Multi-Weighted Reachability Games*

Thomas Brihaye¹ and Aline Goeminne²

 ¹ UMONS - Université de Mons, Belgium
 ² F.R.S.-FNRS & UMONS- Université de Mons, Belgium {thomas.brihaye,aline.goeminne}@umons.ac.be

Abstract. We study two-player multi-weighted reachability games played on a finite directed graph, where an agent, called \mathcal{P}_1 , has several quantitative reachability objectives that he wants to optimize against an antagonistic environment, called \mathcal{P}_2 . In this setting, we ask what cost profiles \mathcal{P}_1 can ensure regardless of the opponent's behavior. Cost profiles are compared thanks to: (i) a lexicographic order that ensures the unicity of an upper value and (ii) a componentwise order for which we consider the Pareto frontier. We synthesize (i) lexico-optimal strategies and (ii) Pareto-optimal strategies. The strategies are obtained thanks to a fixpoint algorithm which also computes the upper value in polynomial time and the Pareto frontier in exponential time. Finally, the constrained existence problem is proved in PTIME for the lexicographic order and PSPACE-complete for the componentwise order.

Keywords: two-player games on graphs; multi-weighted reachability games; Pareto-optimal strategies; lexico-optimal strategies

1 Introduction

Two-player zero-sum games played on graphs are commonly used in the endeavor to synthesize systems that are correct by construction. In the two-player zerosum setting the system wants to achieve a given objective whatever the behavior of the environment. This situation is modeled by a two-player game in which \mathcal{P}_1 (resp. \mathcal{P}_2) represents the system (resp. the environment). Each vertex of the graph is owned by one player and they take turn by moving a token from vertex to vertex by following the graph edges. This behavior leads to an infinite sequence of vertices called a play. The choice of a player's next move is dictated by its strategy. In a quantitative setting, edges are equipped with a weight function and a cost function assigns a cost to each play. This cost depends on the weights of the edges along the play. With this quantitative perspective, \mathcal{P}_1 wants to minimize the cost function. We say that \mathcal{P}_1 can ensure a cost of x if there exists a strategy of \mathcal{P}_1 such that, whatever the strategy followed by \mathcal{P}_2 , the corresponding cost is less than or equal to x. An interesting question is thus to determine what are the costs that can be ensured by \mathcal{P}_1 . In this document, these costs are called

^{*} Thomas Brihaye – Partly supported by the F.R.S.- FNRS under grant n°T.0027.21. Aline Goeminne – F.R.S.-FNRS postdoctoral researcher.

the ensured values. Other frequently studied questions are: Given a threshold x, does there exist a strategy of \mathcal{P}_1 that ensures a cost less than or equal to x? Is it possible to synthesize such a strategy, or even better, if it exists, a strategy that ensures the best ensured value, *i.e.*, an optimal strategy?

A well-known studied quantitative objective is the one of *quantitative reachability objective*. A player who wants to achieve such an objective has a subset of vertices, called *target set*, that he wants to reach as quickly as possible. In terms of edge weights, that means that he wants to minimize the cumulative weights until a vertex of the target set is reached. In this setting it is proved that the best ensured value is computed in polynomial time and that optimal strategies exist and do not require memory [8].

Considering systems with only one cost to minimize may seem too restrictive. Indeed, \mathcal{P}_1 may want to optimize different quantities while reaching his objective. Moreover, optimizing these different quantities may lead to antagonistic behaviors, for instance when a vehicle wants to reach his destination while minimizing both the delay and the energy consumption. This is the reason why in this paper, we study two-player multi-weighted reachability games, where \mathcal{P}_1 aims at reaching a target while minimizing several costs. In this setting each edge of the graph is labeled by a d-tuple of d natural numbers, one per quantity to minimize. Given a sequence of vertices in the game graph, the cost profile of \mathcal{P}_1 corresponds to the sum of the weights of the edges, component by component, until a given target set is reached. We consider the multi-dimension counterpart of the previous studied problems: we wonder what cost profiles are ensured by \mathcal{P}_1 . Thus \mathcal{P}_1 needs to arbitrate the trade-off induced by the multi-dimension setting. In order to do so, we consider two alternatives: the cost profiles can be compared either via (i) a lexicographic order that ranks the objectives a priori and leads to a unique minimal ensured value; or via (*ii*) a componentwise order. In this second situation, \mathcal{P}_1 takes his decision a posteriori while choosing an element of the Pareto frontier (the set of minimal ensured values, which is not necessarily a singleton).

Contributions. Our contributions are threefold. First, in Section 3.1, given a two-player multi-weighted reachability game, independently of the order considered, we provide a fixpoint algorithm, which computes the minimal cost profiles that can be ensured by \mathcal{P}_1 . In Section 3.2, we study the time complexity of this algorithm, depending on the order considered. When considering the lexicographic order (resp. componentwise order), the algorithm runs in polynomial time (resp. exponential time). Moreover, if the number of dimensions is fixed, the computation of the Pareto frontier can be done in pseudo-polynomial time (polynomial if the weights of the game graph are encoded in unary). As a second contribution, in Section 3.3, based on the fixpoint algorithm, we synthesize the optimal strategies (one per order considered). In particular, we show that positional strategies suffice when considering the lexicographic order, although memory is needed in the componentwise case. Finally, in Section 4, we focus on the natural decision problem associated with our model: the constrained existence problem. Given a two-player multi-weighted reachability game and a cost profile \mathbf{x} , the answer to the constrained existence problem is positive when there exists a strategy of \mathcal{P}_1 that ensures \mathbf{x} . In the lexicographic case, we show that the problem belongs to PTIME; although it turns to be PSPACE-complete in the componentwise case.

Detailed proofs of our results are provided in Appendix A and Appendix B.

Related work Up to our knowledge, and quite surprisingly, two-player multi-weighted reachability games, as defined in this paper, were not studied before. Nevertheless, a one-player variant known as multi-constrained routing is known to be NP-complete and exact and approximate algorithms are, for example, provided in [10]. The time complexity of their exact algorithm matches our results since it runs in exponential time and they indicate that it is pseudo-polynomial if d = 2. The one-player setting is also studied in timed automata [9].

If we focus on two-player settings, another closely related model to multiweighted reachability games is the one studied in [6]. The authors consider twoplayer generalized (qualitative) reachability games. In this setting \mathcal{P}_1 wants to reach several target sets in any order but does not take into account the cost of achieving that purpose. They prove that deciding the winner in such a game is PSPACE-complete. Moreover, they discuss the fact that winning strategies need memory. The memory is used in order to remember which target sets have already been reached. In our setting, we assume that there is only one target set but that the cost to reach it depends on the dimension. Memory is needed because we have to take into consideration the partial sum of weights up to now in order to make the proper choices in the future to ensure the required cost profile. Notice that if we would like to study the case where each dimension has its own target set, both types of memory would be needed.

If we consider other objectives than reachability, we can mention different works on multi-dimentional *energy* and *mean-payoff* objectives [7,2,4]. Moreover, in [1], they prove that the Pareto frontier in a multi-dimensional mean-payoff game is definable as a finite union of convex sets obtained from linear inequations. The authors also provide a Σ_2^P algorithm to decide if this set intersects a convex set defined by linear inequations.

Lexicographic preferences are used in stochastic games with lexicographic (qualitative) reachability-safety objectives [3]. The authors prove that lexicooptimal strategies exist but require finite-memory in order to know on which dimensions the corresponding objective is satisfied or not. They also provide an algorithm to compute the best ensured value and compute lexico-optimal strategies thanks to different computations of optimal strategies in single-dimensional games. Finally, they show that deciding if the best ensured value is greater than or equal to a tuple \mathbf{x} is PSPACE-hard and in NEXPTIME \cap CO-NEXPTIME.

2 Preliminaries

2.1 Two-Player Multi-Weighted Reachability Games

Weighted Arena We consider games that are played on an *(weighted) arena* by two players: \mathcal{P}_1 and \mathcal{P}_2 . An arena \mathcal{A}_d is a tuple $(V_1, V_2, E, \mathbf{w})$ where *(i)*

 $(V = V_1 \cup V_2, E)$ is a graph such that vertices V_i for $i \in \{1, 2\}$ are owned by \mathcal{P}_i and $V_1 \cap V_2 = \emptyset$ and (ii) $\mathbf{w} : E \longrightarrow \mathbb{N}^d$ is a weight function which assigns d natural numbers to each edge of the graph. The variable d is called the number of dimensions. For all $1 \leq i \leq d$, we denote by w_i , with $w_i : E \longrightarrow \mathbb{N}$, the projection of **w** on the ith component, *i.e.*, for all $e \in E$, if $\mathbf{w}(e) = (n_1, \ldots, n_d)$ then, $w_i(e) = n_i$. We define W as the largest weight that can appear in the values of the weight function, *i.e.*, $W = \max\{w_i(e) \mid 1 \le i \le d \text{ and } e \in E\}$.

Each time we consider a tuple $\mathbf{x} \in X^d$ for some set X, we write it in bold and we denote the ith component of this tuple by x_i . Moreover, we abbreviate the tuples $(0, \ldots, 0)$ and (∞, \ldots, ∞) by **0** and ∞ respectively.

Plays and Histories A play (resp. history) in \mathcal{A}_d is an infinite (resp. finite) sequence of vertices consistent with the structure of the associated arena \mathcal{A}_{d} , *i.e.*, if $\rho = \rho_0 \rho_1 \dots$ is a play then, for all $n \in \mathbb{N}$, $\rho_n \in V$ and $(\rho_n, \rho_{n+1}) \in E$. A history may be formally defined in the same way. The set of plays (resp. histories) are denoted by $\operatorname{Plays}_{\mathcal{A}_d}$ (resp. $\operatorname{Hist}_{\mathcal{A}_d}$). When the underlying arena is clear from the context we only write Plays (resp. Hist). We also denote by $Hist_1$ the set of histories which end in a vertex owned by \mathcal{P}_1 , *i.e.*, $\text{Hist}_1 = \{h = h_0 h_1 \dots h_n \mid n \}$ $h \in \text{Hist}$ and $h_n \in V_1$. For a given vertex $v \in V$, the sets Plays(v), Hist(v), $Hist_1(v)$ denote the sets of plays or histories starting in v. Finally, for a history $h = h_0 \dots h_n$, the vertex h_n is denoted by Last(h) and |h| = n is the length of h.

Multi-Weighted Reachability Games We consider multi-weighted reachability games such that \mathcal{P}_1 has a target set that he wants to reach from a given initial vertex. Moreover, crossing edges on the arena implies the increasing of the d cumulated costs for \mathcal{P}_1 . While in 1-weighted reachability game \mathcal{P}_1 aims at reaching his target set as soon as possible (minimizing his cost), in the general *d*-weighted case he wants to find a *trade-off* between the different components.

More formally, $F \subseteq V$ which is a subset of vertices that \mathcal{P}_1 wants to reach is called the *target set* of \mathcal{P}_1 . The *cost function* \mathbf{Cost} : Plays $\longrightarrow \overline{\mathbb{N}}^d$ of \mathcal{P}_1 provides, given a play ρ , the cost of \mathcal{P}_1 to reach his target set F along ρ .³ This cost corresponds to the sum of the weight of the edges, component by component, until he reaches F or is equal to ∞ for all components if it is never the case. For all $1 \leq i \leq d$, we denote by $\text{Cost}_i : \text{Plays} \longrightarrow \overline{\mathbb{N}}$, the projection of **Cost** on the ith component. Formally, for all $\rho = \rho_0 \rho_1 \ldots \in$ Plays:

$$\operatorname{Cost}_{i}(\rho) = \begin{cases} \sum_{n=0}^{\ell-1} w_{i}(\rho_{n}, \rho_{n+1}) & \text{if } \ell \text{ is the least index such that } \rho_{\ell} \in \mathbf{F} \\ \infty & \text{otherwise} \end{cases}$$

and $\mathbf{Cost}(\rho) = (\mathrm{Cost}_1(\rho), \dots, \mathrm{Cost}_d(\rho))$ is called a *cost profile*. If $h = h_0 \dots h_\ell$ is a history, $\mathbf{Cost}(h) = \sum_{n=0}^{\ell-1} \mathbf{w}(h_n, h_{n+1})$ is the accumulated costs, component by component, along the history. We assume that Cost(v) = 0, for all $v \in V$.

³ Where the following notation is used: $\overline{\mathbb{N}} = \mathbb{N} \cup \{\infty\}$

Definition 1 (Multi-weighted reachability game). Given a target set $F \subseteq V$, the tuple $\mathcal{G}_d = (\mathcal{A}_d, F, \mathbf{Cost})$ is called a *d*-weighted reachability game, or more generally a multi-weighted reachability game.

In a *d*-weighted reachability game $\mathcal{G}_{d} = (\mathcal{A}_{d}, F, \mathbf{Cost})$, an initial vertex $v_{0} \in V$ is often fixed and the game (\mathcal{G}_{d}, v_{0}) is called an *initialized multi-weighted* reachability game. A play (resp. history) of (\mathcal{G}_{d}, v_{0}) is a play (resp. history) of \mathcal{A}_{d} starting in v_{0} .

In the rest of this document, for the sake of readability we write (initialized) game instead of (initialized) *d*-weighted reachability game.



Fig. 1. Example of the arena A_2 of a game G_2 . The target set is $F = \{v_9\}$ and the weight function is given by the label of the edges. Edges without label have a weight of (1, 1). The dotted rectangle is a restriction of the arena specifically used in Example 2.

Example 1. We consider as a running example the game \mathcal{G}_2 such that its arena $\mathcal{A}_2 = (V_1, V_2, E, \mathbf{w})$ is depicted in Figure 1. In this example the set of vertices of \mathcal{P}_1 (resp. \mathcal{P}_2) are depicted by rounded (resp. rectangular) vertices and the vertices that are part of the target set are doubly circled/framed. The weight function \mathbf{w} labels the corresponding edges. We follow those conventions all along this document. Here, $V_1 = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9\}$, $V_2 = \{v_0, v_{10}\}$, $\mathbf{F} = \{v_9\}$ and, for example, $\mathbf{w}(v_0, v_2) = (2, 4)$. For all edges without label, we assume that the weight is (1, 1), *e.g.*, $\mathbf{w}(v_3, v_4) = (1, 1)$. Do not pay attention to the dotted rectangle for the moment.

Let us now study the cost profiles of two different plays. First, the play $\rho = v_0 v_1 v_4 v_6 v_9^{\omega}$ has a cost profile of $\mathbf{Cost}(\rho) = (4, 2) + (1, 1) + (4, 2) + (1, 1) = (10, 6)$ since ρ visits F in v_9 . Moreover, $\mathbf{Cost}_1(\rho) = 10$ and $\mathbf{Cost}_2(\rho) = 6$. Second, the play $\rho' = v_0 v_3 (v_5 v_8)^{\omega}$ has a cost profile of (∞, ∞) since it does not reach F.

Strategies A strategy of player $i, i \in \{1, 2\}$, provides the next action of \mathcal{P}_i . Formally, a strategy of \mathcal{P}_i from a vertex v is a function $\sigma_i : \text{Hist}_i(v) \longrightarrow V$ such that for all $h \in \text{Hist}_i(v)$, $(\text{Last}(h), \sigma_i(h)) \in E$. We denote by Σ_i^v the set of strategies of \mathcal{P}_i from $v \in V$. Notice that in an initialized game (\mathcal{G}_d, v_0) , if we do not specify something else, we assume that the strategies are defined from v_0 .

Moreover, given two strategies σ_1 of \mathcal{P}_1 and σ_2 of \mathcal{P}_2 , there is only one play which is consistent with (σ_1, σ_2) from v_0 . This play is called the *outcome* of (σ_1, σ_2) from v_0 and is denoted by $\langle \sigma_1, \sigma_2 \rangle_{v_0}$.

We differentiate two classes of strategies: *positional* strategies and *finitememory* strategies. A positional strategy σ_i only depends on the last vertex of the history, *i.e.*, for all $h, h' \in \text{Hist}_i$, if Last(h) = Last(h') then, $\sigma_i(h) = \sigma_i(h')$. It is finite-memory if it can be encoded by a finite-state machine.

Partial Orders Given two cost profiles \mathbf{x} and \mathbf{y} in $\overline{\mathbb{N}}^d$, \mathcal{P}_1 should be able to decide which one is the most beneficial to him. In order to do so, we consider two *partial orders* in the rest of this document: the *componentwise* order and the *lexicographic* order.

We recall some related definitions. A partial order on X is a binary relation $\leq \subseteq X \times X$ which is reflexive, antisymmetric and transitive. The strict partial order < associated with it is given by x < y if and only if $x \leq y$ and $x \neq y$, for all $x, y \in X$. A partial order is called a *total order* if and only if for all $x, y \in X, x \leq y$ or $y \leq x$. Given a set $X' \subseteq X$, the set of minimal elements of X' with respect to \leq is given by minimal $(X') = \{x \in X' \mid \text{ if } y \in X' \text{ and } y \leq x, \text{ then } x = y\}$. Moreover, the upward closure of X' with respect to \leq is the set $\uparrow X' = \{\mathbf{x} \in X \mid \exists \mathbf{y} \in X' \text{ st. } \mathbf{y} \leq \mathbf{x}\}$. A set X' is said upward closed if $\uparrow X' = X'$.

In what follows we consider two partial orders on $\overline{\mathbb{N}}^d$. The *lexicographic* order, denoted by \leq_{L} , is defined as follows: for all $\mathbf{x}, \mathbf{y} \in \overline{\mathbb{N}}^d$, $\mathbf{x} \leq_{\mathrm{L}} \mathbf{y}$ if and only if either $(i) x_i = y_i$ for all $i \in \{1, \ldots, d\}$ or (ii) there exists $i \in \{1, \ldots, d\}$ such that $x_i < y_i$ and for all $k < i, x_k = y_k$. The *componentwise* order, denoted by \leq_{C} , is defined as: for all $\mathbf{x}, \mathbf{y} \in \overline{\mathbb{N}}^d$, $\mathbf{x} \leq_{\mathrm{C}} \mathbf{y}$ if and only if for all $i \in \{1, \ldots, d\}$, $x_i \leq y_i$. Although the lexicographic order is a total order, the componentwise order is not.

2.2 Studied Problems

We are now able to introduce the different problems that are studied in this paper: the *ensured values problem* and the *constrained existence problem*.

Ensured Values Given a game \mathcal{G}_d and a vertex v, we define the *ensured values* from v as the cost profiles that \mathcal{P}_1 can ensure from v whatever the behavior of \mathcal{P}_2 . We denote the set of ensured values from v by $\text{Ensure}_{\leq}(v)$, *i.e.*, $\text{Ensure}_{\leq}(v) = \{\mathbf{x} \in \overline{\mathbb{N}}^d \mid \exists \sigma_1 \in \Sigma_1^v \text{ st. } \forall \sigma_2 \in \Sigma_2^v, \mathbf{Cost}(\langle \sigma_1, \sigma_2 \rangle_v) \lesssim \mathbf{x}\}$. Moreover, we say that a strategy σ_1 of \mathcal{P}_1 from v ensures the cost profile $\mathbf{x} \in \overline{\mathbb{N}}^d$ if for all strategies σ_2 of \mathcal{P}_2 from v, we have that $\mathbf{Cost}(\langle \sigma_1, \sigma_2 \rangle_v) \lesssim \mathbf{x}$.

We denote by minimal (Ensure_{$\leq (v)$}) the set of minimal elements of Ensure_{$\leq (v)$} with respect to \leq . If \leq is the lexicographic order, the set of minimal elements of Ensure_{$\leq L}(v)$ with respect to \leq_L is a singleton, as \leq_L is a total order, and is called the *upper value* from v. We denote it by Val(v). On the other hand, if \leq is the componentwise order, the set of minimal elements of Ensure_{$\leq C$}(v) with respect to \leq_C is called the *Pareto frontier* from v and is denoted by Pareto(v).</sub>

Definition 2 (Ensured Values Problems). Let (\mathcal{G}_d, v_0) be an initialized game. Depending on the partial order, we distinguish two problems: (i) computation of the upper value, $\overline{Val}(v_0)$, and (ii) computation of the Pareto frontier, $Pareto(v_0)$.

Theorem 1. Given an initialized game (\mathcal{G}_d, v_0) ,

- 1. The upper value $\overline{\text{Val}}(v_0)$ can be computed in polynomial time.
- 2. The Pareto frontier can be computed in exponential time.
- 3. If d is fixed, the Pareto frontier can be computed in pseudo-polynomial time.

Statement 1 is obtained by Theorem 3, Statements 2 and 3 are proved by Theorem 4.

A strategy σ_1 of \mathcal{P}_1 from v is said *Pareto-optimal* from v if σ_1 ensures \mathbf{x} for some $\mathbf{x} \in \text{Pareto}(v)$. If we want to explicitly specify the element \mathbf{x} of the Pareto frontier which is ensured by the Pareto-optimal strategy we say that the strategy σ_1 is \mathbf{x} -Pareto-optimal from v. Finally, a strategy σ_1 of \mathcal{P}_1 from v is said *lexico-optimal* if it ensures the only $\mathbf{x} \in \overline{\text{Val}}(v)$.

In Section 3.3, we show how to obtain (i) a **x**-Pareto-optimal strategy from v_0 for each $\mathbf{x} \in \text{Pareto}(v_0)$ and (ii) a lexico-optimal strategy from v_0 which is positional. Notice that, as in Example 2, Pareto-optimal strategies sometimes require finite-memory.

Constrained Existence We are also interested in deciding, given a cost profile \mathbf{x} , whether there exists a strategy σ_1 of \mathcal{P}_1 from v_0 that ensures \mathbf{x} . We call this decision problem the *constrained existence problem* (CE problem).

Definition 3 (Constrained Existence Problem – **CE Problem).** Given an initialized game (\mathcal{G}_d , v_0) and $\mathbf{x} \in \mathbb{N}^d$, does there exist a strategy $\sigma_1 \in \Sigma_1^{v_0}$ such that for all strategies $\sigma_2 \in \Sigma_2^{v_0}$, $\mathbf{Cost}(\langle \sigma_1, \sigma_2 \rangle_{v_0}) \lesssim \mathbf{x}$?

The complexity results of this problem are summarized in the following theorem which is restated and discussed in Section 4.

Theorem 2. If \leq is the lexicographic order, the CE problem is solved in PTIME. If \leq is the componentwise order, the CE problem is PSPACE-complete.

We conclude this section by showing that memory may be required by \mathcal{P}_1 in order to ensure a given cost profile.

Example 2. We consider the game such that its arena is a restriction of the arena given in Figure 1. This restricted arena is inside the dotted rectangle. For clarity, we assume that the arena is only composed by vertices $v_0, v_1, v_2, v_4, v_6, v_7$ and v_9 and their associated edges. We prove that with the componentwise order $\leq_{\rm C}$, memory for \mathcal{P}_1 is required to ensure the cost profile (8,8). There are only two positional strategies of \mathcal{P}_1 : σ_1 defined such that $\sigma_1(v_4) = v_6$ and τ_1 defined such that $\tau_1(v_4) = v_7$. For all the other vertices, \mathcal{P}_1 has no choice. With σ_1 , if \mathcal{P}_2 chooses v_1 from v_0 , the resulting cost profile is (10,6). In the same way, with

 τ_1 , if \mathcal{P}_2 chooses v_2 from v_0 , the resulting cost profile is (6, 10). This proves that \mathcal{P}_1 cannot ensure (8, 8) from v_0 with a positional strategy. This is nevertheless possible if \mathcal{P}_1 plays a finite-memory strategy. Indeed, by taking into account the past choice of \mathcal{P}_2 , \mathcal{P}_1 is able to ensure (8, 8): if \mathcal{P}_2 chooses v_1 (resp. v_2) from v_0 then, \mathcal{P}_1 should choose v_7 (resp. v_6) from v_4 resulting in a cost profile of (8, 8) in both cases.

3 Ensured Values

This section is devoted to the computation of the sets $\min(\operatorname{Ensure}_{\leq}(v))$ for all $v \in V$. In Section 3.1, we provide a fixpoint algorithm which computes these sets. In Section 3.2, we study the time complexity of the algorithm both for the lexicographic and the componentwise orders. Finally, in Section 3.3, we synthesize lexico and Pareto-optimal strategies.

3.1 Fixpoint Algorithm

Our algorithm that computes the sets minimal(Ensure_{\leq}(v)) for all $v \in V$ shares the key idea of some classical shortest path algorithms. First, for each $v \in V$, we compute the set of cost profiles that \mathcal{P}_1 ensures from v in k steps. Then, once all these sets are computed, we compute the sets of cost profiles that can be ensured by \mathcal{P}_1 from each vertex but in k + 1 steps. And so on, until the sets of cost profiles are no longer updated, meaning that we have reached a fixpoint.

For each $k \in \mathbb{N}$ and each $v \in V$, we define the set $\operatorname{Ensure}^{k}(v)$ as the set of cost profiles that can be ensured by \mathcal{P}_{1} within k steps. Formally, $\operatorname{Ensure}^{k}(v) = \{\mathbf{x} \in \overline{\mathbb{N}}^{d} \mid \exists \sigma_{1} \in \Sigma_{1}^{v} \text{ st. } \forall \sigma_{2} \in \Sigma_{2}^{v}, \operatorname{Cost}(\langle \sigma_{1}, \sigma_{2} \rangle_{v}) \lesssim \mathbf{x} \land |\langle \sigma_{1}, \sigma_{2} \rangle_{v}|_{\mathrm{F}} \leq k\}^{4},$ where for all $\rho = \rho_{0}\rho_{1} \ldots \in \operatorname{Plays}, |\rho|_{\mathrm{F}} = k$ if k is the least index such that $\rho_{k} \in \mathrm{F}$ and $|\rho|_{\mathrm{F}} = -\infty$ otherwise.

Notice that the sets $\text{Ensure}^{k}(v)$ are upward closed and that they are infinite sets except if $\text{Ensure}^{k}(v) = \{\infty\}$. This is the reason why, in the algorithm, we only store sets of minimal elements denoted by $I^{k}(v)$. Thus, the correctness of the algorithm relies on the property that for all $k \in \mathbb{N}$ and all $v \in V$, minimal(Ensure^k(v)) = $I^{k}(v)$.

The fixpoint algorithm is provided by Algorithm 1 in which, if X is a set of cost profiles, and $v, v' \in V$, $X + \mathbf{w}(v, v') = {\mathbf{x} + \mathbf{w}(v, v') | \mathbf{x} \in X}$. For the moment, do not pay attention to Lines 10 to 13, we come back to them later.

Example 3. We now explain how the fixpoint algorithm runs on Example 1. Table 1 represents the fixpoint of the fixpoint algorithm both for the lexicographic and componentwise orders. Table 2 in Appendix A.1 provides all steps of the algorithm. Remark that the fixpoint is reached with $k^* = 4$, while the algorithm takes one more step in order to check that $I^4(v) = I^5(v)$ for all $v \in V$. We only focus on some relevant steps of the algorithm with the componentwise order $\leq_{\rm C}$.

 $^{^4}$ To lighten the notations, we omit the mention of \lesssim in subscript.

9

Algorithm 1: Fixpoint algorithm

1 for $v \in F$ do $I^0(v) = \{0\}$
2 for $v \notin F$ do $I^0(v) = \{\infty\}$
3
4 repeat
5 for $v \in V$ do
6 if $v \in F$ then $I^{k+1}(v) = \{0\}$
7
8 else if $v \in V_1$ then
9 $I^{k+1}(v) = \min\left(\bigcup_{v' \in \operatorname{Succ}(v)} \uparrow I^k(v') + \mathbf{w}(v,v')\right)$
10 for $\mathbf{x} \in \mathbf{I}^{k+1}(v)$ do
11 if $\mathbf{x} \in \mathbf{I}^k(v)$ then $f_v^{k+1}(\mathbf{x}) = f_v^k(\mathbf{x})$
12 else
13 $f_v^{k+1}(\mathbf{x}) = (v', \mathbf{x}') \text{ where } v' \text{ and } \mathbf{x}' \text{ are such that} \\ v' \in \operatorname{Succ}(v), \mathbf{x} = \mathbf{x}' + \mathbf{w}(v, v') \text{ and } \mathbf{x}' \in \operatorname{I}^k(v')$
14
15 else if $v \in V_2$ then
16 $I^{k+1}(v) = \min\left(\bigcap_{v' \in \operatorname{Succ}(v)} \uparrow I^k(v') + \mathbf{w}(v,v')\right)$
17 until $I^{k+1}(v) = I^k(v)$ for all $v \in V$

	<2	v_0	v_1, v_2	v_3	v_4	v_5	v_{6}, v_{7}, v_{10}	v_8	v_9
$I^*(\cdot)$	$\leq_{\rm L}$	$\{(8,8)\}$	$\{(4,6)\}$	$\{(4,4)\}$	$\{(3,5)\}$	$\{(3,3)\}$	$\{(1,1)\}$	$\{(2,2)\}$	$\{(0,0)\}$
	$\leq_{\rm C}$	$\{(8,8)\}$	$\{(6,4),(4,6)\}$	$\{(4,4)\}$	$\{(5,3),(3,5)\}$	$\{(3,3)\}$	$\{(1,1)\}$	$\{(2,2)\}$	$\{(0,0)\}$
Table 1 Fixpoint of the fixpoint algorithm reached at stop $k^* = 4$									

Table 1. Fixpoint of the fixpoint algorithm reached at step $k^* = 4$.

Let us first assume that the first step is computed and is such that $I^{1}(v_{9}) =$ $\{(0,0)\}$ since $v_9 \in F$, $I^1(v) = \{(1,1)\}$ if $v \in \{v_6, v_7, v_{10}\}$ and $I^1(v) = \{(\infty, \infty)\}$ for all other vertices. We now focus on the computation of $I^2(v_4)$. By Algorithm 1, $I^{2}(v_{4}) = \min(1 + I^{1}(v_{6}) + (4, 2)) + I^{1}(v_{7}) + (2, 4)) = \min(1 + \{(5, 3)\}) + 1$ $\{(3,5)\} = \{(5,3), (3,5)\}.$

We now assume: $I^{3}(v_{0}) = \{(\infty, \infty)\}, I^{3}(v_{1}) = I^{3}(v_{2}) = I^{3}(v_{3}) = \{(4, 6), (6, 4)\},\$ $I^{3}(v_{4}) = \{(5,3), (3,5)\}$ and $I^{3}(v_{5}) = \{(3,3)\}$. We compute $I^{4}(v_{0})$ which is equal to minimal $(\uparrow \{(4, 6), (6, 4)\} + (4, 2) \cap \uparrow \{(4, 6), (6, 4)\} + (2, 4) \cap \uparrow \{(4, 6), (6, 4)\} + (4, 6), (6, 4)\}$ (1,1) = minimal(\uparrow {(8,8), (10,6)} \cap \uparrow {(6,10), (8,8)} \cap \uparrow {(5,7), (7,5)} = minimal $(\uparrow \{(8,8)\} \cap \uparrow \{(5,7), (7,5)\}) = \{(8,8)\}$. Finally, we compute $I^4(v_3) =$ $\min(\uparrow \{(6,4), (4,6)\} \cup \uparrow \{(4,4)\}) = \min(\{(6,4), (4,6), (4,4)\}) = \{(4,4)\}.$

Termination We focus on the termination of the fixpoint algorithm.

Proposition 1. The fixpoint algorithm terminates in less than |V| + 1 steps.

The proof of this proposition relies on Propositions 2 and 3. Proposition 2 is interesting on its own. It states that if there exists a strategy σ_1 of \mathcal{P}_1 which ensures a cost profile $\mathbf{x} \in \mathbb{N}^d$ from $v \in V$ then, there exists another strategy σ'_1 of \mathcal{P}_1 which also ensures \mathbf{x} from v but such that the number of edges between v and the first occurrence of a vertex in F is less than or equal to |V|, and this regardless of the behavior of \mathcal{P}_2 .

Proposition 2. Given a game \mathcal{G}_d , a vertex $v \in V$ and a cost profile $\mathbf{x} \in \mathbb{N}^d$, if there exists a strategy σ_1 of \mathcal{P}_1 such that for all strategies σ_2 of \mathcal{P}_2 we have that $\mathbf{Cost}(\langle \sigma_1, \sigma_2 \rangle_v) \lesssim \mathbf{x}$ then, there exists σ'_1 of \mathcal{P}_1 such that for all σ_2 of \mathcal{P}_2 we have: (i) $\mathbf{Cost}(\langle \sigma'_1, \sigma_2 \rangle_v) \lesssim \mathbf{x}$ and (ii) $|\langle \sigma'_1, \sigma_2 \rangle_v|_{\mathbf{F}} \leq |V|$.

The strategy σ'_1 is obtained by adequately removing cycles formed by the strategy σ_1 . Let us point out that Proposition 2 does not imply that σ'_1 is positional. Indeed, in Example 2, the finite-memory strategy is the only strategy that ensures the cost profile (8, 8), it satisfies conditions (i) and (ii) of Proposition 2 but requires memory.

Proposition 3. We have: (i) for all $k \in \mathbb{N}$ and for all $v \in V$, $\text{Ensure}^{k}(v) \subseteq \text{Ensure}^{k+1}(v)$; and (ii) there exists $k^* \leq |V|$, such that for all $v \in V$ and for all $\ell \in \mathbb{N}$, $\text{Ensure}^{k^*+\ell}(v) = \text{Ensure}^{k^*}(v)$.

Properties stated in Proposition 3 hold by definition of $\text{Ensure}^{k}(v)$ and Proposition 2. Moreover, the step k^* is a particular step of the algorithm that we call the *fixpoint* of the algorithm. Notice that even if the fixpoint is reached at step k^* , the algorithm needs one more step in order to check that the fixpoint is reached. In the remaining part of this document, we write $\text{Ensure}^{*}(v)$ (resp. $I^{*}(v)$) instead of $\text{Ensure}^{k^*}(v)$ (resp. $I^{k^*}(v)$).

Correctness The fixpoint algorithm (Algorithm 1) exactly computes the sets $\min(\operatorname{Ensure}_{\leq}(v))$ for all $v \in V$, *i.e.*, for all $v \in V$, $\min(\operatorname{Ensure}_{\leq}(v)) = I^*(v)$. This is a direct consequence of Proposition 4.

Proposition 4. For all $k \in \mathbb{N}$ and all $v \in V$, minimal(Ensure^k(v)) = I^k(v).

3.2 Time Complexity

In this section we provide the time complexity of the fixpoint algorithm. The algorithm runs in polynomial time for the lexicographic order and in exponential time for the componentwise order. In this latter case, if d is fixed, the algorithm is pseudo-polynomial, *i.e.*, polynomial if the weights are encoded in unary.

Theorem 3. If \leq is the lexicographic order, the fixpoint algorithm runs in time polynomial in |V| and d.

Theorem 4. If \leq is the componentwise order, the fixpoint algorithm runs in time polynomial in W and |V| and exponential in d.

Theorem 3 relies on the fact that Line 9 and Line 16 can be performed in polynomial time. Indeed, in the lexicographic case, for all $k \in \mathbb{N}$ and all $v \in V$, $I^k(v)$ is a singleton. Thus these operations amounts to computing a minimum or a maximum between at most |V| values. Theorem 4 can be obtained thanks to representations of upward closed sets and operations on them provided in [5].

3.3 Synthesis of Lexico-optimal and Pareto-optimal Strategies

To this point, we have only explained the computation of the ensured values and we have not yet explained how lexico and Pareto-optimal strategies are recovered from the algorithm. This is the reason of the presence of Lines 10 to 13 in Algorithm 1. Notice that in Line 13, we are allowed to assume that \mathbf{x}' is in $\mathbf{I}^{k}(v')$ instead of $\uparrow \mathbf{I}^{k}(v')$ because for all $k \in \mathbb{N}$, for all $v \in V_1 \setminus \mathbf{F}$, $\mathbf{I}^{k+1}(v) = \min(\bigcup_{v' \in \operatorname{Succ}(v)} \mathbf{I}^{k}(v) + \mathbf{w}(v, v'))$.

Roughly speaking, the idea behind the functions f_v^k is the following. At each step $k \ge 1$ of the algorithm and for all vertices $v \in V_1 \setminus F$, we have computed the set $I^k(v)$. At that point, we know that given $\mathbf{x} \in I^k(v)$, \mathcal{P}_1 can ensure a cost profile of \mathbf{x} from v in at most k steps. The role of the function f_v^k is to keep in memory which next vertex, $v' \in \operatorname{Succ}(v)$, \mathcal{P}_1 should choose and what is the cost profile $\mathbf{x}' = \mathbf{x} - \mathbf{w}(v, v')$ which is ensured from v' in at most k - 1 steps. If different such successors exist one of them is chosen arbitrarily.

In other words, f_v^k provides information about how \mathcal{P}_1 should behave locally in v if he wants to ensure one of the cost profile $\mathbf{x} \in \mathbf{I}^k(v)$ from v in at most ksteps. In this section, we explain how, from this local information, we recover a global strategy which is \mathbf{x} -Pareto optimal from v (resp. lexico-optimal from v) for some $v \in V$ and some $\mathbf{x} \in \mathbf{I}^*(v) \setminus \{\infty\}$, if \leq is the componentwise order (resp. the lexicographic order).

We introduce some additional notations. Since for all $k \in \mathbb{N}$ and all $v \in V$, $f_v^k : \mathbf{I}^k(v) \longrightarrow V \times \overline{\mathbb{N}}^d$, if $(v', \mathbf{x}') = f_v^k(\mathbf{x})$ for some $\mathbf{x} \in \mathbf{I}^k(v)$ then, we write $f_v^k(\mathbf{x})[1] = v'$ and $f_v^k(\mathbf{x})[2] = \mathbf{x}'$. Moreover, for all $v \in V$, we write f_v^* instead of f_v^{k*} . Finally, if X is a set of cost profiles, $\min_{\leq_{\mathrm{L}}}(X) = \{\mathbf{x} \in X \mid \forall \mathbf{y} \in X, (\mathbf{y} \leq_{\mathrm{L}} \mathbf{x} \implies \mathbf{y} = \mathbf{x})\}.$

For all $u \in V$ and all $\mathbf{c} \in \mathbf{I}^*(u) \setminus \{\infty\}$, we define a strategy $\sigma_1^* \in \Sigma_1^u$. The aim of this strategy is to ensure \mathbf{c} from u by exploiting the functions f_v^* . The intuition is as follows. If the past history is hv with $v \in V_1$, \mathcal{P}_1 has to take into account the accumulated partial costs $\mathbf{Cost}(hv)$ up to v in order the make adequately his next choice to ensure \mathbf{c} at the end of the play. For this reason, he selects some $\mathbf{x} \in \mathbf{I}^*(v)$ such that $\mathbf{x} \leq \mathbf{c} - \mathbf{Cost}(hv)$ and follows the next vertex dictated by $f_v^*(\mathbf{x})[1]$.

Definition 4. Given $u \in V$ and $\mathbf{c} \in I^*(u) \setminus \{\infty\}$, we define a strategy $\sigma_1^* \in \Sigma_1^u$ such that for all $hv \in \operatorname{Hist}_1(u)$, let $\mathcal{C}(hv) = \{\mathbf{x}' \in I^*(v) \mid \mathbf{x}' \lesssim \mathbf{c} - \operatorname{Cost}(hv) \land$

11

 \mathbf{x}'

$$\leq_{\mathrm{L}} \mathbf{c} - \mathbf{Cost}(hv)\},\$$

$$\sigma_{1}^{*}(hv) = \begin{cases} v' & \text{for some } v' \in \mathrm{Succ}(v), \text{ if } \mathcal{C}(hv) = \emptyset\\ f_{v}^{*}(\mathbf{x})[1] & \text{where } \mathbf{x} = \min_{\leq_{\mathrm{L}}} \mathcal{C}(hv), \text{ if } \mathcal{C}(hv) \neq \emptyset \end{cases}.$$

Remark 1. For some technical issues, when we have to select a representative in a set of incomparable elements, the $\leq_{\rm L}$ order is used in the definitions of $\mathcal{C}(hv)$ and of the strategy. Nevertheless, Definition 4 holds both for the lexicographic and the componentwise orders.

For all $u \in V$ and $\mathbf{c} \in I^*(u) \setminus \{\infty\}$, the strategy σ_1^* defined in Definition 4 ensures \mathbf{c} from u. In particular, σ_1^* is lexico-optimal and \mathbf{c} -Pareto-optimal from u.

Theorem 5. Given $u \in V$ and $\mathbf{c} \in I^*(u) \setminus \{\infty\}$, the strategy $\sigma_1^* \in \Sigma_1^u$ defined in Definition 4 is such that for all $\sigma_2 \in \Sigma_2^u$, $\mathbf{Cost}(\langle \sigma_1^*, \sigma_2 \rangle_u) \lesssim \mathbf{c}$.

Although the strategy defined in Definition 4 is a lexico-optimal strategy from u, it requires finite-memory. However, for the lexicographic order, positional strategies are sufficient.

Proposition 5. If \leq is the lexicographic order, for $u \in V$ and $\mathbf{c} \in I^*(u) \setminus \{\infty\}$, the strategy ϑ_1^* defined as: for all $hv \in \text{Hist}_1(u), \ \vartheta_1^*(hv) = f_v^*(\mathbf{x})[1]$ where \mathbf{x} is the unique cost profile in $I^*(v)$, is a positional lexico-optimal strategy from u.

4 Constrained Existence

Finally, we focus on the constrained existence problem (CE problem).

Theorem 6. If \leq is the lexicographic order, the CE problem is solved in PTIME.

Theorem 6 is immediate since, in the lexicographic case, we can compute the upper value $\overline{\text{Val}}(v_0)$ in polynomial time (Theorem 3).

Theorem 7. If \leq is the componentwise order, the CE problem is PSPACEcomplete.

PSpace-easiness. Proposition 2 allows us to prove that the CE problem with the componentwise order is in APTIME. The alternating Turing machine works as follows: all vertices of the game owned by \mathcal{P}_1 (resp. \mathcal{P}_2) correspond to disjunctive states (resp. conjunctive states). A path of length |V| is accepted if and only if, (i) the target set is reached along that path and (ii) the sum of the weights until an element of the target set is $\leq_C \mathbf{x}$. If such a path exists, there exists a strategy of \mathcal{P}_1 that ensures the cost profile \mathbf{x} . This procedure is done in polynomial time and since APTIME = PSPACE, we get the result.

PSpace-hardness. The hardness part of Theorem 7 is based on a polynomial reduction from the QUANTIFIED SUBSET-SUM problem, proved PSPACE-complete [12, Lemma 4]. This problem is defined as follows. Given a set of natural numbers $N = \{a_1, \ldots, a_n\}$ and a threshold $T \in \mathbb{N}$, we ask if the formula $\Psi = \exists x_1 \in \{0, 1\} \forall x_2 \in \{0, 1\} \exists x_3 \in \{0, 1\} \ldots \exists x_n \in \{0, 1\}, \sum_{1 \leq i \leq n} x_i a_i = T$ is true.

13

References

- Brenguier, R., Raskin, J.: Pareto curves of multidimensional mean-payoff games. In: Kroening, D., Pasareanu, C.S. (eds.) Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9207, pp. 251–267. Springer (2015). https://doi.org/10.1007/978-3-319-21668-3_15, https://doi.org/10.1007/978-3-319-21668-3_15
- Chatterjee, K., Doyen, L., Henzinger, T.A., Raskin, J.: Generalized meanpayoff and energy games. In: Lodaya, K., Mahajan, M. (eds.) IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India. LIPIcs, vol. 8, pp. 505–516. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2010). https://doi.org/10.4230/LIPIcs.FSTTCS.2010.505, https://doi.org/10.4230/LIPIcs.FSTTCS.2010.505
- Chatterjee, K., Katoen, J., Weininger, M., Winkler, T.: Stochastic games with lexicographic reachability-safety objectives. In: Lahiri, S.K., Wang, C. (eds.) Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12225, pp. 398-420. Springer (2020). https://doi.org/10.1007/978-3-030-53291-8_21, https://doi.org/10.1007/978-3-030-53291-8_21
- Chatterjee, K., Randour, M., Raskin, J.: Strategy synthesis for multidimensional quantitative objectives. In: Koutny, M., Ulidowski, I. (eds.) CONCUR 2012 - Concurrency Theory - 23rd International Conference, CONCUR 2012, Newcastle upon Tyne, UK, September 4-7, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7454, pp. 115–131. Springer (2012). https://doi.org/10.1007/978-3-642-32940-1_10, https://doi.org/10.1007/978-3-642-32940-1_10
- Delzanno, G., Raskin, J.: Symbolic representation of upward-closed sets. In: Graf, S., Schwartzbach, M.I. (eds.) Tools and Algorithms for Construction and Analysis of Systems, 6th International Conference, TACAS 2000, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS 2000, Berlin, Germany, March 25 - April 2, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1785, pp. 426–440. Springer (2000). https://doi.org/10.1007/3-540-46419-0_29, https://doi.org/10.1007/3-540-46419-0_29
- Fijalkow, N., Horn, F.: Les jeux d'accessibilité généralisée. Tech. Sci. Informatiques **32**(9-10), 931–949 (2013). https://doi.org/10.3166/tsi.32.931-949, https://doi.org/10.3166/tsi.32.931-949
- 7. Juhl, L., Larsen, K.G., Raskin, J.: Optimal bounds for multiweighted and parametrised energy games. In: Liu, Z., Woodcock, J., Zhu, H. (eds.) Theories of Programming and Formal Methods - Essays Dedicated to Jifeng He on the Occasion of His 70th Birthday. Lecture Notes in Computer Science, vol. 8051, pp. 244–255. Springer (2013). https://doi.org/10.1007/978-3-642-39698-4_15, https://doi.org/10.1007/978-3-642-39698-4_15
- Laroussinie, F., Markey, N., Oreiby, G.: Model-checking timed ATL for durational concurrent game structures. In: Asarin, E., Bouyer, P. (eds.) Formal Modeling and Analysis of Timed Systems, 4th International Conference, FORMATS 2006, Paris, France, September 25-27, 2006, Proceedings. Lecture Notes in Computer Science,

vol. 4202, pp. 245–259. Springer (2006). https://doi.org/10.1007/11867340_18, https://doi.org/10.1007/11867340_18

- Larsen, K.G., Rasmussen, J.I.: Optimal conditional reachability for multipriced timed automata. In: Sassone, V. (ed.) Foundations of Software Science and Computational Structures, 8th International Conference, FOS-SACS 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3441, pp. 234–249. Springer (2005). https://doi.org/10.1007/978-3-540-31982-5_15, https://doi.org/10.1007/978-3-540-31982-5_15
- Puri, A., Tripakis, S.: Algorithms for the multi-constrained routing problem. In: Penttonen, M., Schmidt, E.M. (eds.) Algorithm Theory - SWAT 2002, 8th Scandinavian Workshop on Algorithm Theory, Turku, Finland, July 3-5, 2002 Proceedings. Lecture Notes in Computer Science, vol. 2368, pp. 338–347. Springer (2002). https://doi.org/10.1007/3-540-45471-3_35, https://doi.org/10.1007/3-540-45471-3_35
- 11. Sipser, M.: Introduction to the Theory of Computation. Cengage Learning (2012)
- 12. Travers, S.D.: The complexity membership of problems for integers. $\operatorname{Sci.}$ circuits sets Theor. Comput. 369(1over of 211 - 2293),(2006).https://doi.org/10.1016/j.tcs.2006.08.017, https://doi.org/10.1016/j.tcs.2006.08.017

A Additional Content of Section 3: Ensured Values

A.1 Fixpoint Algorithm

All the steps of the fixpoint algorithm performed on Example 1 are given in Table 2 both for the componentwise order $\leq_{\rm C}$ and the lexicographic order $\leq_{\rm L}$. The sets in color are sets that are updated with respect to the previous step.

Termination

Proposition 2. Given a game \mathcal{G}_d , a vertex $v \in V$ and a cost profile $\mathbf{x} \in \mathbb{N}^d$, if there exists a strategy σ_1 of \mathcal{P}_1 such that for all strategies σ_2 of \mathcal{P}_2 we have that $\mathbf{Cost}(\langle \sigma_1, \sigma_2 \rangle_v) \lesssim \mathbf{x}$ then, there exists σ'_1 of \mathcal{P}_1 such that for all σ_2 of \mathcal{P}_2 we have: (i) $\mathbf{Cost}(\langle \sigma'_1, \sigma_2 \rangle_v) \lesssim \mathbf{x}$ and (ii) $|\langle \sigma'_1, \sigma_2 \rangle_v|_{\mathbf{F}} \leq |V|$.

The proof of Proposition 2 relies on the notion of strategy tree that we introduce hereunder and that is also useful in Section A.3.

Strategy tree Given a game \mathcal{G}_d , \mathcal{T} is a tree rooted at v for some $v \in V$ if (i) \mathcal{T} is a subset of non-empty histories of \mathcal{G}_d , *i.e.*, $\mathcal{T} \subseteq \text{Hist}(v)$, (ii) $v \in \mathcal{T}$ and (iii) if $tu \in \mathcal{T}$ then, $t \in \mathcal{T}$. All $t \in \mathcal{T}$ are called *nodes* of the tree and the particular node v is called the *root* of the tree. As for histories in a game, for all $tu \in \mathcal{T}$, Last(tu) = u. The *depth* of a node $t \in \mathcal{T}$, written depth(t), is equal to |t| and its *height*, denoted by height(t), is given by $\sup\{|\text{Last}(t)t'| \mid t' \in V^* \text{ and } tt' \in \mathcal{T}\}$. The height of the tree corresponds to the height of its root. A node $t \in \mathcal{T}$ is

	\lesssim	v_0	v_1	v_2	v_3	v_4	v_5
$\mathrm{I}^{0}(\cdot)$	$\leq_{\rm C}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$
	$\leq_{\rm L}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$
$I^1(\cdot)$	$\leq_{\rm C}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$
	$\leq_{\rm L}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$
$I^2(\cdot)$	$\leq_{\rm C}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(5,3),(3,5)\}$	$\{(\infty,\infty)\}$
	$\leq_{\rm L}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(\infty,\infty)\}$	$\{(3,5)\}$	$\{(\infty,\infty)\}$
$\mathrm{I}^3(\cdot)$	$\leq_{\rm C}$	$\{(\infty,\infty)\}$	$\{(6,4),(4,6)\}$	$\{(6,4),(4,6)\}$	$\{(6,4),(4,6)\}$	$\{(5,3),(3,5)\}$	$\{(3,3)\}$
	$\leq_{\rm L}$	$\{(\infty,\infty)\}$	$\{(4,6)\}$	$\{(4,6)\}$	$\{(4,6)\}$	$\{(3,5)\}$	$\{(3,3)\}$
$I^4(\cdot) = I^*(\cdot)$	$\leq_{\rm C}$	$\{(8,8)\}$	$\{(6,4),(4,6)\}$	$\{(6,4),(4,6)\}$	$\{(4,4)\}$	$\{(5,3),(3,5)\}$	$\{(3,3)\}$
	$\leq_{\rm L}$	$\{(8,8)\}$	$\{(4,6)\}$	$\{(4,6)\}$	$\{(4,4)\}$	$\{(3,5)\}$	$\{(3,3)\}$
	<>	v_6	v_7	v_8	v_9	v_{10}	
$I^0(\cdot)$	$\stackrel{<}{\sim} \leq_{\rm C}$	$\frac{v_6}{\{(\infty,\infty)\}}$	$\frac{v_7}{\{(\infty,\infty)\}}$	$\frac{v_8}{\{(\infty,\infty)\}}$	$v_9 \\ \{(0,0)\}$	$\frac{v_{10}}{\{(\infty,\infty)\}}$	
$I^0(\cdot)$	$\stackrel{<}{\underset{\leq C}{\leq}}$	$\frac{v_6}{\{(\infty,\infty)\}}$ $\frac{\{(\infty,\infty)\}}{\{(\infty,\infty)\}}$	$\frac{v_7}{\{(\infty,\infty)\}}$ $\frac{\{(\infty,\infty)\}}{\{(\infty,\infty)\}}$	$\frac{v_8}{\{(\infty,\infty)\}}$ $\frac{\{(\infty,\infty)\}}{\{(\infty,\infty)\}}$	$ v_9 \\ {(0,0)} \\ {(0,0)} $	$\frac{v_{10}}{\{(\infty,\infty)\}}$ $\frac{\{(\infty,\infty)\}}{\{(\infty,\infty)\}}$	
$I^{0}(\cdot)$ $I^{1}(\cdot)$	$\stackrel{<}{_\sim}$ $\leq_{\rm C}$ $\leq_{\rm C}$	$ \begin{array}{c} v_6 \\ \{(\infty, \infty)\} \\ \{(\infty, \infty)\} \\ \{(1, 1)\} \end{array} $			$\frac{v_9}{\{(0,0)\}}$ $\{(0,0)\}$ $\{(0,0)\}$	$ \begin{array}{c} v_{10} \\ \hline \{(\infty, \infty)\} \\ \hline \{(\infty, \infty)\} \\ \hline \{(1, 1)\} \end{array} $	
$\mathrm{I}^{0}(\cdot)$ $\mathrm{I}^{1}(\cdot)$	$ \stackrel{<}{\underset{\simeq}{\leq}_{\rm L}} \\ \stackrel{\leq_{\rm L}}{\leq_{\rm L}} \\ \stackrel{\leq_{\rm L}}{\leq_{\rm L}} $	$ \begin{array}{c} v_6 \\ \{(\infty, \infty)\} \\ \{(\infty, \infty)\} \\ \{(1, 1)\} \\ \{(1, 1)\} \end{array} $	$ \begin{array}{c} v_7 \\ \hline \{(\infty, \infty)\} \\ \{(\infty, \infty)\} \\ \hline \{(1, 1)\} \\ \{(1, 1)\} \\ \hline \{(1, 1)\} \end{array} $	$ \begin{array}{c} v_8 \\ \overline{\{(\infty, \infty)\}} \\ \overline{\{(\infty, \infty)\}} \\ \overline{\{(\infty, \infty)\}} \\ \overline{\{(\infty, \infty)\}} \end{array} $	$\begin{array}{c} v_9 \\ \hline \{(0,0)\} \\ \{(0,0)\} \\ \hline \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \end{array}$	$ \begin{array}{c} v_{10} \\ \hline \{(\infty, \infty)\} \\ \{(\infty, \infty)\} \\ \hline \{(1, 1)\} \\ \{(1, 1)\} \\ \hline \{(1, 1)\} \end{array} $	
$I^{0}(\cdot)$ $I^{1}(\cdot)$ $I^{2}(\cdot)$	$\begin{array}{c} \leq \\ \\ \leq \\ \\ \leq \\$	$ \begin{array}{c} v_6 \\ \{(\infty, \infty)\} \\ \{(\infty, \infty)\} \\ \{(1, 1)\} \\ \{(1, 1)\} \\ \{(1, 1)\} \\ \{(1, 1)\} \end{array} $	$ \begin{array}{c} v_7 \\ \hline \{(\infty, \infty)\} \\ \{(\infty, \infty)\} \\ \hline \{(1, 1)\} \\ \hline \{(1, 1)\} \\ \{(1, 1)\} \\ \hline \{(1, 1)\} \end{array} $	$ \begin{array}{c} v_8 \\ \{(\infty, \infty)\} \\ \{(\infty, \infty)\} \\ \{(\infty, \infty)\} \\ \{(\infty, \infty)\} \\ \{(2, 2)\} \end{array} $	$\begin{array}{c} v_9 \\ \hline \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \end{array}$	$\begin{array}{c} v_{10} \\ \hline \{(\infty,\infty)\} \\ \{(\infty,\infty)\} \\ \hline \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \end{array}$	
$I^{0}(\cdot)$ $I^{1}(\cdot)$ $I^{2}(\cdot)$	$ \begin{array}{c} < \\ \leq \\$	$\begin{array}{c} v_6 \\ \{(\infty, \infty)\} \\ \{(\infty, \infty)\} \\ \{(1, 1)\} \\ \{(1, 1)\} \\ \{(1, 1)\} \\ \{(1, 1)\} \\ \{(1, 1)\} \end{array}$	$\begin{array}{c} v_7 \\ \hline \{(\infty,\infty)\} \\ \{(\infty,\infty)\} \\ \hline \{(1,1)\} \\ \{(1,1)\} \\ \hline \{(1,1)\} \\ \{(1,1)\} \\ \hline \{(1,1)\} \\ \hline \{(1,1)\} \end{array}$		$\begin{array}{c} v_9 \\ \hline \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \end{array}$	$\begin{array}{c} v_{10} \\ \hline \{(\infty,\infty)\} \\ \hline \{(\infty,\infty)\} \\ \hline \{(1,1)\} \end{array}$	
$\begin{array}{c} I^{0}(\cdot) \\ I^{1}(\cdot) \\ I^{2}(\cdot) \\ I^{3}(\cdot) \end{array}$	$\stackrel{<}{\scriptstyle \sim} \stackrel{<}{\scriptstyle \sim} \stackrel{~}{\scriptstyle \sim} \stackrel{~}$	$\begin{array}{c} v_6 \\ \hline \{(\infty,\infty)\} \\ \{(\infty,\infty)\} \\ \hline \{(1,1)\} \\ \hline \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$	$\begin{array}{c} v_7 \\ \hline \{(\infty, \infty)\} \\ \{(\infty, \infty)\} \\ \{(1, 1)\} \\ \{(1, 1)\} \\ \{(1, 1)\} \\ \{(1, 1)\} \\ \{(1, 1)\} \\ \{(1, 1)\} \end{array}$		$\begin{array}{c} v_9 \\ \hline \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \end{array}$	$\begin{array}{c} \hline v_{10} \\ \hline \{(\infty,\infty)\} \\ \{(\infty,\infty)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \end{array}$	
$\begin{array}{c} I^{0}(\cdot) \\ I^{1}(\cdot) \\ I^{2}(\cdot) \\ I^{3}(\cdot) \end{array}$	$\stackrel{\scriptstyle <}{\underset{\scriptstyle \sim}{\underset{\scriptstyle \sim}{\atop\scriptstyle \sim}{\underset{\scriptstyle \sim}{\underset{\scriptstyle \sim}{\underset{\scriptstyle \sim}{\underset{\scriptstyle \sim}{\underset{\scriptstyle \sim}{\atop\scriptstyle \sim}{\underset{\scriptstyle}{\atop\scriptstyle \sim}{\underset{\scriptstyle \sim}{\atop\scriptstyle}{\atop\scriptstyle}{\atop\scriptstyle \sim}{\underset{\scriptstyle \sim}{\atop\scriptstyle}{\atop\scriptstyle}{\atop\scriptstyle}{\atop\scriptstyle}{\atop\scriptstyle}{\atop\scriptstyle}{\atop\scriptstyle}{\atop\scriptstyle}{\atop\scriptstyle}{\atop\scriptstyle$	$\begin{array}{c} v_6 \\ \{(\infty,\infty)\} \\ \{(\infty,\infty)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \end{array}$	$\begin{array}{c} v_{7} \\ \hline \{(\infty, \infty)\} \\ \{(\infty, \infty)\} \\ \{(1, 1)\} \\ \{(1, 1)\} \\ \{(1, 1)\} \\ \{(1, 1)\} \\ \{(1, 1)\} \\ \{(1, 1)\} \\ \{(1, 1)\} \end{array}$		$\begin{array}{c} v_9 \\ \hline \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \end{array}$	$\begin{array}{c} \hline v_{10} \\ \hline \{(\infty,\infty)\} \\ \{(\infty,\infty)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \end{array}$	
$\begin{array}{c} I^{0}(\cdot)\\ \hline I^{1}(\cdot)\\ \hline I^{2}(\cdot)\\ \hline I^{3}(\cdot)\\ \hline I^{4}(\cdot) = I^{*}(\cdot) \end{array}$	$\begin{array}{ c c c c c } & < & < & < \\ & < & < & < & < \\ & < & <$	$\begin{array}{c} v_6 \\ \hline \{(\infty,\infty)\} \\ \{(\infty,\infty)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \end{array}$	$\begin{array}{c} \hline v_7 \\ \hline \{(\infty,\infty)\} \\ \{(\infty,\infty)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \hline (1,1)\} \end{array}$		$\begin{array}{c} v_9 \\ \hline \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \\ \{(0,0)\} \end{array}$	$\begin{array}{c} \hline v_{10} \\ \hline \{(\infty,\infty)\} \\ \{(\infty,\infty)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \\ \{(1,1)\} \end{array}$	

Table 2. Execution of the fixpoint algorithm (Algorithm 1) on Example 1 both for $\leq_{\rm C}$ and $\leq_{\rm L}$ orders.

called a *leaf* if height(t) = 0. We denote by $\mathcal{T}_{\restriction t}$, the subtree of \mathcal{T} rooted at t for some $t \in \mathcal{T}$, that is the set of non-empty histories such that $t' \in \mathcal{T}_{\restriction t}$ if and only if t' = tw for some $w \in V^*$. Finally, a (finite or infinite) *branch* of the tree is a (finite or infinite) sequence of nodes $n_0n_1\ldots$ such that for all $k \in \mathbb{N}$, (Last (n_k) , Last (n_{k+1})) $\in E$. The cost of an infinite branch is defined similarly as the cost a play: $\operatorname{Cost}_i(n_0n_1\ldots) = \sum_{k=0}^{\ell-1} w_i(\operatorname{Last}(n_k), \operatorname{Last}(n_{k+1}))$ if ℓ is the least index such that $\operatorname{Last}(n_\ell) \in \mathcal{F}$ and $\operatorname{Cost}_i(n_0n_1\ldots) = \infty$ otherwise. This definition may be easily adapted if the branch is finite.

When we fix a strategy $\sigma_1 \in \Sigma_1^v$ for some $v \in V$, we can see all the possible outcomes consistent with a strategy of \mathcal{P}_2 as a tree consistent with σ_1 . Given $\sigma_1 \in \Sigma_1^v$, the strategy tree \mathcal{T}_{σ_1} of σ_1 is such that: (i) the root of the tree is v, (ii) for all $t \in \mathcal{T}_{\sigma_1}$, if $\text{Last}(t) \in F$ then, for all $t' \in V^+$, $tt' \notin \mathcal{T}_{\sigma_1}$. Otherwise, if $\text{Last}(t) \in V_1 \setminus F$ then, $tv' \in \mathcal{T}_{\sigma_1}$ with $v' = \sigma_1(t)$. Else if $\text{Last}(t) \in V_2 \setminus F$, for all $v' \in \text{Succ}(\text{Last}(t))$ we have $tv' \in \mathcal{T}_{\sigma_1}$.

In the same way, a tree \mathcal{T} which satisfies the following conditions allows to define a strategy $\sigma_{\mathcal{T}}$ of \mathcal{P}_1 . For all $t \in \mathcal{T}$,

- if $\text{Last}(t) = u \in F$, then there is no $u' \in V$ such that $tu' \in \mathcal{T}$;
- if $\operatorname{Last}(t) = u \in V_2 \setminus F$ then, for all $u' \in \operatorname{Succ}(u), tu' \in \mathcal{T}$;
- if $\text{Last}(t) = u \in V_1 \setminus F$ then there exists a unique $u' \in \text{Succ}(u)$ such that $tu' \in \mathcal{T}$; and $\sigma_{\mathcal{T}}(t) = u'$.

Notice that in this way, $\sigma_{\mathcal{T}} \in \Sigma_1^v$ is not well defined on histories which are not consistent with $\sigma_{\mathcal{T}}$ and some $\sigma_2 \in \Sigma_2^v$. This is not a problem for our purpose, we may assume that for such histories $h \in \text{Hist}_1(v)$, $\sigma_{\mathcal{T}}(h) = v'$ for some arbitrary (fixed) $v' \in \text{Succ}(\text{Last}(h))$. We call this well defined strategy the strategy associated with \mathcal{T} .

Example 4. We illustrate the notion of strategy tree by considering the game described in Example 1. Let us recall that the game arena is given in Figure 1.

We define a strategy σ_1 of \mathcal{P}_1 from v_0 as, for all $hv \in \text{Hist}_1(v_0)$: $\sigma(hv) = v'$ with $v' = v_4$ if $v \in \{v_1, v_2, v_3\}$, $v' = v_6$ if $hv \in \{v_0v_2v_4, v_0v_3v_4\}$, $v' = v_7$ if $hv = v_0v_1v_4$, $v' = v_9$ if $v \in \{v_6, v_7, v_9\}$, $v' = v_8$ if $v = v_5$ and $v' = v_{10}$ if $v = v_8$. This strategy is a finite-memory strategy since the choice made in v_4 depends on the past history: if it crossed vertices v_2 or v_3 the next vertex is v_6 while it is v_7 if it crossed v_1 . We also define a positional strategy σ_2 of \mathcal{P}_2 from v_0 as, $\sigma_2(v_0) = v_2$ and $\sigma_2(v_{10}) = v_9$. The outcome of the strategy profile (σ_1, σ_2) from v_0 is $\langle \sigma_1, \sigma_2 \rangle_{v_0} = v_0v_2v_4v_6v_9^{\omega}$ and its cost profile is $\mathbf{Cost}(\langle \sigma_1, \sigma_2 \rangle_{v_0}) = (8, 8)$.

The strategy tree of σ_1 is

$$\mathcal{T}_{\sigma_1} = \{ v_0, v_0 v_1, v_0 v_2, v_0 v_3, v_0 v_1 v_4, v_0 v_2 v_4, v_0 v_3 v_4, v_0 v_1 v_4 v_7, \\ v_0 v_2 v_4 v_6, v_0 v_3 v_4 v_6, v_0 v_1 v_4 v_7 v_9, v_0 v_2 v_4 v_6 v_9, v_0 v_3 v_4 v_6 v_9 \}$$

and is drawn in Figure 2. The root of this tree is the node $n_0 = v_0$ and there are three leaves $v_0v_1v_4v_7v_9$, $v_0v_2v_4v_6v_9$, and $v_0v_3v_4v_6v_9$. The height of the root, height(v_0), is equal to 4 and the depth of the node $n' = v_0v_2v_4$, depth(n'), is equal to 2.



Fig. 2. Strategy tree \mathcal{T}_{σ_1} of the strategy σ_1 defined in Example 4.

Intuition of the proof of Proposition 2 The intuition of the proof is illustrated in Figure 3. This tree represents all the consistent plays with the strategy σ_1 of \mathcal{P}_1 and all the possible strategies of \mathcal{P}_2 . Notice that since for all strategies of \mathcal{P}_2 the target set is reached, all branches of the strategy tree are finite. The main idea is that we remove successively all cycles in the branches of the tree. That ensures that the height of the tree is less than |V|.

If there exists a branch with a cycle beginning with a node ending by a vertex of \mathcal{P}_1 , as the one between the two hatched nodes, \mathcal{P}_1 can directly choose to follow

17

the dotted edge. By considering this new strategy, and so by removing this cycle, the cost profiles of the branches of the new obtained tree are less than or equal to those of the old tree. Once all such kind of cycles are removed, we conclude that there is no more cycle in the tree.

Indeed, the only possibility is that there remains a cycle beginning with a node ending by a vertex of \mathcal{P}_2 , as the black nodes. In this case, either *(i)* all nodes between the black nodes end with a vertex of \mathcal{P}_2 or *(ii)* there exists at least a node ending with a vertex of \mathcal{P}_1 between the two black nodes, as the node in gray. If situation *(i)* occurs there should exist an infinite branch of the tree in which this cycle is repeated infinitely often, but it is impossible because the target set is assumed to be reached along all branches. If it is the situation *(ii)* that occurs, there should exist a branch in which there is a cycle beginning with a node ending by a vertex of \mathcal{P}_1 , as in the figure. But we assumed that this was no longer the case.

Finally, at the end of the procedure, we obtain a tree from which we recover a strategy σ'_1 of \mathcal{P}_1 from v such that for all strategies σ_2 of \mathcal{P}_2 from v, $\mathbf{Cost}(\langle \sigma'_1, \sigma_2 \rangle_v) \lesssim \mathbf{x}$ and $|\langle \sigma'_1, \sigma_2 \rangle_v|_{\mathbf{F}} \leq |V|$.

Remark 2. Let us point out that this construction does not imply that the build strategy σ'_1 is positional. Indeed, it is possible that two nodes ending in some node $u \in V$ appear in two different subtrees and induce two different subtrees.



Fig. 3. Strategy tree \mathcal{T}_{σ_1} associated with a strategy σ_1 of \mathcal{P}_1 .

Proof (Proof of Proposition 2). Let σ_1 be a strategy of \mathcal{P}_1 from v such that for all strategies σ_2 of \mathcal{P}_2 the target set F is reached along $\langle \sigma_1, \sigma_2 \rangle_v$. Let us assume that the strategy tree of σ_1 , \mathcal{T}_{σ_1} , is given by that of Figure 3. As the target set is reached for all strategies of \mathcal{P}_2 , all branches of the tree are finite. Formally, height $(v) \neq \infty$ and all branches $n_0 \dots n_k$ end in a leaf with $\text{Last}(n_k) \in F$ and

 $\operatorname{Cost}(n_0 \dots n_k) \lesssim \mathbf{x}$. Notice that since \mathcal{T}_{σ_1} is rooted at $v, n_0 = v$.

We remove cycles in the tree one by one: we begin with $\mathcal{T}_0 = \mathcal{T}_{\sigma_1}$ and at each step of the procedure these two properties are preserved, (i) the height of the root is finite and (ii) for all (finite) branches $n_0 \ldots n_k$ of the tree, $\mathbf{Cost}(n_0 \ldots n_k) \lesssim \mathbf{x}$. Cycles are removed until there are none left and we obtain a tree \mathcal{T}_{α^*} , for some $\alpha^* \in \mathbb{N}$, which respects (i) and (ii). Moreover, because there is no more cycle, the height of \mathcal{T}_{α^*} is less than |V|. Finally, from \mathcal{T}_{α^*} we recover a strategy σ'_1 such that for all $\sigma_2 \in \Sigma_2^v$, we have that $\mathbf{Cost}(\langle \sigma'_1, \sigma_2 \rangle_v) \lesssim \mathbf{x}$ and $|\langle \sigma'_1, \sigma_2 \rangle_v|_{\mathrm{F}} \leq |V|$.

More precisely, let us assume that we want to build $\mathcal{T}_{\alpha+1}$ from \mathcal{T}_{α} . There still exists a branch $n_0 \ldots n_k \ldots n_\ell \ldots n_m$ in \mathcal{T}_{α} such that

(i) $\exists k, \ell \in \mathbb{N}, 0 \le k < \ell \le m$, such that $\operatorname{Last}(n_k) = \operatorname{Last}(n_\ell)$ and (ii) $\operatorname{Last}(n_k) \in V_1$. (1)

This is for example the case of the hatched nodes in Figure 3. In this situation, \mathcal{P}_1 generates an unnecessary cycle since the target set is not reached along this cycle and the cost profile increase along the cycle. Thus \mathcal{P}_1 can avoid this unnecessary cycle by directly choosing the doted edge.

We have to build a new tree $\mathcal{T}_{\alpha+1}$ from \mathcal{T}_{α} . Let us recall that, while branches are sequences of nodes, trees are sets of non-empty histories. Thus, for all $w \in \mathcal{T}_{\alpha}$, $w \in \text{Hist}(v)$. The new tree $\mathcal{T}_{\alpha+1}$ is obtained as follows: for all $w \in \mathcal{T}_{\alpha}$:

- if $w = \text{Last}(n_0) \dots \text{Last}(n_k) \dots \text{Last}(n_\ell) w_{k+1} \dots w_n$ for some $n \in \mathbb{N}$ then, the cycle $\text{Last}(n_k) \dots \text{Last}(n_\ell)$ is removed and so $\text{Last}(n_0) \dots \text{Last}(n_k) w_{k+1} \dots w_n \in \mathcal{T}_{\alpha+1}$;

We now consider \mathcal{T}_{α^*} in which there is no more branch that satisfies conditions (1). We prove that there is no branch $n_0 \ldots n_k \ldots n_\ell \ldots n_m$ of \mathcal{T}_{α^*} such that (i) $\exists k, \ell \in \mathbb{N}, 0 \leq k < \ell \leq m$, such that $\text{Last}(n_k) = \text{Last}(n_\ell)$ and (ii) $\text{Last}(n_k) \in V_2$.

Let us assume the contrary in order to obtain a contradiction. We consider the following two cases:

- if for all $k < \xi < \ell$, Last $(n_{\xi}) \in V_2$ then, Last $(n_0) \dots (\text{Last}(n_k) \dots \text{Last}(n_{\ell-1}))^j$ should be nodes of \mathcal{T}_{α^*} for all $j \in \mathbb{N}$. This is in contradiction with the fact that the height of the root of \mathcal{T}_{α^*} is finite.
- Otherwise, we consider the least index $\xi \in \mathbb{N}$ such that $k < \xi < \ell$ and Last $(n_{\xi}) \in V_1$. This is for example the case of the gray node between the two black nodes in Figure 3. In this case, the node $t = \text{Last}(n_0) \dots \text{Last}(n_k) \dots \text{Last}(n_{\xi}) \dots \text{Last}(n_{\ell}) \dots \text{Last}(n_{\xi})$ should be a node of \mathcal{T}_{α^*} . Thus, there is at least one branch in the tree which has t as a prefix. Which contradicts the assumption that there is no more cycle generates by \mathcal{P}_1 in \mathcal{T}_{α^*} .

Since we have removed all cycles of the finite branches of \mathcal{T}_{σ_1} . We have that the height of \mathcal{T}_{α^*} is less than |V| and the cost profiles of the branches may only

⁻ else, $w \in \mathcal{T}_{\alpha+1}$.

decrease, because the weights on the edges are natural numbers. It means that for all branches $n_0 \ldots n_k$ of \mathcal{T}_{α^*} , $\mathbf{Cost}(n_0 \ldots n_k) \lesssim \mathbf{x}$.

To conclude, we recover from \mathcal{T}_{α^*} the strategy σ'_1 associated with \mathcal{T}_{α^*} as explained in the paragraph about strategy tree at the beginning of Appendix A.1. For all $\sigma_2 \in \Sigma_2^v$, we have (i) $\mathbf{Cost}(\langle \sigma'_1, \sigma_2 \rangle_v) \lesssim \mathbf{x}$ and (ii) $|\langle \sigma'_1, \sigma_2 \rangle_v|_{\mathbf{F}} \leq |V|$. \Box

Correctness Some of the arguments of our proofs rely on the following lemma and its corollary.

Lemma 1. For all $k \in \mathbb{N}$,

- 1. for all $v \in V_1 \setminus F$, for all $\mathbf{x} \in I^{k+1}(v)$, there exist $v' \in \operatorname{Succ}(v)$ and $\mathbf{x} \in I^k(v')$ such that $\mathbf{x} = \mathbf{x}' + \mathbf{w}(v, v')$.
- 2. for all $v \in V_2 \setminus F$, for all $\mathbf{x} \in I^{k+1}(v)$, for all $v' \in Succ(v)$, there exists $\mathbf{x}' \in \mathbf{I}^k(v')$ such that $\mathbf{x}' + \mathbf{w}(v, v') \leq \mathbf{x}$.

Corollary 1. For all $v \in V$,

- If $v \in V_1 \setminus F$ then, for all $\mathbf{x} \in I^*(v)$, there exist $v' \in Succ(v)$ and $\mathbf{x} \in I^*(v')$ such that $\mathbf{x} = \mathbf{x}' + \mathbf{w}(v, v')$.
- If $v \in V_2 \setminus F$ then, for all $\mathbf{x} \in I^*(v)$, for all $v' \in Succ(v)$, there exists $\mathbf{x}' \in V_2 \setminus F$ $I^*(v')$ such that $\mathbf{x}' + \mathbf{w}(v, v') \lesssim \mathbf{x}$.

This section is devoted to the proof of the following theorem.

Theorem 8. For all $v \in V$, minimal(Ensure_<(v)) = $I^*(v)$.

This is a direct consequence of Proposition 4.

Proposition 4. For all $k \in \mathbb{N}$ and all $v \in V$, minimal(Ensure^k(v)) = I^k(v).

Proof. We proceed by induction on ℓ . <u>Base case</u> $\ell = 0$, let $v \in V$. If $v \in F$ then, Ensure⁰ $(v) = \{ \mathbf{x} \in \overline{\mathbb{N}}^d \mid \mathbf{0} \leq \mathbf{x} \}$ and minimal(Ensure⁰ $(v)) = \{ \mathbf{0} \}$ which is equal to I⁰(v) by Algorithm 1. Else, if $v \notin F$, Ensure⁰ $(v) = \{ \infty \}$ and I⁰ $(v) = \{ \infty \}$.

Let us assume that the assertion is true for all $0 \le \ell \le k$ and let us prove it is still true for $\ell = k + 1$. In particular, the following equality holds:

$$minimal(Ensure^{k}(v)) = I^{k}(v)$$
(2)

Since $\operatorname{Ensure}^{k}(v)$ is upward closed, we have that:

$$\operatorname{Ensure}^{k}(v) = \uparrow \mathbf{I}^{k}(v) \tag{3}$$

If $v \in F$, we have minimal(Ensure^k(v)) = $\{\mathbf{0}\} = I^k(v)$ for all $k \in \mathbb{N}$. This is the reason why we assume $v \notin F$ in the rest of the proof.

For all
$$v \notin F$$
, we prove that $\operatorname{Ensure}^{k+1}(v) = \begin{cases} \bigcup_{\substack{v' \in \operatorname{Succ}(v) \\ v' \in \operatorname{Succ}(v)}} \uparrow \mathbf{I}^k(v') + \mathbf{w}(v,v') & \text{if } v \in V_1 \\ \bigcap_{v' \in \operatorname{Succ}(v)} \uparrow \mathbf{I}^k(v') + \mathbf{w}(v,v') & \text{if } v \in V_2 \end{cases}$
That proves that minimal($\operatorname{Ensure}^{k+1}(v)$) = $\mathbf{I}^{k+1}(v)$.

Т

- We first prove the inclusion ⊆. Let $\mathbf{x} \in \text{Ensure}^{k+1}(v)$. We know that there exists a strategy $\sigma_1^{k+1} \in \Sigma_1^v$ such that for all strategies $\sigma_2 \in \Sigma_2^v$ we have that

$$\mathbf{Cost}(\langle \sigma_1^{k+1}, \sigma_2 \rangle_v) \lesssim \mathbf{x} \quad \text{and} \quad |\langle \sigma_1^{k+1}, \sigma_2 \rangle_v|_{\mathbf{F}} \le k+1.$$
(4)

• If $v \in V_1$, let $v' = \sigma_1^{k+1}(v)$. We consider $\sigma_1^{k+1} \downarrow_v$: Hist₁ $(v') \longrightarrow V : hu \mapsto$ $\begin{aligned} &\sigma_1^{k+1}(vhu).\\ &\text{We have for all }\sigma_2\in \varSigma_2^v: \end{aligned}$

$$\begin{split} \mathbf{Cost}(\langle \sigma_1^{k+1}, \sigma_2 \rangle_v) &= \mathbf{Cost}(v \langle \sigma_1^{k+1} _{\upharpoonright v}, \sigma_{2 \upharpoonright v} \rangle_{v'}) \\ &= \mathbf{w}(v, v') + \mathbf{Cost}(\langle \sigma_1^{k+1} _{\upharpoonright v}, \sigma_{2 \upharpoonright v} \rangle_{v'}) \quad (v \not\in \mathbf{F}) \end{split}$$

Thus in particular, for all $\sigma_2 \in \Sigma_2^{v'}$:

$$\mathbf{w}(v,v') + \mathbf{Cost}(\langle \sigma_1^{k+1}{}_{\restriction v}, \sigma_2
angle_{v'}) \lesssim \mathbf{x}$$

and

$$|\langle \sigma_1^{k+1}_{|v}, \sigma_2 \rangle_{v'}|_{\mathbf{F}} = |\langle \sigma_1^{k+1}, \sigma_2 \rangle_{v}|_{\mathbf{F}} - 1 \le k$$

Meaning that $\mathbf{x} - \mathbf{w}(v, v') \in \text{Ensure}^k(v')$. By Equation (3), $\text{Ensure}^k(v') = \uparrow \mathbf{I}^k(v')$. It follows that $x \in \uparrow \mathbf{I}^k(v') + \mathbf{w}(v, v')$ and we obtain the result we were looking for: $\mathbf{x} \in \bigcup_{v' \in \text{Succ}(v)} \uparrow \mathbf{I}^k(v') + \mathbf{w}(v, v')$.

• If $v \in V_2$, for all $v' \in \operatorname{Succ}(v)$ and for all strategies $\sigma_2 \in \Sigma_2^{v'}$, we have by Equation (4):

$$\mathbf{Cost}(v\langle \sigma_{1}^{k}{}_{\restriction v}, \sigma_{2} \rangle_{v'}) = \mathbf{w}(v, v') + \mathbf{Cost}(\langle \sigma_{1}^{k}{}_{\restriction v}, \sigma_{2} \rangle_{v'}) \qquad (v \notin \mathbf{F})$$

$$\lesssim \mathbf{x}$$

and, since $v \notin F$,

$$\left| \langle \sigma_{1 \upharpoonright v}^{k}, \sigma_{2} \rangle_{v'} \right|_{\mathbf{F}} = \left| \langle \sigma_{1}^{k+1}, \sigma_{2} \rangle_{v} \right|_{\mathbf{F}} - 1 \le k.$$

It follows that for all $v' \in \operatorname{Succ}(v)$, $\mathbf{x} - \mathbf{w}(v, v') \in \operatorname{Ensure}^{k}(v') = \uparrow \mathbf{I}^{k}(v')$, by Equation (3). Thus we conclude that $\mathbf{x} \in \bigcap_{v' \in \text{Succ}(v)} \uparrow \mathbf{I}^k(v') +$ $\mathbf{w}(v, v').$

- We now prove the inclusion \supseteq .
 - If $v \in V_1$, let $\mathbf{x} \in \bigcup_{v' \in \operatorname{Succ}(v)} \uparrow \mathbf{I}^k(v') + \mathbf{w}(v, v')$. It means that there exists

 $\mathbf{y} \in \uparrow \mathbf{I}^k(v')$ such that $\mathbf{x} = \mathbf{y} + \mathbf{w}(v, v')$ for some $v' \in \operatorname{Succ}(v)$. By Equation (3), $\mathbf{y} \in \text{Ensure}^k(v')$, thus there exists $\sigma_1^k \in \Sigma_1^{v'}$ such that for all $\sigma_2 \in \Sigma_2^{v'}$ we have:

$$\mathbf{Cost}(\langle \sigma_1^k, \sigma_2 \rangle_{v'}) \lesssim \mathbf{y} \quad \text{and} \quad |\langle \sigma_1^k, \sigma_2 \rangle_{v'}|_{\mathbf{F}} \le k.$$
(5)

We consider $\sigma_1^{k+1} \in \Sigma_1^v$ defined as

$$\sigma_1^{k+1}(vh) = \begin{cases} v' & \text{if } h \text{ is the empty history, } i.e., vh = v \\ \sigma_1^k(h) & \text{otherwise} \end{cases}$$

Let $\sigma_2 \in \Sigma_2^v$,

$$\begin{aligned} \mathbf{Cost}(\langle \sigma_1^{k+1}, \sigma_2 \rangle_v) &= \mathbf{w}(v, v') + \mathbf{Cost}(\langle \sigma_1^{k+1} \upharpoonright v, \sigma_2 \upharpoonright v \rangle_{v'}) & (v \notin \mathbf{F}) \\ &= \mathbf{w}(v, v') + \mathbf{Cost}(\langle \sigma_1^k, \sigma_2 \upharpoonright v \rangle_{v'}) \lesssim \mathbf{w}(v, v') + \mathbf{y} \quad (\text{By Eq. (5)}) \end{aligned}$$

Moreover, $|\langle \sigma_1^{k+1}, \sigma_2 \rangle_v|_{\mathbf{F}} = 1 + |\langle \sigma_1^k, \sigma_{2 \restriction v} \rangle_{v'}|_{\mathbf{F}} \leq 1 + k.$ We conclude that $\mathbf{x} \in \text{Ensure}^{k+1}(v)$. • If $v \in V_2$, let $\mathbf{x} \in \bigcap_{v' \in \text{Succ}(v)} \uparrow \mathbf{I}^k(v') + \mathbf{w}(v, v')$. For all $v' \in \text{Succ}(v')$, there exists $\mathbf{y}' \in \uparrow \mathbf{I}^k(v')$ such that $\mathbf{x} = \mathbf{y}' + \mathbf{w}(v, v')$. By Eq. (3), there exists $\sigma_1^{v'} \in \Sigma_1^{v'}$ such that for all $\sigma_2 \in \Sigma_2^{v'}$,

$$\mathbf{Cost}(\langle \sigma_1^{v'}, \sigma_2 \rangle_{v'}) \lesssim \mathbf{y}' \quad \text{ and } \quad |\langle \sigma_k^{v'}, \sigma_2 \rangle_{v'}|_{\mathbf{F}} \leq k$$

Let us consider $\sigma_1^{k+1} \in \Sigma_1^v$ defined as: $\sigma_1^{k+1}(vv'h) = \sigma_1^{v'}(v'h)$ for all $vv'h \in \operatorname{Hist}_1(v)$. Let $\sigma_2 \in \Sigma_2^v$, if $\sigma_2(v) = v'$, we have:

$$\begin{aligned} \mathbf{Cost}(\langle \sigma_1^{k+1}, \sigma_2 \rangle_v) &= \mathbf{Cost}(v \langle \sigma_1^{k+1}{}_{\upharpoonright v}, \sigma_2{}_{\upharpoonright v} \rangle_{v'}) = \mathbf{Cost}(v \langle \sigma_1^{v'}, \sigma_2{}_{\upharpoonright v} \rangle_{v'}) \\ &= \mathbf{w}(v, v') + \mathbf{Cost}(\langle \sigma_1^{v'}, \sigma_2{}_{\upharpoonright v} \rangle_{v'}) \\ &\lesssim \mathbf{w}(v, v') + \mathbf{y'}. \end{aligned}$$

Moreover, $|\langle \sigma_1^{k+1}, \sigma_2 \rangle_v|_{\mathbf{F}} = |v \langle \sigma_1^{v'}, \sigma_2_{\uparrow v} \rangle_{v'}|_{\mathbf{F}} = |\langle \sigma_1^{v'}, \sigma_2_{\uparrow v} \rangle_{v'}|_{\mathbf{F}} + 1 \le k+1.$ In conclusion, $\mathbf{x} \in \text{Ensure}^{k+1}(v)$.

A.2 **Time Complexity**

Let us recall that $W = \max\{w_i(e) \mid 1 \le i \le d \text{ and } e \in E\}$. We also explicitly restate a remark done in the main part of the paper (in Section 3.3).

Remark 3. In Line 13, we are allowed to assume that \mathbf{x}' is in $\mathbf{I}^k(v')$ instead of $\uparrow \mathbf{I}^{k}(v')$ thanks to Lemma 2 stated just after this remark.

Lemma 2. For all $k \in \mathbb{N}$, for all $v \in V_1 \setminus \mathbb{F}$,

$$\mathbf{I}^{k+1}(v) = \operatorname{minimal}\left(\bigcup_{v' \in \operatorname{Succ}(v)} \mathbf{I}^k(v) + \mathbf{w}(v, v')\right).$$

Lexicographic Order In this section we prove Theorem 3.

Theorem 3. If \leq is the lexicographic order, the fixpoint algorithm runs in time polynomial in |V| and d.

By abuse of notation, the only $\mathbf{x} \in \mathbf{I}^k(v)$ is denoted by $\overline{\mathrm{Val}}^k(v)$.

Proposition 6. If $v \in V_1 \setminus F$, $I^{k+1}(v) = \min_{\leq_L} \{\overline{\operatorname{Val}}^k(v') + \mathbf{w}(v, v') \mid v' \in \operatorname{Succ}(v)\}.$

Proof. Let $v \in V_1 \setminus F$,

$$\begin{split} \mathbf{I}^{k+1}(v) &= \min \left(\bigcup_{v' \in \operatorname{Succ}(v)} \uparrow \mathbf{I}^{k}(v') + \mathbf{w}(v,v') \right) & \text{By Algorithm 1} \\ &= \min \left(\bigcup_{v' \in \operatorname{Succ}(v)} \mathbf{I}^{k}(v') + \mathbf{w}(v,v') \right) & \text{By Lemma 2} \\ &= \min \left(\{ \overline{\operatorname{Val}}^{k}(v') + \mathbf{w}(v,v') \mid v' \in \operatorname{Succ}(v) \} \right) \\ &= \min_{\leq_{\mathrm{L}}} \{ \overline{\operatorname{Val}}^{k}(v') + \mathbf{w}(v,v') \mid v' \in \operatorname{Succ}(v) \} \end{split}$$

Proposition 7. If $v \in V_2 \setminus F$, $I^{k+1}(v) = \max_{\leq L} \{\overline{\operatorname{Val}}^k(v') + \mathbf{w}(v, v') \mid v' \in \operatorname{Succ}(v)\}.$

Proof. We begin the proof by a remark: if $\mathbf{x}, \mathbf{y} \in \overline{\mathbb{N}}^d$ then,

$$\uparrow \{\mathbf{x}\} \cap \uparrow \{\mathbf{y}\} = \uparrow \{\max_{\leq_{\mathrm{L}}} \{\mathbf{x}, \mathbf{y}\}\}$$
(6)

Let $v \in V_2 \setminus F$,

$$I^{k+1}(v) = \min\left(\bigcap_{v' \in \operatorname{Succ}(v)} \uparrow I^{k}(v') + \mathbf{w}(v,v')\right) \qquad \text{By Algorithm 1}$$
$$= \min\left(\bigcap_{v' \in \operatorname{Succ}(v)} \uparrow \{\overline{\operatorname{Val}}^{k}(v') + \mathbf{w}(v,v')\}\right)$$
$$= \min\left(\max_{\leq L} \{\overline{\operatorname{Val}}^{k}(v') + \mathbf{w}(v,v') \mid v' \in \operatorname{Succ}(v)\}\right) \qquad \text{By Equation (6)}$$
$$= \max_{\leq L} \{\overline{\operatorname{Val}}^{k}(v') + \mathbf{w}(v,v') \mid v' \in \operatorname{Succ}(v)\}$$

Proof (Proof of Theorem 3). By Proposition 1, Algorithm 1 mainly consists in $(|V| + 1) \cdot |V| \approx |V|^2$ operations of the type $\min_{\leq_L} \{\overline{\operatorname{Val}}^k(v') + \mathbf{w}(v,v') \mid v' \in \operatorname{Succ}(v)\}$ or $\max_{\leq_L} \{\overline{\operatorname{Val}}^k(v') + \mathbf{w}(v,v') \mid v' \in \operatorname{Succ}(v)\}$ which are done in $\mathcal{O}(|V| \cdot d)$. It follows that the global complexity of the algorithm if the considered order is the lexicographic order is in $\mathcal{O}(|V|^3 \cdot d)$.

Componentwise Order This section is devoted to the proof of Theorem 4.

Theorem 4. If \leq is the componentwise order, the fixpoint algorithm runs in time polynomial in W and |V| and exponential in d.

During the computation of the fixpoint algorithm, even if we have a finite representation of the infinite sets $\uparrow I^k(v)$ by only storing their minimal elements $I^k(v)$, we need to explain how to manipulate them efficiently. In particular, we explicit how given some accurate representation of $\uparrow I^k(v)$ and $\uparrow I^k(v')$ for some $k \in \mathbb{N}$ and $v, v' \in V$ we compute: (i) the union $\uparrow I^k(v) \cup \uparrow I^k(v')$, (ii) the intersection $\uparrow I^k(v) \cap \uparrow I^k(v')$, (iii) the translation $\uparrow I^k(v) + \mathbf{w}(v, v')$ and (iv) the set of minimal elements minimal($\uparrow I^k(v)$). Inspired by the approach explained in [5], we use a part of the logic of upward closed sets in order to express the infinite sets $\uparrow I^k(v)$ in a convenient way.

Let $D = \{t_1, \ldots, t_d\}$ be a set of d variables, if $G = \{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$ for some $n \in \mathbb{N}$ and $\mathbf{x}^1, \ldots, \mathbf{x}^n \in \overline{\mathbb{N}}^d$, we can express $\uparrow G$ as a formula ϕ :

$$\phi = \bigvee_{1 \le i \le n} (t_1 \ge x_1^i) \land \ldots \land (t_d \ge x_d^i).$$
(7)

We define the size of the formula, denoted by $|\phi|$, by $n \cdot d$. Additionally, the set G is called the set of generators of \uparrow G, or equivalently the set of generators of ϕ . Thus G allows to encode the formula in a succinct way. Notice that the fewer generators there are, the more succinct the formula to express \uparrow G is. Moreover, the set of tuples that evaluates formula ϕ to true are denoted by $\llbracket \phi \rrbracket$, *i.e.*, $\llbracket \phi \rrbracket = \{ \mathbf{c} \in \overline{\mathbb{N}}^d \mid \bigvee_{1 \leq i \leq n} (c_1 \geq x_1^i) \land \ldots (c_d \geq x_d^i) \}$. Thus, in particular, $\llbracket \phi \rrbracket = \uparrow G$. Conversely, if we have a formula ϕ as in Equation (7), it represents an upward closed set $\llbracket \phi \rrbracket$ and its set of generators is given by $\operatorname{gen}(\phi) = \{ \mathbf{x}^1, \ldots, \mathbf{x}^n \}$.

For each $\uparrow I^k(v)$, we denote by $\phi(k, v)$ the corresponding formula. In Proposition 8, we explain how unions, intersections and translations of sets of the type $\uparrow I^k(v)$ are done and what are the complexities of those operations.

Proposition 8 ([5]). Given two sets $I^k(v) = \{\mathbf{x^1}, \ldots, \mathbf{x^n}\}$, for some $n \in \mathbb{N}$, and $I^k(v') = \{\mathbf{y^1}, \ldots, \mathbf{y^m}\}$, for some $m \in \mathbb{N}$, such that their upward closures are expressed respectively by $\phi(k, v)$ and $\phi(k, v')$, we have:

1. Union: the set $X = \uparrow I^k(v) \cup \uparrow I^k(v')$ is expressed thanks to the formula

$$\psi = \bigvee_{1 \le i \le n+m} (t_1 \ge z_1^i) \land \ldots \land (t_d \ge z_d^i)$$

where $\mathbf{z}^{\mathbf{i}} = \mathbf{x}^{\mathbf{i}}$ if $1 \leq i \leq n$ and $\mathbf{z}^{\mathbf{i}} = \mathbf{y}^{\mathbf{i}-\mathbf{n}}$ if $n+1 \leq i \leq n+m$. Thus $|\psi| = |\phi(k,v)| + |\phi(k,v')|$ and this operation is done in $\mathcal{O}(|\phi(k,v)| + |\phi(k,v')|)$. 2. Intersection: the set $X = \uparrow \mathbf{I}^{k}(v) \cap \uparrow \mathbf{I}^{k}(v')$ is expressed thanks to the for-

mula

$$\psi = \bigvee_{1 \le i \le n} \bigvee_{1 \le j \le m} (t_1 \ge \max\{x_1^i, y_1^j\}) \land \ldots \land (t_d \ge \max\{x_d^i, y_d^j\}).$$

Thus $|\psi| = |\phi(k, v)| \cdot |\phi(k, v')|$ and this operation is done in $\mathcal{O}(|\phi(k, v)| \cdot |\phi(k, v')|)$.

3. **Translation**: the set $X = \uparrow I^k(v) + c$ is expressed thanks to the formula

$$\psi = \bigvee_{1 \le i \le n} (t_1 \ge x_1^i + c_1) \land \ldots \land (t_d \ge x_d^i + c_d).$$

Thus $|\psi| = |\phi(k, v)|$ and this operation is done in $\mathcal{O}(|\phi(k, v)|)$.

Even if the sets $I^k(v)$ and $I^k(v')$ are minimal, an union or an intersection as described in Statements 1 and 2 in Proposition 8 may produce a formula ψ such that set gen (ψ) is not minimal. Therefore we consider the minimization of a set of generators in order to obtain a (minimal) new set of generators that encodes a new formula ϕ' in such a way that $[\![\phi']\!] = [\![\phi]\!]$ and $|\phi'|$ is as small as possible. Notice that the translation operation preserves the minimality of the set of generators.

Proposition 9 ([5]). If an upward closed set X is expressed by ϕ with $G = gen(\phi)$ and X' = minimal(X), then G' = minimal(G) can be computed in $\mathcal{O}(|\phi|^2)$.

Remark 4. Notice that in the previous proposition, as X is upward closed, $G' = minimal(G) = minimal(\uparrow G) = minimal(X)$.

The key idea in order to obtain an algorithm at most polynomial in W and |V| and exponential in d is to ensure that the size of the formulae, and so their sets of generators, that represent the sets $\uparrow \mathbf{I}^k(v)$ do not grow too much. The size of such a formula depends on the number of elements in $\mathbf{I}^k(v)$ and the number of dimensions d. Since for all $k \in \mathbb{N}$, $\operatorname{Ensure}^k(v) \subseteq \operatorname{Ensure}^{k+1}(v) \subseteq \mathbf{I}^{|V|}(v)$ and $|\operatorname{Ensure}^{|V|}(v) \setminus \{\infty\}| \leq (\mathbf{W} \cdot |V|)^d$, the maximal size of a set $\mathbf{I}^k(v) = \min(\operatorname{Ensure}^k(v))$ is also bounded by $(\mathbf{W} \cdot |V|)^d$. Let $|\mathbf{I}_{\max}| = \mathbf{W}^d \cdot |V|^d$ be this (rough) upper-bound.

Proposition 10. For all $k \in \mathbb{N}$ and $v \in V_1 \setminus F$, the operation

$$\mathbf{I}^{k+1}(v) = \min\left(\bigcup_{v' \in \operatorname{Succ}(v)} \uparrow \mathbf{I}^k(v') + \mathbf{w}(v,v')\right)$$

can be computed in $\mathcal{O}(d^2 \cdot W^{2d} \cdot |V|^{2 \cdot d+2})$.

Proof. Let $k \in \mathbb{N}$ and $v \in V_1 \setminus F$. For all $v' \in \operatorname{Succ}(v)$, we denote by $\phi(k, v')$ the formulae that express the sets $\uparrow I^k(v')$. By hypothesis, for all $v' \in \operatorname{Succ}(v)$, $|\phi(k, v')| \leq |I_{\max}| \cdot d$.

- **Translations.** We first compute the sets $\uparrow \mathbf{I}^{k}(v') + \mathbf{w}(v, v')$ by computing their associated formulae that we denote by $\phi(k, v') + \mathbf{w}(v, v')$. Notice that computing each of those formulae can be done in $\mathcal{O}(|\phi(k, v')|) = \mathcal{O}(|\mathbf{I}_{\max}| \cdot d)$ and that the obtained formula has a size $|\phi(k, v') + \mathbf{w}(v, v')| = |\phi(k, v')|$, by Proposition 8. In conclusion, the computation of all translations is done in $\mathcal{O}(|V| \cdot |\mathbf{I}_{\max}| \cdot d)$.
- Unions. Let us denote by ψ the formula that expresses $\bigcup_{v' \in \text{Succ}(v)} \uparrow \text{I}^k(v') +$

 $\mathbf{w}(v, v') \text{ which can be obtained thanks to successive unions in } \mathcal{O}(\sum_{v' \in \text{Succ}(v)} |\phi(k, v') + \mathbf{w}(v, v')|) = \mathcal{O}(\sum_{v' \in \text{Succ}(v)} |\phi(k, v')|) = \mathcal{O}(|\mathbf{I}_{\max}| \cdot d \cdot |V|), \text{ by Proposition 8.}$

- Minimization of the set of generators of ψ . It remains to compute thanks to ψ the set minimal $(\bigcup_{v' \in \operatorname{Succ}(v)} \uparrow \mathrm{I}^k(v') + \mathbf{w}(v,v'))$ which corresponds

to the minimization of the set of generators of ψ by Remark 4. This can be done in $\mathcal{O}(|\psi|^2) = \mathcal{O}(|\mathbf{I}_{\max}|^2 \cdot d^2 \cdot |V|^2)$, by Proposition 9.

In conclusion, the global complexity of computing $I^{k+1}(v)$ for $v \in V_1 \setminus F$ is $\mathcal{O}(W^{2d} \cdot |V|^{2d} \cdot d^2 \cdot |V|^2) = \mathcal{O}(d^2 \cdot W^{2d} \cdot |V|^{2d+2}).$

Proposition 11. For all $k \in \mathbb{N}$ and $v \in V_2 \setminus F$, the operation

$$\mathbf{I}^{k+1}(v) = \min\left(\bigcap_{v' \in \operatorname{Succ}(v)} \uparrow \mathbf{I}^k(v') + \mathbf{w}(v,v')\right)$$

can be computed in $\mathcal{O}(d^4 \cdot W^{4d} \cdot |V|^{4d+1})$.

Proof (Proof of Proposition 11).

Let $k \in \mathbb{N}$ and $v \in V_2 \setminus F$. For all $v' \in \operatorname{Succ}(v)$, we denote by $\phi(k, v')$ the formulae that express the sets $\uparrow I^k(v')$. By hypothesis, for all $v' \in \operatorname{Succ}(v)$, $|\phi(k, v')| \leq |I_{\max}| \cdot d$. The line 16 of Algorithm 1 may be replaced by:

1 $I = I^{k}(w)$ for some $w \in \text{Succ}(v)$ 2 for $v' \in \text{Succ}(v)$ do 3 $\begin{vmatrix} J = \uparrow I^{k}(v') + \mathbf{w}(v, v') \\ I = \text{minimal}(\uparrow I \cap J) \\ 5 I^{k+1}(v) = I \end{vmatrix}$

Let us analyze the complexity of those lines. We assume that $\phi(J)$ and $\phi(\uparrow I)$ are the formulae that express the sets J and $\uparrow I$ respectively. Thanks to the

minimization of the set of generators of the corresponding formula in Line 4 , the formulae $\phi(J)$ and $\phi(\uparrow I)$ have a size at most equal to $|I_{max}| \cdot d$.

- Complexity of Line 3: $\mathcal{O}(|\phi(k, v')|) = \mathcal{O}(|\mathbf{I}_{\max}| \cdot d)$, by Proposition 8;
- Complexity of Line 4: the intersection is done in $\mathcal{O}(|\mathbf{I}_{\max}|^2 \cdot d^2)$, by Proposition 8, and generates a formula $\phi(\uparrow I \cap J)$ of size at most $|\phi(\uparrow I \cap J)| \leq |\mathbf{I}_{\max}|^2 \cdot d^2$. The minimization of the set of generators of $\uparrow I \cap J$ is done in $\mathcal{O}(|\mathbf{I}_{\max}|^4 \cdot d^4)$, by Proposition 9, and allows to encode a formula of size at most $|\mathbf{I}_{\max}| \cdot d$ which also expresses $\uparrow I \cap J$.
- The global complexity of Lines 1 to 5, is $|V| \cdot (\mathcal{O}(|\mathbf{I}_{\max}| \cdot d) + \mathcal{O}(|\mathbf{I}_{\max}|^2 d^2) + \mathcal{O}(|\mathbf{I}_{\max}|^4 \cdot d^4)) = \mathcal{O}(|V| \cdot |\mathbf{I}_{\max}|^4 \cdot d^4) = \mathcal{O}(d^4 \cdot \mathbf{W}^{4d} \cdot |V|^{4d+1}).$

Proof (Proof of Theorem 4). Since the algorithm terminates in less than |V|+1 steps (Proposition 1), the fixpoint algorithm consists of about |V| repetitions of the procedure between Line 5 and Line 16. This procedure is a for loop on all the vertices of the game graph which computes essentially either an operation

$$\begin{array}{l} \operatorname{minimal} \left(\bigcup_{v' \in \operatorname{Succ}(v)} \uparrow \mathbf{I}^{k}(v') + \mathbf{w}(v,v') \right) \text{ or minimal} \left(\bigcap_{v' \in \operatorname{Succ}(v)} \uparrow \mathbf{I}^{k}(v') + \mathbf{w}(v,v') \right) \\ \text{Thus, by Proposition 10 and Proposition 11 the complexity of the fixpoint algorithm is in $\cdot \mathcal{O}(|V|^{2} \cdot \max\{d^{2} \cdot \mathbf{W}^{2d} \cdot |V|^{2d+2}, d^{4} \cdot \mathbf{W}^{4d} \cdot |V|^{4d+1}\}) = \mathcal{O}(|V|^{2} \cdot d^{4} \cdot \mathbf{W}^{4d} \cdot |V|^{4d+1}) \\ \mathbb{C}(|V|^{2} \cdot W^{4d} \cdot |V|^{4d+3}). \qquad \Box$$$

A.3 Synthesis of Lexico-optimal and Pareto-optimal Strategies

In this section, we prove Theorem 5 and Proposition 5.

Theorem 5. Given $u \in V$ and $\mathbf{c} \in I^*(u) \setminus \{\infty\}$, the strategy $\sigma_1^* \in \Sigma_1^u$ defined in Definition 4 is such that for all $\sigma_2 \in \Sigma_2^u$, $\mathbf{Cost}(\langle \sigma_1^*, \sigma_2 \rangle_u) \lesssim \mathbf{c}$.

To prove Theorem 5, we consider the strategy tree $\mathcal{T}_{\sigma_1^*}$ of σ_1^* and introduce a labeling function of the tree nodes which allows to keep track some properties on these nodes. This labeling function and properties are detailed in the following proposition.

Proposition 12. For $u \in V$ and $\mathbf{c} \in I^*(u) \setminus \{\infty\}$, if $\mathcal{T}_{\sigma_1^*}$ is the strategy tree of the strategy σ_1^* as defined in Definition 4 then, there exists a labeling function $\tau : \mathcal{T}_{\sigma_1^*} \longrightarrow \mathbb{N}^d$ such that, $\tau(u) = \mathbf{c} \in I^*(u)$ and, for all $hv \in \mathcal{T}_{\sigma_1^*}$ such that $|hv| \geq 1$:

1. $\tau(hv) \in I^*(v);$ 2. If Last(h) $\in V_1$ then, $(v, \tau(hv)) = f^*_{Last(h)}(\tau(h));$ 3. $\tau(hv) \leq_L \tau(h) - \mathbf{w}(Last(h), v);$ 4. $\tau(hv) = \min_{\leq_L} \{\mathbf{x}' \in I^*(v) \mid \mathbf{x}' \lesssim \mathbf{c} - \mathbf{Cost}(hv) \land \mathbf{x}' \leq_L \mathbf{c} - \mathbf{Cost}(hv) \}.$ Remark 5. The same remark as in Remark 1 is applicable in the context of Proposition 12. Even if the lexicographic order $\leq_{\rm L}$ is used in the statement of the properties of the labeling function τ , Proposition 12 holds both for the lexicographic order and the componentwise order.

The intuition behind the properties on the labeling function τ in Propsosition 12 is the following one. The first property ensures that the values of τ is one of the ensured value at the fixpoint in the set corresponding to the last vertex of the node. The second property ensures that the construction of τ is consistent with the strategy σ_1^* . The third property ensures that when we follow a branch of the tree, the value of τ decreases along it, this to guarantee that the target set is actually reached. The fourth condition ensures that when we follow a branch of the tree, at the end, the cost is below **c**. The most important of them are summarized in Figure 4.



Fig. 4. Labeling function associated with the strategy tree $\mathcal{T}_{\sigma_1^*}$

In order to prove Proposition 12, we need some technical results about the sets $I^k(v)$ and the functions f_v^* . For all $v \notin F$, for all $\mathbf{x} \in I^*(v) \setminus \{\infty\}$, we introduce the notation $\operatorname{Ind}_v^{\mathbf{x}}$ to denote the first index such that $\mathbf{x} \in I^{\operatorname{Ind}_v^{\mathbf{x}}}(v)$.

Lemma 3. For all $v \notin F$, if $\mathbf{x} \in I^*(v) \setminus \{\infty\}$ then, for all $\ell \geq \operatorname{Ind}_v^{\mathbf{x}}, \mathbf{x} \in I^{\ell}(v)$.

This lemma states that if a cost profile \mathbf{x} is in the fixpoint $I^*(v)$ for some v then, this cost profile stays in $I^k(v)$ from its first appearance to the fixpoint.

Proof. Let $v \notin F$ and let $\mathbf{x} \in I^*(v) \setminus \{\infty\}$. In the rest of the proof we set $n = \operatorname{Ind}_v^{\mathbf{x}}$. To obtain a contradiction, let us assume that there exists ℓ such that $\ell > n$ such that $\mathbf{x} \notin I^{\ell}(v)$.

Because $\ell > n$ and by Proposition 3, $\mathbf{x} \in \text{Ensure}^{\ell}(v)$. Since, by Proposition 4, $\mathbf{x} \notin I^{\ell}(v) = \min(\text{Ensure}^{\ell}(v))$, there exists $\mathbf{x}^* \in \text{Ensure}^{\ell}(v)$ such that $\mathbf{x}^* < \mathbf{x}$. Once again, by Proposition 3, $\mathbf{x}^* \in \text{Ensure}^*(v)$. But, we have assumed that $\mathbf{x} \in I^*(v)$ and by Theorem 8 we have that $I^*(v) = \min(\text{Ensure}^*(v))$. Thus $\mathbf{x}^* < \mathbf{x}$ and $\mathbf{x}^* \in \text{Ensure}^*(v)$ leads to a contradiction with the minimality of \mathbf{x} in Ensure(v).

Lemma 4. For all $v \in V_1 \setminus F$, for all $\mathbf{x} \in I^*(v) \setminus \{\infty\}$, if $(v', \mathbf{x}') = f_v^*(\mathbf{x})$ then, $\mathbf{x}' \in I^*(v')$ and $\mathbf{x}' = \mathbf{x} - \mathbf{w}(v, v')$.

Proof. In the proof we set $n = \text{Ind}_v^{\mathbf{x}}$.

By construction and Proposition 4, $\mathbf{x} = \mathbf{x}' + \mathbf{w}(v, v')$, $\mathbf{x} \in \mathbf{I}^n(v) = \text{minimal}(\text{Ensure}^n(v))$ and $\mathbf{x}' \in \mathbf{I}^{n-1}(v') = \text{minimal}(\text{Ensure}^{n-1}(v'))$. The second part of the assertion is already proved. Let us prove the other one.

In order to obtain a contradiction, let us assume that there exists ℓ such that $n-1 < \ell \leq k^*$ and $\mathbf{x}' \notin \mathbf{I}^{\ell}(v')$.

Since $n-1 < \ell$ and by Proposition 3, we have that $\mathbf{x}' \in \text{Ensure}^{\ell}(v')$. But as $\mathbf{x}' \notin I^{\ell}(v')$ and $I^{\ell}(v') = \text{minimal}(\text{Ensure}^{\ell}(v'))$ (by Proposition 4), that means that there exists $\mathbf{y}' \in \text{Ensure}^{\ell}(v')$ such that $\mathbf{y}' < \mathbf{x}'$. It follows that $\mathbf{y}' + \mathbf{w}(v, v') < \mathbf{x}' + \mathbf{w}(v, v') = \mathbf{x}$ and so $\mathbf{y}' + \mathbf{w}(v, v') \in \text{Ensure}^{\ell+1}(v)$.

Because $n < \ell+1$, thanks to Lemma 3, we have that $\mathbf{x} \in \mathbf{I}^{\ell+1}(v)$. Moreover, by Proposition 4, $\mathbf{I}^{\ell+1}(v) = \min(\operatorname{Ensure}^{\ell+1}(v))$. Thus because $\mathbf{y}' + \mathbf{w}(v, v) < \mathbf{x}$ and $\mathbf{y}' + \mathbf{w}(v, v') \in \operatorname{Ensure}^{\ell+1}(v)$, we obtain a contradiction with the fact that \mathbf{x} is minimal in $\operatorname{Ensure}^{\ell+1}(v)$.

We are now able to prove Proposition 12.

Proof (Proof of Proposition 12).

Let $u \in V$ and $\mathbf{c} \in I^*(u) \setminus \{\infty\}$. Let $\mathcal{T}^* = \mathcal{T}_{\sigma_1^*}$. We define τ and prove Invariant (1) to (4) step by step, by induction on the length of $h \in \mathcal{T}^*$.

Base case If h = uv for some $v \in V$.

- If $u \in V_1$: We define $\tau(uv) = f_u^*(\tau(u))[2] = f_u^*(\mathbf{c})[2]$. By construction $\tau(u) = \mathbf{c} \in \mathbf{I}^*(u)$ thus $f_u^*(\tau(u))$ is well defined. Since $u \in V_1$, $v = \sigma_1^*(u)$ and by definition of σ_1^* , $v = f_u^*(\mathbf{x})[1]$ where $\mathbf{x} = \min_{\leq_{\mathbf{L}}} \{\mathbf{x}' \in \mathbf{I}^*(u) \mid \mathbf{x}' \lesssim \mathbf{c} - \mathbf{Cost}(u) \land \mathbf{x}' \leq_{\mathbf{L}} \mathbf{c} - \mathbf{Cost}(u) \} = \min_{\leq_{\mathbf{L}}} \{\mathbf{x}' \in \mathbf{I}^*(u) \mid \mathbf{x}' \lesssim \mathbf{c} \land \mathbf{x}' \leq_{\mathbf{L}} \mathbf{c}\} = \min_{\leq_{\mathbf{L}}} C(u)$. Since $\mathbf{c} \in C(u)$, $\mathbf{x} \in \mathbf{I}^*(u)$. But $\mathbf{x}, \mathbf{c} \in \mathbf{I}^*(u) = \min(\mathbf{c})(u)$, by Theorem 8, implies $\mathbf{x} = \mathbf{c}$. It follows that $v = f_u^*(\mathbf{c})[1] = f_u^*(\tau(u))[1]$. Consequently, Invariants (1) and (2) are satisfied.

Since $\tau(uv) = f_u^*(\tau(u))[2]$, by Lemma 4, $\tau(uv) = \tau(u) - \mathbf{w}(u, v)$. That implies Invariant (3).

Since $\tau(uv) = \tau(u) - \mathbf{w}(u, v)$, $\tau(u) = \mathbf{c}$ and $\mathbf{w}(u, v) = \mathbf{Cost}(uv)$, we have that $\tau(uv) \leq \mathbf{c} - \mathbf{Cost}(uv)$ and $\tau(uv) \leq_{\mathbf{L}} \mathbf{c} - \mathbf{Cost}(uv)$. Thus, $\tau(uv) \in \{\mathbf{x}' \in \mathbf{I}^*(v) \mid \mathbf{x}' \leq \mathbf{c} - \mathbf{Cost}(uv) \land \mathbf{x}' \leq_{\mathbf{L}} \mathbf{c} - \mathbf{Cost}(uv)\} = \mathcal{C}(uv)$. It remains to prove that $\tau(uv) = \min_{\leq_{\mathbf{L}}} \mathcal{C}(uv)$. By contradiction, we assume that there exists $\mathbf{y} \in \mathbf{I}^*(v)$ such that (i) $\mathbf{y} \leq \mathbf{c} - \mathbf{Cost}(uv)$, (ii) $\mathbf{y} \leq_{\mathbf{L}} \mathbf{c} - \mathbf{Cost}(uv)$ and (iii) $\mathbf{y} <_{\mathbf{L}} \tau(uv)$. Let us recall that $\tau(uv) = \tau(u) - \mathbf{w}(u, v)$ and $\mathbf{w}(u, v) =$ $\mathbf{Cost}(uv)$. Therefore, by (i), we have that $\mathbf{y} \leq \mathbf{c} - \mathbf{w}(u, v) = \tau(u) - \mathbf{w}(u, v) =$ $\tau(uv)$. Finally, as $\mathbf{y}, \tau(uv) \in \mathbf{I}^*(v) = \minimal(\mathrm{Ensure}_{\leq}(v))$, by Theorem 8, $\mathbf{y} = \tau(uv)$ which is a contradiction with (iii). That concludes the proof of Invariant (4).

- If $u \in V_2$: we define $\tau(uv) = \mathbf{x}$ where $\mathbf{x} = \min_{\leq_{\mathrm{L}}} \{ \mathbf{x}' \in \mathrm{I}^*(v) \mid \mathbf{x}' \lesssim \mathbf{c} - \mathbf{Cost}(uv) \land \mathbf{x}' \leq_{\mathrm{L}} \mathbf{c} - \mathbf{Cost}(uv) \} = \min_{\leq_{\mathrm{L}}} \mathcal{C}(uv).$

Since $u \in V_2$ and $\tau(u) = \mathbf{c} \in I^*(u)$, by Corollary 1, there exists $\mathbf{x}'' \in I^*(v)$ such that $\mathbf{x}'' + \mathbf{w}(u, v) \lesssim \tau(u) = \mathbf{c}$. That implies $\mathbf{x}'' \leq_{\mathrm{L}} \mathbf{c} - \mathbf{Cost}(uv)$ as $\mathbf{Cost}(uv) = \mathbf{w}(u, v)$. In particular, $\mathcal{C}(uv) \neq \emptyset$ and so $\tau(uv) \in \mathbf{I}^*(v)$ and Invariants (1) and (4) hold. Since $u \in V_2$, Invariant (2) has not to be satisfied.

It remains to prove Invariant (3). Since $\tau(uv) \in \mathcal{C}(uv), \tau(uv) \leq_{\mathrm{L}} \mathbf{c} - \mathbf{Cost}(uv) =$ $\tau(u) - \mathbf{w}(u, v).$

Induction Hypothesis Let us assume that Invariant (1) to (4) hold for all $hv \in \mathcal{T}^*$ such that $|hv| \leq k$.

Let us now prove that for all $hvv' \in \mathcal{T}^*$ such that |hvv'| = k+1 these invariants are still satisfied.

- If $v \in V_1$: we define $\tau(hvv') = f_v^*(\tau(hv))[2]$. By induction hypothesis $\tau(hv) \in$ $I^*(v)$ thus $f_v^*(\tau(hv))$ is well defined. By definition of σ_1^* , we have that v' = $\sigma_1^*(hv) = f_v^*(\mathbf{x})[1]$ with $\mathbf{x} = \min_{\leq \mathbf{L}} \{ \mathbf{x}' \in \mathbf{I}^*(v) \mid \mathbf{x}' \lesssim \mathbf{c} - \mathbf{Cost}(hv) \land$ $\mathbf{x}' \leq_{\mathrm{L}} \mathbf{c} - \mathbf{Cost}(hv)$ and $\mathbf{x} = \tau(hv)$ by induction hypothesis.

Moreover by Lemma 4, $\tau(hvv') \in I^*(v')$. That proves Invariant (1) and (2). Invariant (3) is obtain thanks to the fact that $\tau(hvv') = \tau(hv) - \mathbf{w}(v, v')$ (by Lemma 4).

It remains to prove Invariant (4). Thanks to the induction hypothesis we obtain:

$$\tau(hvv') = \tau(hv) - \mathbf{w}(v, v') \leq_{\mathrm{L}} \mathbf{c} - \mathbf{Cost}(hv) - \mathbf{w}(v, v') = \mathbf{c} - \mathbf{Cost}(hvv')$$

and

$$\tau(hvv') = \tau(hv) - \mathbf{w}(v, v') \lesssim \mathbf{c} - \mathbf{Cost}(hv) - \mathbf{w}(v, v') = \mathbf{c} - \mathbf{Cost}(hvv').$$

Let us assume now that $\tau(hvv') \neq \min_{\leq L} \{ \mathbf{x}' \in I^*(v') \mid \mathbf{x}' \lesssim \mathbf{c} - \mathbf{Cost}(hvv') \land \}$ $\mathbf{x}' \leq_{\mathrm{L}} \mathbf{c} - \mathbf{Cost}(hvv')$. That means that there exists $\mathbf{y}' \in \mathrm{I}^*(v')$ such that $\mathbf{y}' \lesssim \mathbf{c} - \mathbf{Cost}(hvv'), \, \mathbf{y}' \leq_{\mathrm{L}} \mathbf{c} - \mathbf{Cost}(hvv') \text{ and } \mathbf{y}' <_{\mathrm{L}} \tau(hvv').$

Then $\mathbf{y}' + \mathbf{w}(v, v') \in \uparrow \mathbf{I}^*(v)$, thus there exists $\mathbf{z}' \in \mathbf{I}^*(v)$ such that $\mathbf{z}' \leq \mathbf{y}' + \mathbf{z}' \leq \mathbf{y}' \leq \mathbf{z}' \leq \mathbf{$ $\mathbf{w}(v, v')$. Since $\mathbf{z}' \leq \mathbf{y}' + \mathbf{w}(v, v')$ implies $\mathbf{z}' \leq_{\mathrm{L}} \mathbf{y}' + \mathbf{w}(v, v')$, we have that $\mathbf{z}' \leq_{\mathrm{L}} \mathbf{y}' + \mathbf{w}(v, v')$ $\mathbf{c} - \mathbf{Cost}(hv)$ and $\mathbf{z}' \leq_{\mathrm{L}} \mathbf{c} - \mathbf{Cost}(hv)$. Since $\mathbf{y}' <_{\mathrm{L}} \tau(hvv') = \tau(hv) - \mathbf{w}(v, v')$, that leads to $\mathbf{z}' <_{\mathrm{L}} \tau(hv)$ which is a contradiction with the induction hypothesis $\tau(hv) = \min_{\leq_{\mathrm{L}}} \{ \mathbf{x}' \in \mathrm{I}^*(v) \mid \mathbf{x}' \lesssim \mathbf{c} - \mathbf{Cost}(hv) \land \mathbf{x}' \leq_{\mathrm{L}} \mathbf{c} - \mathbf{Cost}(hv) \}.$

- If $v \in V_2$: we define $\tau(hvv') = \mathbf{x}$ where $\mathbf{x} = \min_{\leq_{\mathbf{L}}} \{ \mathbf{x}' \in \mathbf{I}^*(v') \mid \mathbf{x}' \lesssim$ $\mathbf{c} - \mathbf{Cost}(hvv') \land \mathbf{x}' \leq_{\mathrm{L}} \mathbf{c} - \mathbf{Cost}(hvv') \}$. Since $v \in V_2$ and $\tau(hv) \in \mathrm{I}^*(v)$, by Corollary 1, there exists $\mathbf{x}'' \in \mathbf{I}^*(v')$ such that $\mathbf{x}'' + \mathbf{w}(v, v') \lesssim \tau(hv)$. Which implies $\mathbf{x}'' + \mathbf{w}(v, v') \leq_{\mathrm{L}} \tau(hv)$. Moreover, by induction hypothesis, $\mathbf{x}'' \leq \mathbf{c} - \mathbf{Cost}(hv) - \mathbf{w}(v, v') = \mathbf{c} - \mathbf{Cost}(hvv') \text{ and } \mathbf{x}'' \leq_{\mathrm{L}} \mathbf{c} - \mathbf{Cost}(hvv').$ Therefore $\tau(hvv')$ and \mathbf{x}'' are in the set $\mathcal{C}(hvv') = \{\mathbf{x}' \in \mathbf{I}^*(v') \mid \mathbf{x}' \lesssim$

 $\mathbf{c} - \mathbf{Cost}(hvv') \wedge \mathbf{x}' \leq_{\mathrm{L}} \mathbf{c} - \mathbf{Cost}(hvv')\}$. So, in particular $\tau(hvv') \in \mathrm{I}^*(v')$ and Invariant (1) is satisfied. Moreover, as $\tau(hvv')$ is the minimum of the elements of the set $\mathcal{C}(hvv')$, we have that $\tau(hvv') \leq_{\mathrm{L}} \mathbf{x}'' \leq_{\mathrm{L}} \tau(hv) - \mathbf{w}(v,v')$. We can conclude that Invariants 3 and 4 are satisfied. As $v \in V_2$, the second invariant has not to be satisfied.

Before proving Theorem 5 we still need two technical results.

Lemma 5. For all $v \in V_1 \setminus F$, for all $\mathbf{x} \in I^*(v) \setminus \{\infty\}$, if $(v', \mathbf{x}') = f_v^*(\mathbf{x})$ then, $\operatorname{Ind}_{v'}^{\mathbf{x}'} < \operatorname{Ind}_v^{\mathbf{x}}$.

Proof. We set $n = \operatorname{Ind}_{v}^{\mathbf{x}}$ and $n' = \operatorname{Ind}_{v'}^{\mathbf{x}'}$.

By construction, we have that $\mathbf{x} = \mathbf{x}' + \mathbf{w}(v, v')$, $\mathbf{x} \in \mathbf{I}^n(v)$ and $\mathbf{x}' \in \mathbf{I}^{n-1}(v')$. Thus, $n' \leq n-1$ holds by definition of $\operatorname{Ind}_{v'}^{\mathbf{x}'}$.

Lemma 6. For all $v \in V_2 \setminus F$, for all $\mathbf{x} \in I^*(v) \setminus \{\infty\}$, for all $v' \in \operatorname{Succ}(v)$ and for all $\mathbf{x}' \in I^*(v')$ such that $\mathbf{x}' + \mathbf{w}(v, v') \leq_{\mathrm{L}} \mathbf{x}$, either, (i) $\mathbf{x}' <_{\mathrm{L}} \mathbf{x}$ or, (ii) $\operatorname{Ind}_{v'}^{\mathbf{x}'} < \operatorname{Ind}_{v}^{\mathbf{x}}$.

Proof. Let $v \in V_2 \setminus F$, $\mathbf{x} \in I^*(v) \setminus \{\infty\}$, $v' \in Succ(v)$ and $\mathbf{x}' \in I^*(v')$ such that $\mathbf{x}' + \mathbf{w}(v, v') \leq_L \mathbf{x}$.

To obtain a contradiction, we assume that $\neg(\mathbf{x}' <_{\mathrm{L}} \mathbf{x})$ and $\mathrm{Ind}_{v'}^{\mathbf{x}'} \ge \mathrm{Ind}_{v}^{\mathbf{x}}$. Since $\mathbf{x}' \leq_{\mathrm{L}} \mathbf{x}$ by hypothesis, $\neg(\mathbf{x}' <_{\mathrm{L}} \mathbf{x})$ implies $\mathbf{x}' = \mathbf{x}$. Therefore, $\mathbf{w}(v, v') = \mathbf{0}$.

Let $n = \operatorname{Ind}_{v}^{\mathbf{x}}$, by definition $\mathbf{x} \in I^{n}(v)$ and by Proposition 4, $\mathbf{x} \in \operatorname{Ensure}^{n}(v)$. Since $\mathbf{w}(v, v') = 0$, $\mathbf{x}' = \mathbf{x} \in \operatorname{Ensure}^{n-1}(v')$. In conclusion, the contradiction we were looking for is given by $\operatorname{Ind}_{v'}^{\mathbf{x}'} \leq n-1 < \operatorname{Ind}_{v}^{\mathbf{x}}$.

We are now able to prove Theorem 5. This proof exploit the notions of tree and strategy tree already defined in Appendix A.1.

Proof (Proof of Theorem 5). Let $u \in V$ and $\mathbf{c} \in I^*(u) \setminus \{\infty\}$. Let $\sigma_1^* \in \Sigma_1^u$ as defined in Definition 4. Let us consider the strategy tree $\mathcal{T}_{\sigma_1^*}$.

The first step of the proof is to prove that all branches of $\mathcal{T}_{\sigma_1^*}$ are finite and end with a node *n* such that $\text{Last}(n) \in \text{F}$.

Let us proceed by contradiction and assume that there exists a branch $b = n_0 n_1 n_2 \dots$ which is infinite. By Statement 3 of Proposition 12, we know that the sequence $(\tau(n_k))_{k \in \mathbb{N}}$ is non increasing w.r.t. \leq_{L} and it is lower bounded by **0**. It follows that:

$$\exists \xi \in \mathbb{N} \text{ such that } \forall \ell \in \mathbb{N}, \ \tau(n_{\xi}) = \tau(n_{\xi+\ell}).$$
(8)

Either there exists $\ell \in \mathbb{N}$ such that $\text{Last}(n_{\xi+\ell}) \in \mathcal{F}$ which contradicts the fact that branch b is infinite. Or, for all $\ell \in \mathbb{N}$,

$$\operatorname{Ind}_{\operatorname{Last}(n_{\xi+\ell+1})}^{\tau(n_{\xi+\ell+1})} < \operatorname{Ind}_{\operatorname{Last}(n_{\xi+\ell})}^{\tau(n_{\xi+\ell})}.$$

Let $hv = n_{\xi+\ell+1}$ then, $h = n_{\xi+\ell}$.

- If $\text{Last}(h) \in V_1$, by Statement 2 of Proposition 12 we have $(v, \tau(hv)) = f^*_{\text{Last}(h)}(\tau(h))$. Moreover, by Lemma 5, we obtain $\text{Ind}_v^{\tau(hv)} < \text{Ind}_{\text{Last}(h)}^{\tau(h)}$.
- If $\operatorname{Last}(h) \in V_2$, by Statement 3 of Proposition 12, we have that $\tau(hv) + \mathbf{w}(\operatorname{Last}(h), v) \leq_{\mathrm{L}} \tau(h)$. Additionally, by Lemma 6, either $\tau(hv) <_{\mathrm{L}} \tau(h)$ which is assumed to be impossible by Eq. (8) or $\operatorname{Ind}_{v}^{\tau(hv)} < \operatorname{Ind}_{\operatorname{Last}(h)}^{\tau(h)}$.

That means that the sequence $\left(\operatorname{Ind}_{\operatorname{Last}(n_{\xi+\ell})}^{\tau(n_{\xi+\ell})}\right)_{\ell\in\mathbb{N}}$ is strictly decreasing w.r.t. the classical order < on the natural numbers and is lower bounded by 0. It follows that such an infinite branch cannot exist.

In what precedes, we proved that F is reached whatever the behavior of \mathcal{P}_2 , in particular each branch $b = n_0 n_1 \dots n_k$ ends in a node n_k which is a leaf, and such that $\text{Last}(n_k) \in F$. Thus, $\tau(n_k) = 0$. Moreover, if $n_k = hv$, the cost of the branch corresponds to Cost(hv) and by Proposition 12, we have that $\tau(hv) \leq \mathbf{c} - \text{Cost}(hv)$. That inequality implies that $\text{Cost}(hv) \leq \mathbf{c}$.

In the first part of this section we proved, given $u \in V$ and $c \in I^*(u) \setminus \{\infty\}$, how to obtain a startegy σ_1^* of \mathcal{P}_1 that ensures **c** from u (see Definition 4 and Theorem 5). Thus, in particular, σ_1^* is both a lexico-optimal strategy from u and a **c**-Pareto-optimal strategy from u. However, σ_1^* requires finite-memory.

In the remainder of this section, we prove that if we consider the lexicographic order, the strategy ϑ_1^* , given in Proposition 5, is a positional lexico-optimal strategy from u.

Proposition 5. If \leq is the lexicographic order, for $u \in V$ and $\mathbf{c} \in I^*(u) \setminus \{\infty\}$, the strategy ϑ_1^* defined as: for all $hv \in \text{Hist}_1(u)$, $\vartheta_1^*(hv) = f_v^*(\mathbf{x})[1]$ where \mathbf{x} is the unique cost profile in $I^*(v)$, is a positional lexico-optimal strategy from u.

We now prove that the strategy ϑ_1^* , as defined in Proposition 5, is a lexicooptimal strategy from u. We proceed in the same way as we proved that σ_1^* ensures **c** from u: we prove that a labeling function of the strategy tree $\mathcal{T}_{\vartheta_1^*}$ exists and has the same kind of properties as in Proposition 12. From that follows, for the same arguments as these exploited in the proof of Theorem 5, that ϑ_1^* is a lexico-optimal strategy from u.

Proposition 13. If \leq is the lexicographic order, for $u \in V$ and $\mathbf{c} \in I^*(u) \setminus \{\infty\}$, if $\mathcal{T}_{\tau_1^*}$ is the strategy tree of the strategy ϑ_1^* as defined in Proposition 5 then, there exists a labeling function $\tau : \mathcal{T}_{\vartheta_1^*} \longrightarrow \mathbb{N}^d$ such that, $\tau(u) = \mathbf{c} \in I^*(u)$ and, for all $hv \in \mathcal{T}_{\vartheta_1^*}$ such that $|hv| \geq 1$:

- 1. $\tau(hv) \in I^*(v);$
- 2. If $\operatorname{Last}(h) \in V_1$ then, $(v, \tau(hv)) = f^*_{\operatorname{Last}(h)}(\tau(h));$
- 3. $\tau(hv) \leq_{\mathrm{L}} \tau(h) \mathbf{w}(\mathrm{Last}(h), v);$
- 4. $\tau(hv) \leq_{\mathrm{L}} \mathbf{c} \mathbf{Cost}(hv).$

Proof. Let $u \in V$ and $\mathbf{c} \in I^*(u) \setminus \{\infty\}$. Let $\mathcal{T}^* = \mathcal{T}_{\vartheta_1^*}$. We define τ and prove Invariant (1) to (4) step by step, by induction on the length of $h \in \mathcal{T}^*$.

Before beginning the proof, we recall that, because we consider the lexicographic order, each time we consider some $\mathbf{x}, \mathbf{x}' \in I^*(v)$ for some $v \in V$, we have that $\mathbf{x} = \mathbf{x}'$ since $I^*(v)$ is a singleton.

Base case If h = uv for some $v \in V$.

- If $u \in V_1$: we define $\tau(uv) = f_u^*(\tau(u))[2]$. By hypothesis, $\tau(u) = \mathbf{c} \in I^*(u)$ so $f_u^*(\tau(u))$ is well defined. Let us prove that the invariants are satisfied.
 - Invariant (1). By Lemma 4, as $\tau(uv) = f_u^*(\tau(u))[2], \tau(uv) \in I^*(v)$.
 - Invariant (2). By construction of \mathcal{T}^* , $v = \vartheta_1^*(u)$ and by definition of ϑ_1^* , $\vartheta_1^*(u) = f_1^*(\mathbf{x})[1]$ where \mathbf{x} is the only cost profile in $\mathbf{I}^*(u)$. Thus, $\mathbf{x} = \mathbf{c}$. Finally, since $\tau(u) = \mathbf{c}$, we obtain that $v = f_u^*(\tau(u))[1]$.
 - Invariants (3) and (4). Since $\tau(uv) = f_u^*(\tau(u))[2]$, by Lemma 4, $\tau(uv) = \tau(u) \mathbf{w}(u, v)$. Moreover, because $\tau(u) = \mathbf{c}$ and $\mathbf{w}(u, v) = \mathbf{Cost}(uv)$, we also have that $\tau(uv) = \mathbf{c} \mathbf{Cost}(uv)$.
- If $u \in V_2$: we define $\tau(uv) = \mathbf{x}$ where \mathbf{x} is the only cost profile in $I^*(v)$. Let us prove that the invariants are satisfied.
 - Invariant 1. We have that $\tau(uv) \in I^*(v)$ by construction.
 - Invariant 2. It does not have to be satisfied since $u \in V_2$.
 - Invariants 3 and 4. We have that $v \in \text{Succ}(u)$ and $\tau(u) \in I^*(u)$, thus by Corollary 1, there exists $\mathbf{x}' \in I^*(v)$ such that $\mathbf{x}' + \mathbf{w}(u, v) \leq_{\mathrm{L}} \tau(u)$. But, $\mathbf{x}, \mathbf{x}' \in I^*(v)$ implies that $\mathbf{x} = \mathbf{x}'$ ($I^*(v)$ is a singleton). Moreover $\tau(uv) = \mathbf{x}$, it follows that $\tau(uv) \leq_{\mathrm{L}} \tau(u) - \mathbf{w}(u, v)$. Finally, since $\tau(u) = \mathbf{c}$ and $\mathbf{w}(u, v) = \mathbf{Cost}(uv)$, we also obtain that $\tau(uv) \leq_{\mathrm{L}} \mathbf{c} - \mathbf{Cost}(uv)$.

Induction Hypothesis Let us assume that Invariant (1) to (4) hold for all $hv \in \mathcal{T}^*$ such that $|hv| \leq k$.

Let us now prove that for all $hvv' \in \mathcal{T}^*$ such that |hvv'| = k + 1 those invariants are still satisfied.

- If $v \in V_1$: we define $\tau(hvv') = f_v^*(\tau(hv))[2]$. By induction hypothesis, $\tau(hv) \in I^*(v)$, so $f_v^*(\tau(hv))$ is well defined. We have that $v' = \vartheta_1^*(hv)$ and by definition of ϑ_1^* , $\vartheta_1^*(hv) = f_v^*(\mathbf{x})[1]$ with $\mathbf{x} \in I^*(v)$. Since we consider the lexicographic order, $I^*(v)$ is a singleton and $\mathbf{x} = \tau(hv)$. Moreover, by Lemma 4, $\tau(hvv') \in I^*(v')$. It follows that Invariants (1) and (2) are satisfied. Invariant (3) is obtained thanks to Lemma 4 and the fact that $\tau(hvv') = \tau(hv) - \mathbf{w}(v, v')$. It remains to prove that $\tau(hvv') \leq_{\mathbf{L}} \mathbf{c} - \mathbf{Cost}(hvv')$ (Invariant 4). By induction hypothesis, we know that $\tau(hv) \leq_{\mathbf{L}} \mathbf{c} - \mathbf{Cost}(hv)$. Since
 - $\tau(hvv') = \tau(hv) \mathbf{w}(v, v')$, we have: $\tau(hvv') = \tau(hv) \mathbf{w}(v, v') \leq_{\mathrm{L}} \mathbf{c} \mathbf{Cost}(hv) \mathbf{w}(v, v')$ by induction hypothesis. The fact that $\mathbf{Cost}(hv) \mathbf{w}(v, v') = \mathbf{Cost}(hvv')$ concludes the proof.
- if $v \in V_2$: we define $\tau(hvv') = \mathbf{x}$ where \mathbf{x} is the only cost profile in $I^*(v')$. Notice that in this way, $\tau(hvv') \in I^*(v')$ (Invariant 1) is already satisfied. Since $v \in V_2$, we do not have to check if Invariant (2) holds. As by induction hypothesis $\tau(hv) \in I^*(v)$, we have by Corollary 1 that there exists $\mathbf{x}' \in I^*(v')$ such that $\mathbf{x}' + \mathbf{w}(v, v') \leq_{\mathrm{L}} \tau(hv)$. But since we consider the

lexicographic order, the set $I^*(v')$ is a singleton, meaning that $\mathbf{x} = \mathbf{x}'$. The fact that $\mathbf{x} = \tau(hvv')$ allows to conclude that Invariant 3 is satisfied. By what we have just proved $\tau(hvv') \leq_{\mathrm{L}} \tau(hv) - \mathbf{w}(v, v')$ and $\tau(hv) - \mathbf{w}(v, v') \leq_{\mathrm{L}} \mathbf{c} - \mathbf{Cost}(hv) - \mathbf{w}(v, v') = \mathbf{c} - \mathbf{Cost}(hvv')$ by induction hypothesis. It is exactly what Invariant (4) states.

B Additional content of Section 4: Constrained Existence

Proposition 14. If \leq is the componentwise order, the constrained existence problem is PSPACE-complete, even if d = 2 and $|\mathbf{F}| = 1$.

PSpace-easiness of the constrained existence problem Proposition 2 allows us to prove that the constrained existence problem is in APTIME. The alternating Turing machine works as follows: all vertices of the game owned by \mathcal{P}_1 (resp. \mathcal{P}_2) correspond to disjunctive states (resp. conjunctive states). A path of length |V| is accepted if and only if, (i) the target set is reached along that path and (ii) the sum of the weights until a element of the target set is $\leq_{\rm C} \mathbf{x}$. If such a path exists, there exists a strategy of \mathcal{P}_1 that ensures the cost profile \mathbf{x} . This procedure is done in polynomial time and since APTIME = PSPACE, we get the result.

The hardness of Proposition 14 is obtained thanks to a polynomial reduction from the QUANTIFIED SUBSET-SUM problem which is proved PSPACEcomplete [12, Lemma 4]. We begin by providing some intuition on the PSPACEhardness and we conclude this section by providing a formal proof of this result.



Fig. 5. Initialized game used in the reduction for the PSPACE-hardness

PSpace-hardness of the constrained existence problem In the same spirit as for the QBF problem [11], the QUANTIFIED SUBSET-SUM problem can be seen as a two-player game in which two players (Player \exists and Player \forall) take turn in order to assign a value to the variables x_1, \ldots, x_n : Player \exists (resp. Player \forall) chooses the value of the variables under an existential quantifier (resp. universal quantifier). When a player assigns a value 1 to a variable x_k , $1 \le k \le n$, this

player selects the natural number a_k and he does not select it if x_k is assigned to 0. The goal of Player \exists is that the sum of the selected natural numbers is exactly equal to T while the goal of Player \forall is to avoid that. Thus, with this point of view, the formula Ψ is true if and only if Player \exists has a winning strategy.

In order to encode the equality presents in the Quantified Subset-Sum problem, we use the two inequality constraints in a two-player two-weighted game. The arena of the game is given in Figure 5, \mathcal{P}_1 aka Player \exists (resp. \mathcal{P}_2 aka Player \forall) owns the rounded (resp. rectangular) vertices corresponding to variables under an existential (resp. universal) quantifier. The target set is only composed of the vertex y. When a player assigns the value 1 to a variable x_k , $1 \leq k \leq n$, the resulting weight of this choice is $(a_k, 0)$, while if he assigns the value 0, the weight is $(0, a_k)$. In this way, if we sum all those weights, we have on the first component the sum of the selected natural numbers. Thereby, there exists a strategy of \mathcal{P}_1 that ensures the cost profile $(T, \sum_{1 \leq i \leq n} a_i - T)$ if and only if the formula Ψ is true.

Notice that as \mathcal{P}_1 can consider the previous assignations of variables x_1, \ldots, x_{k-1} to choose the assignation of a variable x_k to 0 or 1, the resulting strategy needs finite-memory.

Proof (Formal proof of the PSPACE-hardness).

For all odd (resp. even) numbers $k, 1 \leq k \leq n$, we denote by $f_k^{\exists} : \{0,1\}^{k-1} \longrightarrow \{0,1\}$ (resp. $f_k^{\forall} : \{0,1\}^{k-1} \longrightarrow \{0,1\}$) the valuation of the variable x_k taking into account the valuation of previous variables x_1, \ldots, x_{k-1} . We assume that $f_1^{\exists} : \emptyset \longrightarrow \{0,1\}$. Given a sequence $f^{\exists} = f_1^{\exists}, f_3^{\exists}, \ldots$ and a sequence $f^{\forall} = f_2^{\forall}, f_4^{\forall}, \ldots$, we define the function $\nu_{(f^{\exists}, f^{\forall})} : \{x_1, \ldots, x_n\} \longrightarrow \{0,1\}$ such that $\nu_{(f^{\exists}, f^{\forall})}(x_1) = f_1^{\exists}(\emptyset), \nu_{(f^{\exists}, f^{\forall})}(x_2) = f_2^{\forall}(\nu_{(f^{\exists}, f^{\forall})}(x_1)), \nu_{(f^{\exists}, f^{\forall})}(x_3) = f_3^{\exists}(\nu_{(f^{\exists}, f^{\forall})}(x_1)\nu_{(f^{\exists}, f^{\forall})}(x_2)), \ldots$ We also define the set $S(f^{\exists}, f^{\forall}) = \{p \mid \nu_{(f^{\exists}, f^{\forall})}(x_p) = 1\}$.

Thanks to these notations we rephrase the QUANTIFIED SUBSET-SUM problem as: does there exist a sequence of functions $f^{\exists} = f_1^{\exists}, f_3^{\exists}, \ldots$ such that for all sequences $f^{\forall} = f_2^{\forall}, f_4^{\forall}, \ldots, \sum_{p \in S(f^{\exists}, f^{\forall})} a_p = T$?

We now describe the reduction from the QUANTIFIED SUBSET-SUM problem to the constrained existence problem.

The $\mathcal{A}_2 = (V_1, V_2, E, \mathbf{w})$ of the initialized 2-weighted reachability game $(\mathcal{G}_2, v_0) = (\mathcal{A}_2, F, \mathbf{Cost})$ is given in Figure 5. Formally, the game is build as follows:

- $-V_1$ is composed by the following vertices: a vertex y, for each variable x_p under an existential quantifier there is a vertex x_p and finally for all $a_p \in I$ there are two vertices x_p^0 and x_p^1 ; $-V_2$ is the set of vertices denoted by x_p such that the variable x_p is under an
- $-V_2$ is the set of vertices denoted by x_p such that the variable x_p is under an universal quantifier. Notice that in Figure 5 we assume that n is odd, and so x_n is under an existential quantifier.
- E is composed of the edges of the form:
 - (x_{ℓ}, x_{ℓ}^0) and (x_{ℓ}, x_{ℓ}^1) , for all $1 \leq \ell \leq n$;
 - $(x_{\ell}^1, x_{\ell+1})$ and $(x_{\ell}^0, x_{\ell+1})$, for all $1 \le \ell \le n-1$;

• $(x_n^1, y), (x_n^0, y) \text{ and } (y, y).$

- the weight function \mathbf{w} is defined as:
 - $\mathbf{w}(x_{\ell}, x_{\ell}^{1}) = (a_{\ell}, 0) \text{ and } \mathbf{w}(x_{\ell}, x_{\ell}^{0}) = (0, a_{\ell}), \text{ for all } 1 \le \ell \le d;$
 - for all the other edges $e \in E$, $\mathbf{w}(e) = (0, 0)$.
- $-F = \{y\};$
- $-v_0=x_1.$

We prove the following equivalence.

There exists a sequence $f^{\exists} = f_1^{\exists}, f_3^{\exists}, \ldots$ such that for all sequences $f^{\forall} = f_2^{\forall}, f_4^{\forall}, \ldots, \sum_{p \in S(f^{\exists}, f^{\forall})} a_p = T$ if and only if there exists a finite-memory strategy σ_1 of \mathcal{P}_1 from x_1 such that for all strategies σ_2 of \mathcal{P}_2 from x_1 , $\mathbf{Cost}(\langle \sigma_1, \sigma_2 \rangle_{x_1}) \leq_{\mathbf{C}} (T, \sum_{1 \leq p \leq n} a_p - T).$

Remark Before proving this equivalence, let us notice that:

- 1. For each sequence $f^{\exists} = f_1^{\exists}, f_3^{\exists}, \dots$ (resp. each sequence $f^{\forall} = f_2^{\forall}, f_4^{\forall}, \dots$), there exists a corresponding finite-memory strategy σ_1 of \mathcal{P}_1 (resp. σ_2 of \mathcal{P}_2) in (\mathcal{G}_2, x_1) ;
- 2. For each finite-memory strategy σ_1 of \mathcal{P}_1 (resp. each strategy σ_2 of \mathcal{P}_2), there exists a corresponding sequence $f^{\exists} = f_1^{\exists}, f_3^{\exists}, \dots$ (resp. $f^{\forall} = f_2^{\forall}, f_4^{\forall}, \dots$).

Construction of strategies of Statement 1. Let $f^{\exists} = f_1^{\exists}, f_3^{\exists}, \ldots$ and $f^{\forall} = f_2^{\forall}, f_4^{\forall}, \ldots$ We define σ_1 : for all $1 \leq \ell \leq n$ such that ℓ is odd, $\sigma_1(x_1) = x_1^i$ if $f_1^{\exists}(\emptyset) = i$ and $\sigma_1(x_1v_1x_2v_2\ldots x_\ell) = x_\ell^i$ if $f_\ell^{\exists}(\overline{v_1v_2}\ldots \overline{v_{\ell-1}}) = i$ with, for all $1 \leq p \leq \ell, v_p \in \{x_p^1, x_p^0\}$ and $\overline{v_p} = 1$ if $v_p = x_p^1$ and $\overline{v_p} = 0$ otherwise. The strategy σ_2 is defined exactly in the same way for all $1 \leq \ell \leq n$ such that ℓ is even, except that f_ℓ^{\exists} is replaced by f_ℓ^{\forall} .

Construction of strategies of Statement 2. Let σ_1 be a finite-memory strategy of \mathcal{P}_1 and σ_2 be a strategy of \mathcal{P}_2 . We build f^{\exists} as follows: $f_1^{\exists}(\emptyset) = i$ if $\sigma_1(x_1) = x_1^i$ and, for all $1 \leq \ell \leq n$ such that ℓ is odd, $f_\ell^{\exists}(\overline{v}_1 \dots \overline{v}_{\ell-1}) = i$ if $\sigma_1(x_1v_1x_2v_2\dots x_\ell) = x_\ell^i$ with for all $1 \leq p \leq \ell - 1$, $\overline{v}_p \in \{0, 1\}$ and $v_p = x_p^1$ if $\overline{v}_p = 1$ and $v_p = x_p^0$ otherwise. The f_ℓ^{\forall} are defined exactly in the same way for all $1 \leq \ell \leq n$ such that ℓ is even and by replacing σ_1 by σ_2 .

We come back to the proof of the equivalence. (\Rightarrow) We assume that there exists a sequence $f^{\exists} = f_1^{\exists}, f_3^{\exists}, \ldots$ such that for all sequences $f^{\forall} = f_2^{\forall}, f_4^{\forall}, \ldots, \sum_{p \in S(f^{\exists}, f^{\forall})} a_p = T$.

We consider σ_1 as defined previously (Remark, Statement 1). We have to prove that for all strategies σ_2 of \mathcal{P}_2 : $\mathbf{Cost}(\langle \sigma_1, \sigma_2 \rangle_{x_1}) \leq_{\mathbf{C}} (T, \sum_{1 .$

Let σ_2 be a strategy of \mathcal{P}_2 . As explained in Remark, Statement 2, we have that σ_2 corresponds to some sequence $f^{\forall} = f_2^{\forall}, f_4^{\forall}, \ldots$ Thus, by construction of the game arena and by hypothesis we have:

$$\operatorname{Cost}_1(\langle \sigma_1, \sigma_2 \rangle_{x_1}) = \sum_{p \in S(f^{\exists}, f^{\forall})} a_p = T \text{ and } \operatorname{Cost}_2(\langle \sigma_1, \sigma_2 \rangle_{x_1}) = \sum_{p \notin S(f^{\exists}, f^{\forall})} a_p = \sum_{1 \le p \le n} a_p - T.$$

(⇐) Let us assume that there exists a finite-memory strategy σ_1 of \mathcal{P}_1 such that for all strategies σ_2 of \mathcal{P}_2 , $\mathbf{Cost}(\langle \sigma_1, \sigma_2 \rangle_{x_1}) \leq_{\mathbf{C}} (T, \sum_{1 \leq p \leq n} a_p - T)$.

We define the sequence $f^{\exists} = f_1^{\exists}, f_3^{\exists}, \ldots$ as explained in Remark, Statement 2. Let $f^{\forall} = f_2^{\forall}, f_4^{\forall}, \ldots$, we have to prove that $\sum_{p \in S(f^{\exists}, f^{\forall})} a_p = T$. By Remark, Statement 1, the sequence f^{\forall} corresponds to a strategy σ_2 of \mathcal{P}_2 in (\mathcal{G}_2, x_1) . It follows by hypothesis and construction of the game arena:

 $- \operatorname{Cost}_1(\langle \sigma_1, \sigma_2 \rangle_{x_1}) = \sum_{p \in S(f^{\exists}, f^{\forall})} a_p \\ - \operatorname{Cost}_1(\langle \sigma_1, \sigma_2 \rangle_{x_1}) \le T$

and

$$-\operatorname{Cost}_{2}(\langle \sigma_{1}, \sigma_{2} \rangle_{x_{1}}) = \sum_{p \notin S(f^{\exists}, f^{\forall})} a_{p} = \sum_{1 \leq p \leq n} a_{p} - \sum_{p \in S(f^{\exists}, f^{\forall})} a_{p} - \operatorname{Cost}_{2}(\langle \sigma_{1}, \sigma_{2} \rangle_{x_{1}}) \leq \sum_{1 \leq p \leq n} a_{p} - T.$$

Thus, we can conclude that $\sum_{p \in S(f^{\exists}, f^{\forall})} a_p = T$.