_0 v_1 \cdots$ such that the greatest colour appearing infinitely often in the sequence $Col(v_0), Col(v_1), \ldots$ is even. Given a colouring function *Col*, we let $|Col|$ be the number of different colours of the vertices, i.e., $|Col| = |\{Col(v) \mid v \in V\}|$.
- for all value function $f$, and $T \in \mathbb{R}$, *Threshold*$^{\leqslant}(f, T)$ is the set of plays $\pi$ such that $f(\pi) \leqslant T$.

**Proposition 2.** *Qualitative two-player games with all winning conditions considered above are determined, i.e., one player is winning if and only if his opponent is losing.*

*Proof.* Martin's determinacy theorem [12] applies here since all the above mentioned objectives are Borel sets. □

*Sabotage in parity games.* In order to apply a sabotage semantics to qualitative game, where Min wants to satisfy a condition while minimising a cost, one must study some mixture between qualitative and quantitative aspects. We see how one can combine winning conditions and value functions, as introduced in [3]. Intuitively, in a weighted arena $G = (V_{\text{Min}}, V_{\text{Max}}, E, w, v_I)$ with a winning condition $C$ and a value function $f$, Min could want to satisfy $C$ while minimising $f$. We formalise this by building a new value function, denoted by $C \wedge f$, and defined by $C \wedge f(\pi) = +\infty$ if $\pi \notin C$, and $C \wedge f(\pi) = f(\pi)$ otherwise. The quantitative two-player game $(G, C \wedge f)$ now contains the combination of both objectives.

We may finally introduce a sabotage semantics for parity games. Instead of deciding whether a player has a winning strategy, which would be a standard semantics, we decide whether he has a winning strategy that guarantees (or simply avoids in the case of a threshold 0) a certain threshold over the quantity of penalties when the game is subject to failures.

Formally, given a two-player parity game $G = ((V_{\text{Min}}, V_{\text{Max}}, E, v_I), Parity(Col))$, a budget $B > 0$, an initial distribution $\delta_I \in \Delta(E, B)$, and a cost function $f \in \{\mathsf{Inf}, \mathsf{LimInf}, \mathsf{Sup}, \mathsf{LimSup}, \mathsf{Avg}\}$, the $B, \delta_I, f$-*sabotage semantics of $G$* is the quantitative game

$$[\![G]\!]_{B,\delta_I,f} = ((V'_{\text{Min}}, V'_{\text{Max}}, E', w, v_I^c), Parity(Col') \wedge f_w),$$

where :

- $V'_{\text{Min}} = V_{\text{Min}} \times \Delta(E, B)$, and $V'_{\text{Max}} = (V_{\text{Max}} \times \Delta(E, B)) \uplus (E \times \Delta(E, B))$: with respect to the one-player case of Appendix B, we add some vertices to player Max that has now in charge both the moves of the environment (uncontrollable actions), and redistributions of Saboteur;

- $E' = \{((v,\delta),(e,\delta)) \mid e = (v,v') \in E \wedge \delta \in \Delta(E,B)\} \cup \{((e,\delta),(v',\delta')) \mid e = (v,v') \in E \wedge \delta \rhd \delta'\}$;
- $w((v,\delta),(e,\delta)) = w((e,\delta),(v',\delta')) = \delta(e)$;
- $v_I^c = (v_I, \delta_I)$ is the initial configuration;
- $Col'((v,v'),\delta) = Col'(v',\delta) = Col(v')$.

To simplify our study, we do not consider the discounted-sum in this section. The threshold problem, describing the cost that player Min can ensure, is then defined as previously.

**Definition 1 (Threshold problem for cost function $f$).**
INPUT: *A parity game* $(G, Parity(Col))$, *a budget* $B$, *an initial distribution* $\delta_I \in \Delta(E,B)$, *and a threshold* $T$,
OUTPUT: *Is there a strategy* $\sigma$ *of* Min *such that* $\mathbf{Val}(\llbracket G \rrbracket_{B,\delta_I,f}, \sigma) \leqslant T$?

We are able to show that, even with the extension, the threshold problem stays in EXPTIME.

**Theorem 3.** *The threshold problem for cost functions* Inf, LimInf, Sup, LimSup *and* Avg *is in* EXPTIME.

To prove this theorem, we first establish a crude (but sufficient) upper bound on the complexity of solving quantitative games obtained by combining parity winning conditions and the previous cost functions.

**Proposition 3.** *There exists three polynomials* $p_1, p_2, p_3$ *such that we can decide the threshold problem of any quantitative game* $(G, Parity(Col) \wedge f_w)$ *with* $f \in \{$Inf, LimInf, Sup, LimSup, Avg$\}$ *with a complexity in* $O\big(p_1(|w|) \cdot p_2(|V|)^{p_3(|Col|)}\big)$.

*Proof.* We start with the case $f = $ Avg. In [19,1] it has been shown that one can decide who wins in a qualitative game with a parity condition with a complexity in $O(|V|^{2+|Col|})$. In [20], it has been shown that one can compute the value of a quantitative game with an average cost function with a complexity in $O(|w| \cdot |V|^5)$. The combination has been studied thoroughly in [3]. There, it has been shown that if one can solve average cost games in $c_1(|V|,|w|)$ and parity games in $c_2(|V|,|Col|)$, then one can solve games $(G, Parity(Col) \wedge $Avg$_w)$ with a complexity in $O(|V|^{|Col|}(|V|^2 + c_1(|V|,|w|) + c_2(|V|,|Col|)))$. By combining this result with the two above, we obtain a complexity in $O((|w|+1)|V|^{5+|Col|})$.

We then turn to the case $f \in \{$Inf, Sup, LimInf, LimSup$\}$. Our proof goes by encoding $f$ into a qualitative winning condition, and then using classical results of algorithmic game theory. Observe that deciding the threshold problem in the game $(G, Parity(Col) \wedge f_w)$ amounts to solving the following problem:

> INPUT: A weighted arena $G = (V_{\mathrm{Min}}, V_{\mathrm{Max}}, E, w, v_I)$, a colouring function $Col$, a rational threshold $T$
> OUTPUT: Does Min have a winning strategy in $(G, Parity(Col) \cap Threshold^{\leqslant}(f_w, T))$?

The crucial remark is that, if we let $F = \{e \in E \mid w(e) \leqslant T\}$, we can rewrite the threshold sets for all payoff functions as follows:

$$Threshold^{\leqslant}(\mathsf{Inf}_w, T) = Reach(F), \qquad Threshold^{\leqslant}(\mathsf{LimInf}_w, T) = B\ddot{u}chi(F),$$
$$Threshold^{\leqslant}(\mathsf{Sup}_w, T) = Safe(E \setminus F), \qquad Threshold^{\leqslant}(\mathsf{LimSup}_w, T) = coB\ddot{u}chi(E \setminus F).$$

Notice that $F$ is a subset of edges, and not vertices. However, it is easy to transform the problem into an equivalent problem where $F$ is indeed a subset of edges. Informally, it suffices to enrich the vertex set by letting $V' = V \times \{0,1\}$, letting $(v_I, 0)$ the initial vertex instead of $v_I$, and replacing each edge $(v,v') \in E$ by the set of edges in $E'$:

- $\{((v,0),(v',0)), ((v,1),(v',0))\}$ if $(v,v') \notin F$;
- $\{((v,0),(v',1)), ((v,1),(v',1))\}$ if $(v,v') \in F$.

Then, letting $F' = V \times \{1\}$ and $Col'(v) = Col'(v,0) = Col'(v,1) = Col(v)$, allows us to keep track, in the vertices, of whether the last seen edge is in $F$ or not.

Now to conclude the proof, we describe two polynomials $p_1$ and $p_2$ such that deciding if Min can win qualitative problem with a winning condition obtained by the intersection of a parity condition and another one from $\{Reach(F), B\ddot{u}chi(F), Safe(F), coB\ddot{u}chi(F)\}$ with $F \subseteq V$ can be done with a complexity in

$O(p_1(V)^{p_2(|Col|)})$. We let $G = (V_{\mathrm{Min}}, V_{\mathrm{Max}}, E, v_I)$ the arena on which we play (the weight function is of no use anymore).

For $Reach(F)$ (respectively, $Safe(F)$), one can construct in polynomial time a parity game $(G' = (V'_{\mathrm{Min}}, V'_{\mathrm{Max}}, E', v_I), Parity(Col'))$ such that $V \subseteq V'$, $|Col'| \leqslant |Col| + 1$, and Min wins in $(G', Parity(Col'))$ if and only if Min wins in $(G, Parity(Col) \cap Reach(F))$ (respectively, $(G, Parity(Col) \cap Safe(F))$). For $Safe(F)$, we remove for each vertex $v \in F$ every outgoing edge in $F$, and add a self loop, colouring them with an odd colour. Hence, if a play reaches such a vertex the play is losing, and otherwise, it is winning if and only if the greatest colour seen infinitely often is even. For $Reach(F)$, we create two copies of the game. In the first copy, every colour is odd, and for every vertex $v$ in $F$ the outgoing edges are modified to go to the same target but in the second copy. In the second copy, nothing is changed. The play start in the first copy. In order to win, Min must go to the second copy (otherwise the colour will always be odd), i.e., must reach a vertex in $F$ and then the greatest colour seen infinitely often must be even. As we have seen above parity games can be solved $O(|V|^{2+|Col'|})$ which concludes the proof for these cases.

For $Büchi(F)$ and $coBüchi(F)$, there exist two colouring functions $Col'$ and $Col''$ such that: $|Col'| = |Col''| = 2$, $Büchi(F) = Parity(Col')$, and $coBüchi(F) = Parity(Col'')$. Indeed, for $v \in F$, simply consider $Col'(v) = 2$ and $Col''(v) = 1$, and for $v \notin F$ consider $Col'(v) = 1$ and $Col''(v) = 0$. Therefore solving a game with a winning condition of the form $Parity(Col) \cap Büchi(F)$ or $Parity(Col) \cap (co)Büchi(F)$ can be turned into solving a game with a winning condition of the form $Parity(Col) \cap Parity(Col')$ with $|Col'| \leqslant 2$. Such games have been studied in [4]. They have shown that they can be solved with a complexity in $O(|V|+2)^{5(|Col|+2)^2})$, which concludes the proof of the proposition. □

We can finally establish the complexity of solving sabotage parity games.

*Proof (of Theorem 3).* From a parity game $(G, Parity(Col))$, a budget $B$, an initial distribution $\delta_I$, and a threshold $T$, one can construct $[\![G]\!]_{B,\delta_I,f} = ((V'_{\mathrm{Min}}, V'_{\mathrm{Max}}, E', w, v'_I), Parity(Col') \wedge f_w)$ in exponential time. Proposition 3 shows that we can decide who wins from $(v_I, \delta_I)$ in this game with a complexity in $O(p_1(|w|) \cdot p_2(|V'|)^{p_3(|Col'|)})$.

We have $|w| = B$ and $|Col'| = |Col|$. Furthermore $|V'| \leqslant |E| \times |\Delta(B, E)|$. As $|\Delta(B, E)| \leqslant B^{|E|}$ and $|E| \leqslant |V|^2$, we have $|V'| \leqslant |V|^2 \times B^{|V|^2}$. Since $B$ is given in binary, we can suppose that $B$ is at most exponential in the size of the input of the problem, which, in summary, shows that we can solve the threshold problem in exponential time. □

*Sabotage semantics on LTL games.* The linear temporal logic (LTL) is a logic whose formulas describe properties of infinite sequences of predicate. More formally, given a game arena $G$, a mapping $Pred$ from vertices to a set of predicate $P$ and an LTL formula $\phi$, the winning condition $\phi(Pred)$ is the set of plays $v_0 v_1 \cdots$ such that the sequence $Pred(v_0)Pred(v_1) \cdots$ satisfies $\phi$.

Solving LTL-games, with their standard semantics, is already 2-EXPTIME-complete [14]. The 2-EXPTIME membership can be obtained by turning an LTL formula $\phi$ into a parity automaton whose size is doubly exponential in the size of $\phi$, and solving the parity game obtained by taking the product of the game arena with the automaton.

When applying a sabotage semantics to an LTL game $\mathcal{G}$, we obtain a game $[\![\mathcal{G}]\!]$ of size exponential in the initial arena, and whose value function is a combination of a cost function and the LTL formula. By applying the same method as above, using the parity automaton associated with the formula and taking the product of the automaton with $[\![\mathcal{G}]\!]$, we obtain a game whose size is doubly exponential in the size of $\mathcal{G}$, and whose value function is a combination of a cost function and a parity objective. Applying the above result, one can show that this game can be solved in 2-EXPTIME with respect to the size of $\mathcal{G}$.

As the standard semantics is equivalent to a sabotage semantics with budget 0, the problem remains 2-EXPTIME-hard, and thus 2-EXPTIME-complete.