Faster $O(|V|^2|E|W)$-Time Energy Algorithms for Optimal Strategy Synthesis in Mean Payoff Games

Carlo Comin[a,b], Romeo Rizzi[c]

[a] *Department of Mathematics, University of Trento, Trento, Italy*
[b] *LIGM, Université Paris-Est, Marne-la-Vallée, Paris, France*
[c] *Department of Computer Science, University of Verona, Verona, Italy*

Abstract

This study strengthens the links between Mean Payoff Games (MPGs) and Energy Games (EGs). Firstly, we offer a faster $O(|V|^2|E|W)$ pseudo-polynomial time and $\Theta(|V|+|E|)$ space deterministic algorithm for solving the Value Problem and Optimal Strategy Synthesis in MPGs. This improves the best previously known estimates on the pseudo-polynomial time complexity to:

$$O(|E|\log|V|)+\Theta\Big(\sum_{v\in V}\texttt{deg}_\Gamma(v)\cdot\ell_\Gamma(v)\Big)=O(|V|^2|E|W),$$

where $\ell_\Gamma(v)$ counts the number of times that a certain energy-lifting operator $\delta(\cdot,v)$ is applied to any $v\in V$, along a certain sequence of Value-Iterations on reweighted EGs; and $\texttt{deg}_\Gamma(v)$ is the degree of v. This improves significantly over a previously known pseudo-polynomial time estimate, i.e., $\Theta\big(|V|^2|E|W+\sum_{v\in V}\texttt{deg}_\Gamma(v)\cdot\ell_\Gamma(v)\big)$ (Comin and Rizzi, 2015, 2016), as the pseudo-polynomiality is now confined to depend solely on ℓ_Γ. Secondly, we further explore on the relationship between Optimal Positional Strategies (OPSs) in MPGs and Small Energy-Progress Measures (SEPMs) in reweighted EGs. It is observed that the space of all OPSs, $\texttt{opt}_\Gamma\Sigma^M_0$, admits a unique complete decomposition in terms of extremal-SEPMs in reweighted EGs. This points out what we called the "Energy-Lattice $\mathscr{X}^*_\Gamma$ associated to $\texttt{opt}_\Gamma\Sigma^M_0$". Finally, it is offered a pseudo-polynomial total-time recursive procedure for enumerating (w/o repetitions) all the elements of $\mathscr{X}^*_\Gamma$, and for computing the corresponding partitioning of $\texttt{opt}_\Gamma\Sigma^M_0$.

Keywords: Mean Payoff Games, Value Problem, Optimal Strategy Synthesis, Pseudo-Polynomial Time, Energy Games, Small Energy-Progress Measures.

1. Introduction

A *Mean Payoff Game* (MPG) is a two-player infinite game $\Gamma=(V,E,w,\langle V_0,V_1\rangle)$, which is played on a finite weighted directed graph, denoted $G^\Gamma\triangleq(V,E,w)$, where $w:E\to\mathbb{Z}$, the vertices of which are partitioned into two classes, V_0 and V_1, according to the player to which they belong.

At the beginning of the game a pebble is placed on some vertex $v_s\in V$, and then the two players, named Player 0 and Player 1, move it along the arcs ad infinitum. Assuming the pebble

Email addresses: `carlo.comin@unitn.it` (Carlo Comin), `romeo.rizzi@univr.it` (Romeo Rizzi)

is currently on some $v \in V_0$, then Player 0 chooses an arc $(v, v') \in E$ going out of v and moves the pebble to the destination vertex v'. Similarly, if the pebble is currently on some $v \in V_1$, it is Player 1 to choose an outgoing arc. The infinite sequence $v_s, v, v' \ldots$ of all the encountered vertices forms a *play*. In order to play well, Player 0 wants to maximize the limit inferior of the long-run average weight of the traversed arcs, i.e., to maximize $\liminf_{n\to\infty} \frac{1}{n}\sum_{i=0}^{n-1} w(v_i, v_{i+1})$, whereas Player 1 wants to minimize the $\limsup_{n\to\infty} \frac{1}{n}\sum_{i=0}^{n-1} w(v_i, v_{i+1})$. Ehrenfeucht and Mycielski (1979) proved that each vertex v admits a *value*, denoted $\texttt{val}^{\Gamma}(v)$, which each player can secure by means of a *memoryless* (or *positional*) strategy, i.e., one depending only on the current vertex position and not on the previous choices.

Solving an MPG consists in computing the values of all vertices (*Value Problem*) and, for each player, a positional strategy that secures such values to that player (*Optimal Strategy Synthesis*). The corresponding decision problem lies in $\mathsf{NP} \cap \mathsf{coNP}$ (Zwick and Paterson, 1996) and it was later shown to be in $\mathsf{UP} \cap \mathsf{coUP}$ (Jurdziński, 1998).

The problem of devising efficient algorithms for solving MPGs has been studied extensively in the literature. The first milestone was settled in Gurvich et al. (1988), in which it was offered an *exponential* time algorithm for solving a slightly wider class of MPGs called *Cyclic Games*. Afterwards, Zwick and Paterson (1996) devised the first deterministic procedure for computing values in MPGs, and optimal strategies securing them, within a pseudo-polynomial time and polynomial space. In particular, it was established an $O(|V|^3|E|W)$ upper bound for the time complexity of the Value Problem, as well as an upper bound of $O(|V|^4|E|W\log(|E|/|V|))$ for that of Optimal Strategy Synthesis (Zwick and Paterson, 1996).

Several research efforts have been spent in studying quantitative extensions of infinite games for modeling quantitative aspects of reactive systems (Chakrabarti et al., 2003; Bouyer et al., 2008; Brim et al., 2011). In this context quantities may represent, e.g., the power usage of an embedded component, or the buffer size of a networking element. These studies unveiled interesting connections with MPGs, and recently they have led to the design of faster procedures for solving them. In particular, Brim et al. (2011) devised a faster deterministic algorithm for solving the Value Problem and Optimal Strategy Synthesis in MPGs within $O\big(|V|^2|E|\,W\log(|V|\,W)\big)$ pseudo-polynomial time and polynomial space. Essentially, a binary search is directed by the resolution of multiple reweighted EGs. The determination of EGs comes from repeated applications of energy-lifting operators $\delta(\cdot, v)$ for any $v \in V$; these are all monotone functions defined on a complete lattice (i.e., the Energy-Lattice of a reweighted EG). At this point the correct termination is ensured by the Knaster–Tarski fixed-point theorem.

Recently, the worst-cast time complexity of the Value Problem and Optimal Strategy Synthesis was given an improved pseudo-polynomial upper bound (Comin and Rizzi, 2015, 2016). Those works focused on offering a simple proof of the improved time complexity bound. The algorithm there proposed, henceforth called Algorithm 0, had the advantage of being very simple; its existence made it possible to discover and analyze some of the underlying fundamental ideas, that ultimately led to the improved upper bound, more directly; it was shown appropriate to supersede (at least in the perspective of sharpened bounds) the above mentioned binary search by sort of a linear search that succeeds at amortizing all the energy-liftings throughout the computation. However, its running time turns out to be also $\Omega(|V|^2|E|W)$, the actual time complexity being $\Theta\big(|V|^2|E|\,W + \sum_{v\in V} \texttt{deg}_{\Gamma}(v) \cdot \ell^0_{\Gamma}(v)\big)$, where $\ell^0_{\Gamma}(v) \leq (|V|-1)|V|W$ denotes the total number of times that the energy-lifting operator $\delta(\cdot, v)$ is applied to any $v \in V$ by Algorithm 0.

After the appearance of those works, a way to overcome this issue was found.

Contribution. This study aims at strenghtening the relationship between MPGs and EGs.

Our results are summarized as follows.

- *A Faster $O(|V|^2|E|W)$-Time Algorithm for MPGs by Jumping through Reweighted EGs.*

We introduce a novel algorithmic scheme, named *Jumping* (Algorithm 1), which tackles on some further regularities of the problem, thus reducing the estimate on the pseudo-polynomial time complexity of MPGs to:

$$O(|E|\log|V|)+\Theta\Big(\sum_{v\in V}\texttt{deg}_\Gamma(v)\cdot\ell^1_\Gamma(v)\Big),$$

where $\ell^1_\Gamma(v)$ is the total number of applications of $\delta(\cdot,v)$ to $v\in V$ that are made by Algorithm 1; $\ell^1_\Gamma\leq(|V|-1)|V|W$ (worst-case; but experimentally, $\ell^1_\Gamma\ll\ell^0_\Gamma$), and the working space is $\Theta(|V|+|E|)$. Overall the worst-case complexity is still $O(|V|^2|E|W)$, but the pseudo-polynomiality is now confined to depend solely on the total number ℓ^1_Γ of required energy-liftings; this is not known to be $\Omega(|V|^2|E|W)$ generally, and future theoretical or practical advancements concerning the Value-Iteration framework for EGs could help reducing this metric. Under this perspective, theoretically, the computational equivalence between MPGs and EGs seems now like a bit more unfolded and subtle. In practice, Algorithm 1 allows us to reduce the magnitude of ℓ_Γ considerably, w.r.t. Comin and Rizzi (2015, 2016), and therefore the actual running time of the algorithm; our experiments suggest that $\ell^1_\Gamma\ll\ell^0_\Gamma$ holds for wide families of MPGs (see SubSection 4.4).

In summary, the present work offers a *faster* pseudo-polynomial time algorithm; theoretically the pseudo-polynomiality now depends only on ℓ^1_Γ, and in practice the actual performance also improves considerably w.r.t. Comin and Rizzi (2015, 2016). With hindsight, Algorithm 0 turned out to be a high-level description and the tip of a more technical and efficient underlying procedure. This is the first truly $O(|V|^2|E|W)$ time deterministic algorithm, for solving the Value Problem and Optimal Strategy Synthesis in MPGs, that can be effectively applied in practice (optionally, in interleaving with some other known sub-exponential time algorithms).

Indeed, a wide spectrum of different approaches have been investigated in the literature. For instance, Andersson and Vorobyov (2006) provided a fast *randomized* algorithm for solving MPGs in *sub-exponential* time $O\big(|V|^2|E|\exp\big(2\sqrt{|V|\ln(|E|/\sqrt{|V|})}+O(\sqrt{|V|}+\ln|E|)\big)\big)$. Lifshits and Pavlov (2007) devised a deterministic $O(2^{|V|}|V||E|\log W)$ *singly-exponential* time procedure by considering a so called potential-theory of MPGs, one that is akin to EGs.

Table 1 offers a summary of past and present results in chronological order.

- *An Energy-Lattice Decomposition of the Space of all Optimal Positional Strategies.*

Let us denote by $\texttt{opt}_\Gamma\Sigma^M_0$ the space of all the optimal positional strategies in a given MPG Γ. What allows Algorithm 1 to compute at least one $\sigma^*_0\in\texttt{opt}_\Gamma\Sigma^M_0$ is a so called *compatibility* relation, linking optimal arcs in MPGs to arcs that are *compatible* w.r.t. least-SEPMs in reweighted EGs. The family $\mathscr{E}_\Gamma$ of all SEPMs of a given EG Γ forms a complete lattice, which we call the Energy-Lattice of the EG Γ. Firstly, we observe that even though compatibility w.r.t. *least*-SEPMs in reweighted EGs implies optimality of positional strategies in MPGs (see Theorem 4), the converse doesn't hold generally (see Proposition 13). Thus a natural question was whether compatibility w.r.t. SEPMs was really appropriate to capture (e.g., to provide a recursive enumeration of) the whole $\texttt{opt}_\Gamma\Sigma^M_0$ and not just a proper subset of it. Partially motivated by this question we further explored on the relationship between $\texttt{opt}_\Gamma\Sigma^M_0$ and $\mathscr{E}_\Gamma$. In Theorem 7, it is observed a unique complete decomposition of $\texttt{opt}_\Gamma\Sigma^M_0$ which is expressed in terms of so called

Table 1: Time Complexity of the main Algorithms for solving MPGs.

Algorithm	Optimal Strategy Synthesis	Value Problem
This work	$O(\|E\|\log\|V\|)+\Theta\big(\sum_{v\in V}\texttt{deg}_{\Gamma}(v)\cdot\ell^1_{\Gamma}(v)\big)$	*same complexity*
CR15-16	$\Theta\big(\|V\|^2\|E\|W+\sum_{v\in V}\texttt{deg}_{\Gamma}(v)\cdot\ell^0_{\Gamma}(v)\big)$	*same complexity*
BC11	$O(\|V\|^2\|E\|W\log(\|V\|W))$	*same complexity*
LP07	n/a	$O(\|V\|\|E\|2^{\|V\|}\log W)$
AV06	$O\Big(\|V\|^2\|E\|e^{2\sqrt{\|V\|\ln\left(\frac{\|E\|}{\sqrt{\|V\|}}\right)}+O(\sqrt{\|V\|}+\ln\|E\|)}\Big)$	*same complexity*
ZP96	$\Theta(\|V\|^4\|E\|W\log\frac{\|E\|}{\|V\|})$	$\Theta(\|V\|^3\|E\|W)$

extremal-SEPMs in reweighted EGs. This points out what we called the "Energy-Lattice $\mathscr{X}^*_{\Gamma}$ associated to $\texttt{opt}_{\Gamma}\Sigma^M_0$", i.e., the family of all the extremal-SEPMs of a given Γ. So, compatibility w.r.t. SEPMs actually turns out to be appropriate for constructing the whole $\texttt{opt}_{\Gamma}\Sigma^M_0$; but an entire lattice $\mathscr{X}^*_{\Gamma}$ of extremal-SEPMs then arises (and not only the least-SEPM, which accounts only for the "join/top" component of $\texttt{opt}_{\Gamma}\Sigma^M_0$).

- *A Recursive Enumeration of Extremal-SEPMs and of Optimal Positional Strategies.*

Finally, it is offered a pseudo-polynomial total time recursive procedure for enumerating (w/o repetitions) all the elements of $\mathscr{X}^*_{\Gamma}$, and for computing the associated partitioning of $\texttt{opt}_{\Gamma}\Sigma^M_0$. This shows that the above mentioned compatibility relation is appropriate so to extend our algorithms, recursively, in order to compute the whole $\texttt{opt}_{\Gamma}\Sigma^M_0$ and $\mathscr{X}^*_{\Gamma}$. It is observed that the corresponding recursion tree actually defines an additional lattice $\mathscr{B}^*_{\Gamma}$, whose elements are certain sub-games $\Gamma'\subseteq\Gamma$ that we call *basic*. The extremal-SEPMs of a given Γ coincide with the least-SEPMs of the basic sub-games of Γ; so, $\mathscr{X}^*_{\Gamma}$ is the energy-lattice comprising all and only the *least*-SEPMs of the *basic* sub-games of Γ. The total time complexity of the proposed enumeration for both $\mathscr{B}^*_{\Gamma}$ and $\mathscr{X}^*_{\Gamma}$ is $O(|V|^3|E|W|\mathscr{B}^*_{\Gamma}|)$, it works in space $O(|V||E|)+\Theta\big(|E||\mathscr{B}^*_{\Gamma}|\big)$.

Organization. The manuscript is organized as follows. In Section 2, we introduce some notation and provide the required background on infinite two-player pebble-games and related algorithmic results. In Section 3, a suitable relation between values, optimal strategies, and certain reweighting operations is recalled from Comin and Rizzi (2015, 2016). In Section 4, it is offered an $O(|E|\log|V|)+\Theta\big(\sum_{v\in V}\texttt{deg}_{\Gamma}(v)\cdot\ell^1_{\Gamma}(v)\big)=O(|V|^2|E|W)$ pseudo-polynomial time and $\Theta(|V|+|E|)$ space deterministic algorithm for solving the Value Problem and Optimal Strategy Synthesis in MPGs; SubSection 4.4 offers an experimental comparison between Algorithm 1 and Algorithm 0 (Comin and Rizzi, 2015, 2016). Section 5 offers a unique and complete energy-lattice decomposition of $\texttt{opt}_{\Gamma}\Sigma^M_0$. Finally, Section 6 provides a recursive enumeration of $\mathscr{X}^*_{\Gamma}$ and the corresponding partitioning of $\texttt{opt}_{\Gamma}\Sigma^M_0$. The manuscript concludes in Section 7.

2. Notation and Preliminaries

We denote by $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ the set of natural, integer, and rational numbers. It will be sufficient to consider integral intervals, e.g., $[a,b]\triangleq\{z\in\mathbb{Z}\mid a\leq z\leq b\}$ and $[a,b)\triangleq\{z\in\mathbb{Z}\mid a\leq z<b\}$ for

any $a,b \in \mathbb{Z}$. If $(a,b),(a',b') \in \mathbb{Z}$, then $(a,b) < (a',b')$ *iff* $a < a'$, or $a = a'$ and $b < b'$. Our graphs are directed and weighted on the arcs; thus, if $G = (V,E,w)$ is a graph, then every arc $e \in E$ is a triplet $e = (u,v,w_e)$, where $w_e = w(u,v) \in \mathbb{Z}$. Let $W \triangleq \max_{e\in E} |w_e|$ be the maximum absolute weight. Given a vertex $u \in V$, the set of its successors is $N^{\text{out}}_\Gamma(u) \triangleq \{v \in V \mid (u,v) \in E\}$, whereas the set of its predecessors is $N^{\text{in}}_\Gamma(u) \triangleq \{v \in V \mid (v,u) \in E\}$. Let $\mathtt{deg}_\Gamma(v) \triangleq |N^{\text{in}}_\Gamma(v)| + |N^{\text{out}}_\Gamma(v)|$. A *path* is a sequence $v_0v_1 \dots v_n \dots$ such that $\forall^{i\in[n]}\,(v_i,v_{i+1}) \in E$. Let V^* be the set of all (possibly empty) finite paths. A *simple path* is a finite path $v_0v_1 \dots v_n$ having no repetitions, i.e., for any $i,j \in [0,n]$ it holds $v_i \neq v_j$ if $i \neq j$. A *cycle* is a path $v_0v_1 \dots v_{n-1}v_n$ such that $v_0 \dots v_{n-1}$ is simple and $v_n = v_0$. The *average weight* of a cycle $v_0 \dots v_n$ is $w(C)/|C| = \frac{1}{n}\sum_{i=0}^{n-1} w(v_i,v_{i+1})$. A cycle $C = v_0v_1 \dots v_n$ is *reachable* from v in G if there is some path p in G such that $p \cap C \neq \emptyset$.

An *arena* is a tuple $\Gamma = (V,E,w,\langle V_0,V_1\rangle)$ where $G^\Gamma \triangleq (V,E,w)$ is a finite weighted directed graph and (V_0,V_1) is a partition of V into the set V_0 of vertices owned by Player 0, and V_1 owned by Player 1. It is assumed that G^Γ has no sink, i.e., $\forall^{v\in V} N^{\text{out}}_\Gamma(v) \neq \emptyset$; we remark that G^Γ is not required to be a bipartite graph on colour classes V_0 and V_1. A sub-arena Γ' (or *sub-game*) of Γ is any arena $\Gamma' = (V',E',w',\langle V'_0,V'_1\rangle)$ such that: $V' \subseteq V$, $\forall^{i\in\{0,1\}} V'_i = V' \cap V_i$, $E' \subseteq E$, and $\forall^{e\in E'} w'_e = w_e$. Given $S \subseteq V$, the sub-arena of Γ induced by S is denoted $\Gamma_{|S}$, its vertex set is S and its edge set is $E' = \{(u,v) \in E \mid u,v \in S\}$. A game on Γ is played for infinitely many rounds by two players moving a pebble along the arcs of G^Γ. At the beginning of the game the pebble is found on some vertex $v_s \in V$, which is called the *starting position* of the game. At each turn, assuming the pebble is currently on a vertex $v \in V_i$ (for $i = 0,1$), Player i chooses an arc $(v,v') \in E$ and then the next turn starts with the pebble on v'. Below, Fig. 1 depicts an example arena Γ_{ex}.

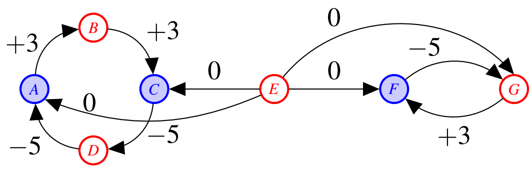


Figure 1: An arena $\Gamma_{\text{ex}} = \langle V,\mathsf{E},w,(V_0,V_1)\rangle$. Here, $V = \{A,B,C,D,E,F,G\}$ and $\mathsf{E} = \{(A,B,+3),(B,C,+3),(C,D,-5),(D,A,-5),(E,A,0),(E,C,0),(E,F,0),(E,G,0),(F,G,-5),(G,F,+3)\}$. Also, $V_0 = \{B,D,E,G\}$ is colored in red, while $V_1 = \{A,C,F\}$ is filled in blue.

A *play* is any infinite path $v_0v_1 \dots v_n \dots \in V^\omega$ in Γ. For any $i \in \{0,1\}$, a strategy of Player i is any function $\sigma_i : V^* \times V_i \to V$ such that for every finite path $p'v$ in G^Γ, where $p' \in V^*$ and $v \in V_i$, it holds that $(v,\sigma_i(p',v)) \in E$. A strategy σ_i of Player i is *positional* (or *memoryless*) if $\sigma_i(p,v_n) = \sigma_i(p',v'_m)$ for every finite paths $pv_n = v_0 \dots v_{n-1}v_n$ and $p'v'_m = v'_0 \dots v'_{m-1}v'_m$ in G^Γ such that $v_n = v'_m \in V_i$. The set of all the positional strategies of Player i is denoted by Σ^M_i. A play $v_0v_1 \dots v_n \dots$ is *consistent* with a strategy $\sigma \in \Sigma_i$ if $v_{j+1} = \sigma(v_0v_1 \dots v_j)$ whenever $v_j \in V_i$.

Given a starting position $v_s \in V$, the *outcome* of two strategies $\sigma_0 \in \Sigma_0$ and $\sigma_1 \in \Sigma_1$, denoted $\mathtt{outcome}^\Gamma(v_s,\sigma_0,\sigma_1)$, is the unique play that starts at v_s and is consistent with both σ_0 and σ_1.

Given a memoryless strategy $\sigma_i \in \Sigma^M_i$ of Player i in Γ, then $G(\sigma_i,\Gamma) = (V,E_{\sigma_i},w)$ is the graph obtained from G^Γ by removing all the arcs $(v,v') \in E$ such that $v \in V_i$ and $v' \neq \sigma_i(v)$; we say that $G(\sigma_i,\Gamma)$ is obtained from G^Γ *by projection* w.r.t. σ_i.

For any weight function $w' : E \to \mathbb{Z}$, the *reweighting* of $\Gamma = (V,E,w,\langle V_0,V_1\rangle)$ w.r.t. w' is the arena $\Gamma^{w'} = (V,E,w',\langle V_0,V_1\rangle)$. Also, for $w : E \to \mathbb{Z}$ and any $\nu \in \mathbb{Z}$, we denote by $w+\nu$ the weight function w' defined as $\forall^{e\in E} w'_e \triangleq w_e + \nu$. Indeed, we shall consider reweighted games of the form Γ^{w-q}, for some $q \in \mathbb{Q}$. Notice that the corresponding weight function $w' : E \to$

$\mathbb{Q} : e \mapsto w_e - q$ is rational, while we required the weights of the arcs to be always integers. To overcome this issue, it is sufficient to re-define Γ^{w-q} by scaling all weights by a factor equal to the denominator of $q \in \mathbb{Q}$; i.e., when $q \in \mathbb{Q}$, say $q = N/D$ for $\gcd(N,D) = 1$ we define $\Gamma^{w-q} \triangleq \Gamma^{D\cdot w-N}$. This rescaling operation doesn't change the winning regions of the corresponding games (we denote this equivalence as $\Gamma^{w-q} \cong \Gamma^{D\cdot w-N}$), and it has the significant advantage of allowing for a discussion (and an algorithmics) strictly based on integer weights.

2.1. Mean Payoff Games

A *Mean Payoff Game* (MPG) (Brim et al., 2011; Zwick and Paterson, 1996; Ehrenfeucht and Mycielski, 1979) is a game played on some arena Γ for infinitely many rounds by two opponents, Player 0 gains a payoff defined as the long-run average weight of the play, whereas Player 1 loses that value. Formally, the Player 0's *payoff* of a play $v_0 v_1 \ldots v_n \ldots$ in Γ is defined as follows:

$$\texttt{MP}_0(v_0 v_1 \ldots v_n \ldots) \triangleq \liminf_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} w(v_i, v_{i+1}).$$

The value *secured* by a strategy $\sigma_0 \in \Sigma_0$ in a vertex v is defined as:

$$\texttt{val}^{\sigma_0}(v) \triangleq \inf_{\sigma_1 \in \Sigma_1} \texttt{MP}_0\big(\texttt{outcome}^{\Gamma}(v, \sigma_0, \sigma_1)\big),$$

Notice that payoffs and secured values can be defined symmetrically for the Player 1 (i.e., by interchanging the symbol *0* with *1* and *inf* with *sup*).

Ehrenfeucht and Mycielski Ehrenfeucht and Mycielski (1979) proved that each vertex $v \in V$ admits a unique *value*, denoted $\texttt{val}^{\Gamma}(v)$, which each player can secure by means of a *memoryless* (or *positional*) strategy. Moreover, *uniform* positional optimal strategies do exist for both players, in the sense that for each player there exist at least one positional strategy which can be used to secure all the optimal values, independently with respect to the starting position v_s. Thus, for every MPG Γ, there exists a strategy $\sigma_0 \in \Sigma_0^M$ such that $\forall^{v\in V}\texttt{val}^{\sigma_0}(v) \geq \texttt{val}^{\Gamma}(v)$, and there exists a strategy $\sigma_1 \in \Sigma_1^M$ such that $\forall^{v\in V}\texttt{val}^{\sigma_1}(v) \leq \texttt{val}^{\Gamma}(v)$. The *(optimal) value* of a vertex $v \in V$ in the MPG Γ is given by:

$$\texttt{val}^{\Gamma}(v) = \sup_{\sigma_0 \in \Sigma_0} \texttt{val}^{\sigma_0}(v) = \inf_{\sigma_1 \in \Sigma_1} \texttt{val}^{\sigma_1}(v).$$

Thus, a strategy $\sigma_0 \in \Sigma_0$ is *optimal* if $\texttt{val}^{\sigma_0}(v) = \texttt{val}^{\Gamma}(v)$ for all $v \in V$. We denote $\text{opt}_{\Gamma}\Sigma_0^M \triangleq \big\{\sigma_0 \in \Sigma_0^M(\Gamma) \mid \forall^{v\in V}\,\texttt{val}^{\Gamma}_{\sigma_0}(v) = \texttt{val}^{\Gamma}(v)\big\}$. A strategy $\sigma_0 \in \Sigma_0$ is said to be *winning* for Player 0 if $\forall^{v\in V}\texttt{val}^{\sigma_0}(v) \geq 0$, and $\sigma_1 \in \Sigma_1$ is winning for Player 1 if $\texttt{val}^{\sigma_1}(v) < 0$. Correspondingly, a vertex $v \in V$ is a *winning starting position* for Player 0 if $\texttt{val}^{\Gamma}(v) \geq 0$, otherwise it is winning for Player 1. The set of all winning starting positions of Player i is denoted by $\mathscr{W}_i$ for $i \in \{0,1\}$.

A refined formulation of the determinacy theorem is offered in Björklund et al. (2004).

Theorem 1 (Björklund et al. (2004))**.** *Let Γ be an MPG and let $\{C_i\}_{i=1}^m$ be a partition (called* ergodic*) of its vertices into $m \geq 1$ classes each one having the same optimal value $\nu_i \in \mathbb{Q}$. Formally, $V = \bigsqcup_{i=1}^m C_i$ and $\forall^{i\in[m]}\forall^{v\in C_i}\,\mathit{val}^{\Gamma_i}(v) = \nu_i$, where $\Gamma_i \triangleq \Gamma_{|C_i}$.*

Then, Player 0 has no vertices with outgoing arcs leading from C_i to C_j whenever $\nu_i < \nu_j$, and Player 1 has no vertices with outgoing arcs leading from C_i to C_j whenever $\nu_i > \nu_j$;

moreover, there exist $\sigma_0 \in \Sigma_0^M$ and $\sigma_1 \in \Sigma_1^M$ such that:

– If the game starts from any vertex in C_i, then σ_0 secures a gain at least ν_i to Player 0 and σ_1 secures a loss at most ν_i to Player 1;

– Any play that starts from C_i always stays in C_i, if it is consistent with both strategies σ_0, σ_1, i.e., if Player 0 plays according to σ_0, and Player 1 according to σ_1.

A finite variant of MPGs is well-known in the literature (Ehrenfeucht and Mycielski, 1979; Zwick and Paterson, 1996; Brim et al., 2011), where the game stops as soon as a cyclic sequence of vertices is traversed. It turns out that this is equivalent to the infinite game formulation (Ehrenfeucht and Mycielski, 1979), in the sense that the values of an MPG are in a strong relationship with the average weights of its cycles, as in the next lemma.

Proposition 1 (Brim, *et al.* Brim et al. (2011))**.** *Let Γ be an MPG. For all $\nu \in \mathbb{Q}$, for all $\sigma_0 \in \Sigma_0^M$, and for all $v \in V$, the value $\mathtt{val}^{\sigma_0}(v)$ is greater than ν iff all cycles C reachable from v in the projection graph $G^{\Gamma}_{\sigma_0}$ have an average weight $w(C)/|C|$ greater than ν.*

The proof of Proposition 1 follows from the memoryless determinacy of MPGs. We remark that a proposition which is symmetric to Proposition 1 holds for Player 1 as well: for all $\nu \in \mathbb{Q}$, for all positional strategies $\sigma_1 \in \Sigma_1^M$ of Player 1, and for all vertices $v \in V$, the value $\mathtt{val}^{\sigma_1}(v)$ is less than ν *iff* if all cycles reachable from v in the projection graph $G^{\Gamma}_{\sigma_1}$ have an average weight less than ν. Also, it is well-known (Brim et al., 2011; Ehrenfeucht and Mycielski, 1979) that each value $\mathtt{val}^{\Gamma}(v)$ is contained within the following set of rational numbers:

$$S_\Gamma = \Big\{ N/D \mid D \in [1, |V|], N \in [-D \cdot W, D \cdot W] \Big\}.$$

Notice, $|S_\Gamma| \leq |V|^2 W$.

The present work tackles on the algorithmics of the following two classical problems:

– *Value Problem.* Compute for each vertex $v \in V$ the (rational) optimal value $\mathtt{val}^{\Gamma}(v)$.

– *Optimal Strategy Synthesis.* Compute an optimal positional strategy for Player 0 in Γ.

In Section 6 we shall consider the problem of computing the whole $\mathtt{opt}_\Gamma \Sigma_0^M$.

– *Optimal Strategy Enumeration.* Provide a listing[1] of all the optimal positional strategies of Player 0 in the MPG Γ.

2.2. Energy Games and Small Energy-Progress Measures

An *Energy Game* (EG) is a game that is played on an arena Γ for infinitely many rounds by two opponents, where the goal of Player 0 is to construct an infinite play $v_0 v_1 \ldots v_n \ldots$ such that for some initial *credit* $c \in \mathbb{N}$ the following holds: $c + \sum_{i=0}^{j} w(v_i, v_{i+1}) \geq 0$, for all $j \geq 0$. Given a credit $c \in \mathbb{N}$, a play $v_0 v_1 \ldots v_n \ldots$ is *winning* for Player 0 if it satisfies (1), otherwise it is winning for Player 1. A vertex $v \in V$ is a winning starting position for Player 0 if there exists an initial credit $c \in \mathbb{N}$ and a strategy $\sigma_0 \in \Sigma_0$ such that, for every strategy $\sigma_1 \in \Sigma_1$, the play $\mathtt{outcome}^{\Gamma}(v, \sigma_0, \sigma_1)$ is winning for Player 0. As in the case of MPGs, the EGs are memoryless determined Brim et al. (2011), i.e., for every $v \in V$, either v is winning for Player 0 or v is winning for Player 1, and (uniform) memoryless strategies are sufficient to win the game. In fact, as shown in the next lemma, the decision problems of MPGs and EGs are intimately related.

[1]The listing have to be exhaustive (i.e., each element is listed, eventually) and without repetitions (i.e., no element is listed twice).

Proposition 2 (Brim et al. (2011))**.** *Let Γ be an arena. For all threshold $\nu \in \mathbb{Q}$, for all vertices $v \in V$, Player 0 has a strategy in the MPG Γ that secures value at least ν from v if and only if, for some initial credit $c \in \mathbb{N}$, Player 0 has a winning strategy from v in the reweighted EG $\Gamma^{w-\nu}$.*

In this work we are especially interested in the *Minimum Credit Problem* (MCP) for EGs: for each winning starting position v, compute the minimum initial credit $c^* = c^*(v)$ such that there exists a winning strategy $\sigma_0 \in \Sigma_0^M$ for Player 0 starting from v. A fast pseudo-polynomial time deterministic procedure for solving MCPs comes from Brim et al. (2011).

Theorem 2 (Brim et al. (2011))**.** *There exists a deterministic algorithm for solving the MCP within $O(|V||E|W)$ pseudo-polynomial time, on any input EG $(V, E, w, \langle V_0, V_1\rangle)$.*

The algorithm mentioned in Theorem 2 is the *Value-Iteration* algorithm (Brim et al., 2011). Its rationale relies on the notion of *Small Energy-Progress Measures* (SEPMs).

2.3. Energy-Lattices of Small Energy-Progress Measures

Small-Energy Progress Measures are bounded, non-negative and integer-valued functions that impose local conditions to ensure global properties on the arena, in particular, witnessing that Player 0 has a way to enforce conservativity (i.e., non-negativity of cycles) in the resulting game's graph. Recovering standard notation, see e.g. Brim et al. (2011), let us denote $\mathscr{C}_\Gamma = \{n \in \mathbb{N} \mid n \leq (|V|-1)W\} \cup \{\top\}$ and let $\preceq$ be the total order on $\mathscr{C}_\Gamma$ defined as: $x \preceq y$ *iff* either $y = \top$ or $x, y \in \mathbb{N}$ and $x \leq y$. In order to cast the minus operation to range over $\mathscr{C}_\Gamma$, let us consider an operator $\ominus : \mathscr{C}_\Gamma \times \mathbb{Z} \to \mathscr{C}_\Gamma$ defined as follows:

$$a \ominus b \triangleq \begin{cases} \max(0, a-b) & \text{, if } a \neq \top \text{ and } a-b \leq (|V|-1)W; \\ a \ominus b = \top & \text{, otherwise.} \end{cases}$$

Given an EG Γ on vertex set $V = V_0 \cup V_1$, a function $f : V \to \mathscr{C}_\Gamma$ is a *Small Energy-Progress Measure* (SEPM) for Γ *iff* the following two conditions are met:

1. if $v \in V_0$, then $f(v) \succeq f(v') \ominus w(v, v')$ for *some* $(v, v') \in E$;
2. if $v \in V_1$, then $f(v) \succeq f(v') \ominus w(v, v')$ for *all* $(v, v') \in E$.

The values of a SEPM, i.e., the elements of the image $f(V)$, are called the *energy levels* of f. It is worth to denote by $V_f = \{v \in V \mid f(v) \neq \top\}$ the set of vertices having finite energy. Given a SEPM $f : V \to \mathscr{C}_\Gamma$ and a vertex $v \in V_0$, an arc $(v, v') \in E$ is said to be *compatible* with f whenever $f(v) \succeq f(v') \ominus w(v, v')$; otherwise (v, v') is said to be *incompatible* with f. Moreover, a positional strategy $\sigma_0 \in \Sigma_0^M$ is said to be *compatible* with f whenever: $\forall^{v \in V_0}$ if $\sigma_0(v) = v'$ then $(v, v') \in E$ is compatible with f; otherwise, σ_0 is *incompatible* with f.

It is well-known that the family of all the SEPMs of a given Γ forms a complete (finite) lattice, which we denote by $\mathscr{E}_\Gamma$ call it the *Energy-Lattice* of Γ. Therefore, we shall consider:

$$\mathscr{E}_\Gamma \triangleq \big(\{f : V \to \mathscr{C}_\Gamma \mid f \text{ is SEPM of } \Gamma\}, \sqsubseteq\big),$$

where for any two SEPMs f, g define $f \sqsubseteq g$ *iff* $\forall v \in V\, f(v) \preceq g(v)$. Notice that, whenever f and g are SEPMs, then so is the *minimum function* defined as: $\forall^{v \in V} h(v) \triangleq \min\{f(v), g(v)\}$. This fact allows one to consider the *least* SEPM, namely, the unique SEPM $f^* : V \to \mathscr{C}_\Gamma$ such that, for any other SEPM $g : V \to \mathscr{C}_\Gamma$, the following holds: $\forall^{v \in V} f^*(v) \preceq g(v)$. Thus, $\mathscr{E}_\Gamma$ is a complete lattice. So, $\mathscr{E}_\Gamma$ enjoys of *Knaster–Tarski Theorem*, which states that the set of fixed-points of a monotone function on a complete lattice is again a complete lattice.

Also concerning SEPMs, we shall rely on the following lemmata. The first one relates SEPMs to the winning region $\mathscr{W}_0$ of Player 0 in EGs.

Proposition 3 (Brim et al. (2011)). *Let Γ be an EG. Then the following hold.*

1. *If f is any SEPM of the EG Γ and $v \in V_f$, then v is a winning starting position for Player 0 in the EG Γ. Stated otherwise, $V_f \subseteq \mathscr{W}_0$;*
2. *If f^* is the least SEPM of the EG Γ, and v is a winning starting position for Player 0 in the EG Γ, then $v \in V_{f^*}$. Thus, $V_{f^*} = \mathscr{W}_0$.*

The following bound holds on the energy-levels of any SEPM (by definition of $\mathscr{C}_\Gamma$).

Proposition 4. *Let Γ be an EG. Let f be any SEPM of Γ.*
Then, for every $v \in V$ either $f(v) = \top$ or $0 \le f(v) \le (|V|-1)W$.

2.4. The Value-Iteration Algorithm for solving MCPs in EGs

In order to resolve MCPs in EGs, the well-known *Value-Iteration* (Brim et al., 2011) algorithm is employed. Given an EG Γ as input, the Value-Iteration aims to compute the least SEPM f^* of Γ. This simple procedure basically relies on an *energy-lifting operator* δ. Given $v \in V$, the energy-lifting operator $\delta(\cdot, v) : [V \to \mathscr{C}_\Gamma] \to [V \to \mathscr{C}_\Gamma]$ is defined by $\delta(f, v) \triangleq g$, where:

$$g(u) \triangleq \begin{cases} f(u) & \text{if } u \neq v \\ \min\{f(v') \ominus w(v,v') \mid v' \in N^{\text{out}}_\Gamma(v)\} & \text{if } u = v \in V_0 \\ \max\{f(v') \ominus w(v,v') \mid v' \in N^{\text{out}}_\Gamma(v)\} & \text{if } u = v \in V_1 \end{cases}$$

We also need the following definition. Given a function $f : V \to \mathscr{C}_\Gamma$, we say that f is *inconsistent* in v whenever one of the following two holds:
1. $v \in V_0$ and $\forall^{v' \in N^{\text{out}}_\Gamma(v)} f(v) \prec f(v') \ominus w(v,v')$;
2. $v \in V_1$ and $\exists^{v' \in N^{\text{out}}_\Gamma(v)} f(v) \prec f(v') \ominus w(v,v')$.
In that case, we also say that v is inconsistent w.r.t. f in $\Gamma_{i,j}$.

To start with, the Value-Iteration algorithm initializes f to the constant zero function, i.e., $\forall^{v \in V} f(v) = 0$. Furthermore, the procedure maintains a list L^{inc} of vertices in order to witness the inconsistencies of f. Initially, $v \in V_0 \cap L^{\text{inc}}$ *iff* all arcs going out of v are negative, while $v \in V_1 \cap L^{\text{inc}}$ if and only if v is the source of at least one negative arc. Notice that checking the above conditions takes time $O(|E|)$.

While L^{inc} is nonempty, the algorithm picks a vertex v from L^{inc} and performs the following:

1. Apply the lifting operator $\delta(f, v)$ to f in order to resolve the inconsistency of f in v;
2. Insert into L^{inc} all vertices $u \in N^{\text{in}}_\Gamma(v) \setminus L^{\text{inc}}$ witnessing a new inconsistency due to the increase of $f(v)$. (Here, the same vertex can't occur twice in L^{inc}.)

The algorithm terminates when L^{inc} is empty. This concludes the description of the Value-Iteration algorithm. As shown in Brim et al. (2011), the update of L^{inc} following an application of the lifting operator $\delta(f, v)$ requires $O(|N^{\text{in}}_\Gamma(v)|)$ time. Moreover, a single application of the lifting operator $\delta(\cdot, v)$ takes $O(|N^{\text{out}}_\Gamma(v)|)$ time at most. This implies that the algorithm can be implemented so that it will always halt within $O(|V||E|W)$ time (the reader is referred to Brim et al. (2011) for all the details of the correctness and complexity proofs).

Remark 1. *The Value-Iteration procedure lends itself to the following basic generalization, which is of a pivotal importance for us. Let f^* be the least SEPM of the EG Γ. Recall that, as a first step, the Value-Iteration algorithm initializes f to be the constant zero function. Here, we remark that it is not necessary to do that really. Indeed, it is sufficient to initialize f to be any* *function f_0 which bounds f^* from below, that is to say, to initialize f to any $f_0 : V \to \mathscr{C}_\Gamma$ such that $\forall^{v\in V}\ f_0(v) \preceq f^*(v)$. Soon after, L can be initialized in a natural way: just insert v into L^{inc} iff f_0 is inconsistent at v. This initialization still requires $O(|E|)$ time and it doesn't affect the correctness of the procedure.*

3. Values and Optimal Strategies from Reweightings

Values and Farey sequences. Recall that each value $\texttt{val}^\Gamma(v)$ is contained within the following set of rational numbers: $S_\Gamma = \{N/D \mid D \in [1,|V|], N \in [-D\cdot W, D\cdot W]\}$. Let us introduce some notation in order to handle S_Γ in a way that is suitable for our purposes. Firstly, we write every $\nu \in S_\Gamma$ as $\nu = i + F$, where $i = i_\nu = \lfloor \nu \rfloor$ is the integral and $F = F_\nu = \{\nu\} = \nu - i$ is the fractional part. Notice that $i \in [-W,W]$ and that $F \in \mathbb{Q}$ is non-negative with denominator at most $|V|$.

Therefore, it is worthwhile to consider the *Farey sequence* $\mathscr{F}_n$ of order $n = |V|$. This is the increasing sequence of all irreducible fractions from the (rational) interval $[0,1]$ with denominators less than or equal to n. In the rest of this paper, $\mathscr{F}_n$ denotes the following sorted set:

$$\mathscr{F}_n = \{N/D \mid 0 \le N \le D \le n, \gcd(N,D) = 1\}.$$

Farey sequences have numerous and interesting properties, in particular, many algorithms for generating the entire sequence $\mathscr{F}_n$ in time $O(n^2)$ are known in the literature Graham et al. (1994). The above mentioned quadratic running time is optimal, since $\mathscr{F}_n$ has $s(n) = \Theta(n^2)$ many elements. We shall assume that $F_0,\ldots,F_{s-1}$ is an increasing ordering of $\mathscr{F}_n$, so that $\mathscr{F}_n = \{F_j\}_{j=0}^{s-1}$ and $F_j < F_{j+1}$ for every j. Notice that $F_0 = 0$ and $F_{s-1} = 1$.

For example, $\mathscr{F}_5 = \{0, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, 1\}$.

We will be interested in generating the sequence $F_0,\ldots,F_{s-1}$, one term after another, iteratively and efficiently. As mentioned in Pawlewicz and Pătraşcu (2009), combining several properties satisfied by the Farey sequence, one can get a trivial iterative algorithm, which generates the next term $F_j = N_j/D_j$ of $\mathscr{F}_n$ based on the previous two:

$$N_j \leftarrow \left\lfloor \frac{D_{j-2}+n}{D_{j-1}} \right\rfloor \cdot N_{j-1} - N_{j-1}; \qquad D_j \leftarrow \left\lfloor \frac{D_{j-2}+n}{D_{j-1}} \right\rfloor \cdot D_{j-1} - D_{j-1}.$$

Given F_{j-2}, F_{j-1}, this computes F_j in $O(1)$ time and space. It will perfectly fit our needs.

At this point, it is worth observing that S_Γ can be represented as follows, this will be convenient in a while:

$$S_\Gamma = [-W,W) + \mathscr{F}_{|V|} = \left\{ i + F_j \;\middle|\; i \in [-W,W),\ j \in [0,s-1] \right\}.$$

A Characterization of Values in MPGs by reweighted EGs. It is now recalled a suitable characterization of optimal values in MPGs in terms of winning regions.

Theorem 3 (Comin and Rizzi (2016)). *Given an MPG $\Gamma = (V,E,w,\langle V_0,V_1\rangle)$, let us consider the reweightings:*

$$\Gamma_{i,j} \cong \Gamma^{w-i-F_j},\ \textit{for any } i \in [-W,W] \textit{ and } j \in [1,s-1],$$

where $s = |\mathscr{F}_{|V|}|$ and F_j is the j-th term of the Farey sequence $\mathscr{F}_{|V|}$.

Then, the following holds:

$$\mathtt{val}^\Gamma(v) = i + F_{j-1} \textit{ iff } v \in \mathscr{W}_0(\Gamma_{i,j-1}) \cap \mathscr{W}_1(\Gamma_{i,j}).$$

Proof. See the proof of [Theorem 3 in Comin and Rizzi (2016)]. □

3.1. A Description of Optimal Positional Strategies in MPGs from reweighted EGs

We provide a sufficient condition, for a positional strategy to be optimal, which is expressed in terms of reweighted EGs and their SEPMs.

Theorem 4 (Comin and Rizzi (2016))**.** *Let $\Gamma = (V, E, w, \langle V_0, V_1\rangle)$ be an MPG. For each $u \in V$, consider the reweighted EG $\Gamma_u \cong \Gamma^{w-\mathtt{val}^{\Gamma}(u)}$. Let $f_u : V \to \mathscr{C}_{\Gamma_u}$ be any SEPM of Γ_u such that $u \in V_{f_u}$ (i.e., $f_u(u) \neq \top$). Moreover, we assume: $f_{u_1} = f_{u_2}$ whenever $\mathtt{val}^{\Gamma}(u_1) = \mathtt{val}^{\Gamma}(u_2)$.*

When $u \in V_0$, let $v_{f_u} \in N^{out}_{\Gamma}(u)$ be any vertex such that $(u, v_{f_u}) \in E$ is compatible with f_u in EG Γ_u, and consider the positional strategy $\sigma_0^* \in \Sigma_0^M$ defined as follows: $\forall^{u\in V_0}\, \sigma_0^*(u) \triangleq v_{f_u}$.

Then, σ_0^* is an optimal positional strategy for Player 0 in the MPG Γ.

Proof. See the proof of [Theorem 4 in Comin and Rizzi (2016)]. □

Remark 2. *Notice that Theorem 4 holds, in particular, when f_u is the least SEPM f_u^* of the reweighted EG Γ_u. This is because $u \in V_{f_u^*}$ always holds for the least SEPM f_u^* of the EG Γ_u: indeed, by Proposition 2 and by definition of Γ_u, then u is a winning starting position for Player 0 in the EG Γ_u (for some initial credit); thus, by Proposition 3, it follows that $u \in V_{f_u^*}$.*

4. A Faster $O(|V|^2|E|W)$-Time Algorithm for MPGs by Jumping through Reweighted EGs

This section offers an $O(|E|\log|V|) + \Theta\big(\sum_{v\in V} \mathtt{deg}_{\Gamma}(v) \cdot \ell^1_{\Gamma}(v)\big) = O(|V|^2|E|W)$ time algorithm for solving the Value Problem and Optimal Strategy Synthesis in MPGs $\Gamma = (V, E, w, \langle V_0, V_1\rangle)$, where $W \triangleq \max_{e\in E} |w_e|$; it works with $\Theta(|V| + |E|)$ space. Its name is Algorithm 1.

In order to describe it in a suitable way, let us firstly recall some notation. Given an MPG Γ, we shall consider the following reweightings:

$$\Gamma_{i,j} \cong \Gamma^{w-i-F_j}, \text{ for any } i \in [-W, W] \text{ and } j \in [1, s-1],$$

where $s \triangleq |\mathscr{F}_{|V|}|$, and F_j is the j-th term of $\mathscr{F}_{|V|}$.

Assuming $F_j = N_j/D_j$ for some (co-prime) $N_j, D_j \in \mathbb{N}$, we work with the following weights:

$$w_{i,j} \triangleq w - i - F_j = w - i - N_j/D_j; \qquad w'_{i,j} \triangleq D_j\, w_{i,j} = D_j\,(w-i) - N_j.$$

Recall $\Gamma_{i,j} \triangleq \Gamma^{w'_{i,j}}$ and $\forall^{e\in E} w'_{i,j}(e) \in \mathbb{Z}$. Notice, since $F_1 < \ldots < F_{s-1}$ is monotone increasing, $\{w_{i,j}\}_{i,j}$ can be ordered (inverse)-lexicographically w.r.t. (i, j); i.e., $w_{(i,j)} > w_{(i',j')}$ *iff* either: $i < i'$, or $i = i'$ and $j < j'$; e.g., $w_{W^-,1} > w_{W^-,2} > \ldots > w_{W^-,s-1} > \ldots > w_{W^+-1,s-1} > w_{W^+,s-1}$. Also, we denote the least-SEPM of the reweighted EG $\Gamma_{i,j}$ by $f^*_{w'_{i,j}} : V \to \mathscr{C}_{\Gamma_{i,j}}$. In addition, $f^*_{i,j} : V \to \mathbb{Q}$ denotes the *rational-scaling* of $f^*_{w'_{i,j}}$, which is defined as: $\forall^{v\in V} f^*_{i,j}(v) \triangleq \frac{1}{D_j} \cdot f^*_{w'_{i,j}}(v)$. Finally, if f is any SEPM of the EG $\Gamma_{i,j}$, then $\text{Inc}(f, i, j) \triangleq \{v \in V \mid v \text{ is inconsistent w.r.t. } f \text{ in } \Gamma_{i,j}\}$.

4.1. Description of Algorithm 1

Outline. Given an input arena $\Gamma = (V, E, w, \langle V_0, V_1\rangle)$, Algorithm 1 aims at returning a tuple $(\mathscr{W}_0, \mathscr{W}_1, \nu, \sigma_0^*)$ where: $\mathscr{W}_0$ is the winning set of Player 0 in the MPG Γ, and $\mathscr{W}_1$ is that of Player 1; $\nu : V \to S_{\Gamma}$ maps each starting position $v_s \in V$ to $\mathtt{val}^{\Gamma}(v_s)$; finally, $\sigma_0^* : V_0 \to V$ is an optimal positional strategy for Player 0 in the MPG Γ.

Let $W^- \triangleq \min_{e\in E} w_e$ and $W^+ \triangleq \max_{e\in E} w_e$. The first aspect underlying Algorithm 1 is that of ordering $[W^-, W^+] \times [1, s-1]$ lexicographically, by considering the above mentioned (decreasing) sequence of weights:

$$\rho : [W^-, W^+] \times [1, s-1] \to \mathbb{Z}^E : (i, j) \mapsto w_{i,j},$$

$$\rho : w_{W^-,1} > w_{W^-,2} > \ldots > w_{W^-,s-1} > w_{W^-+1,1} > w_{W^-+1,2} > \ldots > w_{W^+-1,s-1} > \ldots > w_{W^+,s-1},$$

then, to rely on Theorem 3, at each step of ρ, testing whether some *transition of winning regions* occurs. At the generic (i, j)-th step of ρ, we run a Value-Iteration (Brim et al., 2011) in order to compute the least-SEPM of $\Gamma_{i,j}$, and then we check for every $v \in V$ whether v is winning for Player 1 w.r.t. the *current* weight $w_{i,j}$ (i.e., w.r.t. $\Gamma_{i,j}$), meanwhile recalling whether v was winning for Player 0 w.r.t. the (immediately, inverse-lex) *previous* weight $w_{\texttt{prev}_\rho(i,j)}$ (i.e., w.r.t. $\Gamma_{\texttt{prev}_\rho(i,j)}$). This step relies on Proposition 3, as in fact $\mathscr{W}_0(\Gamma_{\texttt{prev}_\rho(i,j)}) = V_{f^*_{\texttt{prev}_\rho(i,j)}}$ and $\mathscr{W}_1(\Gamma_{i,j}) = V \setminus V_{f^*_{i,j}}$.

If a transition occurs, say for some $\hat{v} \in \mathscr{W}_0(\Gamma_{\texttt{prev}_\rho(i,j)}) \cap \mathscr{W}_1(\Gamma_{i,j})$, then $\texttt{val}^\Gamma(\hat{v})$ can be computed easily by relying on Theorem 3, i.e., $\nu(\hat{v}) \leftarrow i + F[j-1]$; also, an optimal positional strategy can be extracted from $f^*_{\texttt{prev}_\rho(i,j)}$ thanks to Theorem 4 and Remark 2, provided that $\hat{v} \in V_0$.

Each phase, in which one does a Value-Iteration and looks at transitions of winning regions, it is named *Scan-Phase*. Remarkably, for every $i \in [W^-, W^+]$ and $j \in [1, s-1]$, the (i, j)*-th* Scan-Phase performs a Value-Iteration (Brim et al., 2011) on the reweighted EG $\Gamma_{i,j}$ by initializing all the energy-levels to those computed by the previous Scan-Phase (subject to a suitable re-scaling and a rounding-up, i.e., $\lceil D_j \cdot f^*_{\texttt{prev}_\rho(i,j)} \rceil$). As described in Comin and Rizzi (2016), the main step of computation that is carried on at the (i, j)-th Scan-Phase goes therefore as follows:

$$f_{i,j} \leftarrow \frac{1}{D_j} \texttt{Value-Iteration}\Big(\Gamma_{i,j}, \lceil D_j \cdot f^*_{\texttt{prev}_\rho(i,j)} \rceil\Big),$$

where D_j is the denominator of F_j. Then, one can prove that $\forall (i, j)\, f_{i,j} = f^*_{i,j}$ [Comin and Rizzi (2016), Lemma 8, Item 4]. Indeed, by Propositions 2 and Proposition 3, $\mathscr{W}_0(\Gamma_{\texttt{prev}_\rho(i,j)}) = V_{f^*_{\texttt{prev}_\rho(i,j)}}$ and $\mathscr{W}_1(\Gamma_{i,j}) = V \setminus V_{f^*_{i,j}}$. And since ρ is monotone decreasing, the sequence of energy-levels $\psi_\rho : (i, j) \mapsto f^*_{i,j}$ is monotone non-decreasing [Comin and Rizzi (2016), Lemma 8, Item 1]:

$$\psi_\rho : f^*_{W^-,1} \preceq f^*_{W^-,2} \preceq \cdots \preceq f^*_{W^-,s-1} \preceq f^*_{W^-+1,1} \preceq f^*_{W^-+1,2} \preceq \cdots \preceq f^*_{W^+-1,s-1} \preceq \cdots \preceq f^*_{W^+,s-1};$$

Our algorithm will succeed at *amortizing* the cost of the corresponding sequence of Value-Iterations for computing ψ_ρ. A similar amortization takes place already in Algorithm 0.

However, Algorithm 0 performs exactly one Scan-Phase (i.e., one Value-Iteration, plus the tests $v \in^? \mathscr{W}_0(\Gamma_{\texttt{prev}_\rho(i,j)}) \cap \mathscr{W}_1(\Gamma_{i,j})$) for each term of ρ –without making any *Jump* in ρ–. So, Algorithm 0 performs $\Theta(|V|^2 W)$ Scan-Phases overall, each one costing $\Omega(|E|)$ time (i.e., the cost of initializing the Value-Iteration as in Brim et al. (2011)). This brings an overall $\Omega(|V|^2|E|W)$ time complexity, which turns out to be also $O(|V|^2|E|W)$; leading us to an improved pseudo-polynomial time upper bound for solving MPGs (Comin and Rizzi, 2015, 2016).

The present work shows that it is instead possible, and actually very convenient, to perform many *Jumps* in ρ; thus introducing "gaps" between the weights that are considered along the sequence of Scan-Phases. The corresponding sequence of weights is denoted by ρ^J. This is Algorithm 1. In Fig. 2, a graphical intuition of Algorithm 1 and ρ^J is given, in which a Jump is depicted with an arc going from $w_{W^-,2}$ to $w_{\texttt{prev}_{\rho^J}(i,j)}$, e.g., $w_{\texttt{prev}_{\rho^J}(\texttt{prev}_{\rho^J}(i,j))} = w_{W^-,2}$.

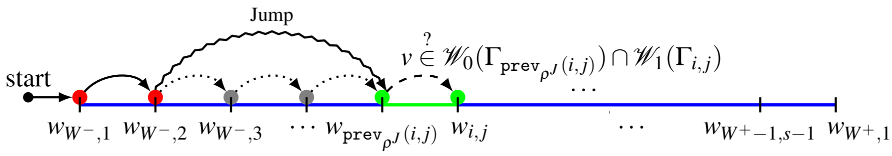


Figure 2: An illustration of Algorithm 1.

Two distinct kinds of Jumps are employed: *Energy-Increasing-Jumps (EI-Jumps)* and *Unitary-Advance-Jumps (UA-Jumps)*. Briefly, EI-Jumps allow us to satisfy a suitable invariant:

[*Inv-EI*] Whenever a Scan-Phase is executed (each time that a Value-Iteration is invoked), an energy-level $f(v)$ strictly increases for at least one $v \in V$. There will be no *vain* Scan-Phase (i.e., such that all the energy-levels stand still); so, δ will be applied (successfully) at least once per Scan-Phase. Therefore, ψ_{ρ^J} will be monotone increasing (except at the steps of backtracking introduced next, but there will be at most $|V|$ of them). □

Indeed, the UA-Jumps are employed so to scroll through $\mathscr{F}_{|V|}$ only *when* and *where* it is really necessary. Consider the following facts.

– Suppose that Algorithm 1 came at the end of the $(i, s-1)$-th Scan-Phase, for some $i \in [W^-, W^+]$; recall that $F_{s-1} = 1$, so $w_{i,s-1} = w'_{i,s-1}$ is integral. Then, Algorithm 0 would scroll through $\mathscr{F}_{|V|}$ entirely, by invoking one Scan-Phase per each term, going from the $(i+1,1)$-th to the $(i+1, s-1)$-th, meanwhile testing whether a transition of winning regions occurs; notice, $w_{i+1,s-1}$ is integral again. Instead, to UA-Jump means to jump in advance (proactively) from $w_{i,s-1}$ to $w_{i+1,s-1}$, by making a Scan-Phase on input $\Gamma_{i+1,s-1}$, thus skipping all those from the $(i+1,1)$-th to the $(i+1, s-2)$-th one. After that, Algorithm 1 needs to *backtrack* to $w_{i,s-1}$, and to scroll through $\mathscr{F}_{|V|}$, if and only if $\mathscr{W}_0(\Gamma_{w_{i,s-1}}) \cap \mathscr{W}_1(\Gamma_{w_{i+1,s-1}}) \neq \emptyset$. Otherwise, it is safe to keep the search going on, from $w_{i+1,s-1}$ on out, making another UA-Jump to $w_{i+2,s-1}$. The backtracking step may happen at most $|V|$ times overall, because some value $\nu(v)$ is assigned to some $v \in V$ at each time. So, Algorithm 1 scrolls entirely through $\mathscr{F}_{|V|}$ at most $|V|$ times; i.e., only *when* it is really necessary.

– Remarkably, when scrolling through $\mathscr{F}_{|V|}$, soon after the above mentioned backtracking step, the corresponding sequence of Value-Iterations really need to lift-up again (more slowly) *only* the energy-levels of the sub-arena of Γ that is induced by $S \triangleq \mathscr{W}_0(\Gamma_{w_{i,s-1}}) \cap \mathscr{W}_1(\Gamma_{w_{i+1,s-1}})$. All the energy-levels of the vertices in $V \setminus S$ can be confirmed and left unchanged during the UA-Jump's backtracking step; and they will all stand still, during the forthcoming sequence of Value-Iterations (at least, until a new EI-Jump will occur), as they were computed just *before* the occurence of the UA-Jump's backtracking step. This is why Algorithm 1 scrolls through $\mathscr{F}_{|V|}$ only *where* it is really necessary.

– Also, Algorithm 1 succeeds at *interleaving* EI-Jumps and UA-Jumps, thus making only one single pass through ρ^J (plus the backtracking steps).

Altogether these facts are going to reduce the running time considerably.

Definition 1 (ℓ^1_Γ). *Given an input MPG Γ, let $\ell^1_\Gamma(v)$ be the total number of times that the energy-lifting operator $\delta(\cdot, v)$ is applied to any $v \in V$ by Algorithm 1 (notice that it will be applied only at line 3 of* `J-VI`*(), see SubProcedure 4).*

Then, the following remark holds on Algorithm 1.

Remark 3. *Jumping is not heuristic, the theoretical running time of the procedure improves exactly, from $\Theta(|V|^2|E|W+\sum_{v\in V}\texttt{deg}_\Gamma(v)\cdot\ell^0_\Gamma(v))$ (Comin and Rizzi, 2016) to $O(|E|\log|V|)+\Theta\big(\sum_{v\in V}\texttt{deg}_\Gamma(v)\cdot\ell^1_\Gamma(v)\big)$ (Algorithm 1), where $\ell^1_\Gamma\leq(|V|-1)|V|W$; which is still $O(|V|^2|E|W)$ in the worst-case, but it isn't known to be $\Omega(|V|^2|E|W)$ generally. In practice, this reduces the magnitude of ℓ_Γ significantly, i.e., $\ell^1_\Gamma\ll\ell^0_\Gamma$ is observed in our experiments (see SubSection 4.4).*

To achieve this, we have to overcome some subtle issues. Firstly, we show that it is unnecessary to re-initialize the Value-Iteration at each Scan-Phase (this would cost $\Omega(|E|)$ each time otherwise), even when making wide jumps in ρ. Instead, it will be sufficient to perform an initialization phase only at the beginning, by paying only $O(|E|\log|V|)$ total time and a linear space in pre-processing. For this, we will provide a suitable readjustment of the Value-Iteration; it is named $\texttt{J-VI}()$ (SubProcedure 4). Briefly, the Value-Iteration of Brim et al. (2011) employs an array of counters, $\texttt{cnt}:V_0\to\mathbb{N}$, in order to check in time $O(|N^{\text{in}}_\Gamma(v)|)$ which vertices $u\in N^{\text{in}}_\Gamma(v)\cap V_0$ have become inconsistent (soon after that the energy-level $f(v)$ was increased by applying $\delta(f,v)$ to some $v\in V$), and should therefore be added to the list L^{inc} of inconsistent vertices. One subtle issue here is that, when going from the $\texttt{prev}_{\rho^J}(i,j)$-th to the (i,j)-th Scan-Phase, the *coherency* of $\texttt{cnt}$ can break (i.e., $\texttt{cnt}$ may provide false-positives, thus classifying a vertex as consistent when it isn't really so). This may happen when $w_{\texttt{prev}_{\rho^J}(i,j)}>w_{(i,j)}$ (which is always the case, except for the UA-Jump's backtracking steps). This is even amplified by the EI-Jumps, as they may lead to wide jumps in ρ. The algorithm in Comin and Rizzi (2016) recalculates $\texttt{cnt}$ from scratch, at the beginning of each Scan-Phase, thus paying $\Omega(|E|)$ time per each. In this work, we show how to keep $\texttt{cnt}$ coherent throughout the Jumping Scan-Phases, efficiently. Actually, even in Algorithm 1 the coherency of $\texttt{cnt}$ can possibly break, but Algorithm 1 succeeds at *repairing* all the incoherencies that may happen during the whole computation in $\Theta(|E|)$ *total* time – just by paying $O(|E|\log|V|)$ time in pre-processing. This is a very convenient trade-off. At this point we should begin entering into the details of Algorithm 1.

Jumper. We employ a container data-structure, denoted by J. It comprises a bunch of arrays, maps, plus an integer variable $J.i$. Concerning maps, the key universe is V or E; i.e., keys are restricted to a narrow range of integers ($[1,|V|]$ or $[1,|E|]$, depending on the particular case).

We suggest direct addressing: the value binded to a key $v\in V$ (or $(u,v)\in E$) is stored at $A[v]$ (resp., $A[(u,v)]$); if there is no binding for key v (resp., (u,v)), the cell stores a sentinel, i.e., $A[v]=\bot$. Also, we would need to iterate efficiently through A (i.e., without having to scroll entirely through A). This is easy to implement by handling pointers in a suitable way; one may also keep a list L_A associated to A, explicitly, storing one element for each $(k,v)\neq\bot$ of A; every time that an item is added to or removed from A, then L_A is updated accordingly, in time $O(1)$ (by handling pointers). The *last* entry inserted into A (the key of which isn't already binded at insertion time) goes in *front* of L_A. We say that $L\triangleq(A,L_A)$ is an *array-list*, and we dispose of the following operations: $\texttt{insert}((k,\texttt{val}),L)$, which binds $\texttt{val}$ to k by inserting $(k,\texttt{val})$ into L (if any $(k,\texttt{val}')$ is already in L, then $\texttt{val}'$ gets overwritten by $\texttt{val}$); $\texttt{remove}(k,L)$ deletes an entry $(k,\texttt{val})$ from L; $\texttt{pop\_front}(L)$, removes from L the *last* $(k,\texttt{val})$ that was inserted (and whose key was not already binded at the time of the insertion, i.e., the *front*) also returning it; $\texttt{for\_each}\big((k,\texttt{val})\in L\big)$ iterates through the entries of L efficiently (i.e., skipping the sentinels). Notice, any sequence of $\texttt{insert}$ and $\texttt{pop\_front}$ on L implements a LIFO policy.

So, J comprises: an integer variable $J.i$; an array $J.f:V\to\mathbb{Q}$; an array $J.\texttt{cnt}:V_0\to\mathbb{N}$; an array $J.\texttt{cmp}:\{(u,v)\in E\mid u\in V_0\}\to\{\texttt{T},\texttt{F}\}$; a bunch of array-lists, $L_f:V\to\mathbb{N}$, and $L^{\text{inc}},L^{\text{inc}}_{\text{nxt}},L^{\text{inc}}_{\text{cpy}},L_\top:V\to\{*\}$; finally, a special array-list L_ω indexed by $\{w_e\mid e\in E\}$, whose values are in turn (classical, linked) lists of arcs, denoted L_α; L_ω is filled in pre-processing as follows: $(\hat{w}, L_\alpha) \in L_\omega$ *iff* $L_\alpha = \{e \in E \mid w_e = \hat{w}\}$. The subprocedure `init_jumper()` (SubProcedure 1) takes care of initializing J.

SubProcedure 1: Init Jumper J

SubProcedure *init_jumper*(J, Γ)
input : Jumper J, an MPG Γ.
1 $L_f, L^{\text{inc}}, L^{\text{inc}}_{\text{nxt}}, L^{\text{inc}}_{\text{cpy}}, L_\top, L_\omega \leftarrow \emptyset$;
2 **foreach** $v \in V$ **do**
3 $J.f[v] \leftarrow 0$;
4 **if** $v \in V_0$ **then**
5 $J.\mathtt{cnt}[v] \leftarrow |N^{\text{out}}_\Gamma(v)|$;
6 **foreach** $(u, v, w) \in E$ **do**
7 **if** $v \in V_0$ **then**
8 $J.\mathtt{cmp}[(u, v)] \leftarrow \mathtt{T}$;
9 **if** $L_\omega[w] = \bot$ **then**
10 `insert`$((w, \emptyset), L_\omega)$;
11 `insert`$((u, v), L_\omega[w])$;
12 Sort L_ω in increasing order w.r.t. the keys w;

At the beginning, all array-lists are empty (line 1). For every $v \in V$ (line 2), we set $J.f[u] = 0$ and, if $v \in V_0$, then $J.\mathtt{cnt}[v] \leftarrow |N^{\text{out}}_\Gamma(v)|$ (lines 3-5). Then, each arc $(u, v, w) \in E$ is flagged as *compatible*, i.e., $J.\mathtt{cmp}[(u, v)] \leftarrow \mathtt{T}$ (lines 6-8); also, if L_ω doesn't contain an entry already binded to $w(u, v)$, then an empty list of arcs is inserted into L_ω as an entry $(w, \emptyset)$ (lines 9-10); then, in any case, the arc (u, v) is added to the unique L_α which is binded to $w = w(u, v)$ in L_ω (line 11). At the end (line 12), all the elements of L_ω are sorted in increasing order w.r.t. their weight keys, w_e for $e \in E$ (e.g., (W^-, L_α) goes in front of L_ω). This concludes the initialization of J; it takes $O(|E| \log |V|)$ time and $\Theta(|V| + |E|)$ space.

Main Procedure: *solve_MPG*(). The main procedure of Algorithm 1 is organized as follows. Firstly, the algorithm performs an initialization phase; which includes `init_jumper`(J, Γ).

The variables $\mathscr{W}_0, \mathscr{W}_1, \nu, \sigma_0^*$ are initially empty (line 1). Also, $W^- \leftarrow \min_{e \in E} w_e$, $W^+ \leftarrow \max_{e \in E} w_e$ (line 2). And F is a reference to the Farey's terms, say $\{F[j] \mid j \in [0, s-1]\} = \mathscr{F}_{|V|}$, and $s \leftarrow |\mathscr{F}_{|V|}|$ (line 3). At line 4, J is initialized by `init_jumper`(J, Γ) (SubProcedure 1).

Then the Scan-Phases start.

Algorithm 1: Main Procedure

Procedure *solve_MPG*(Γ)
input : An MPG $\Gamma = (V, E, w, \langle V_0, V_1 \rangle)$.
output: $(\mathscr{W}_0, \mathscr{W}_1, \nu, \sigma_0^*)$.
1 $\mathscr{W}_0, \mathscr{W}_1, \nu, \sigma_0^* \leftarrow \emptyset$; */*Init Phase*/*
2 $W^- \leftarrow \min_{e \in E} w_e$; $W^+ \leftarrow \max_{e \in E} w_e$;
3 $F \leftarrow$ reference to $\mathscr{F}_{|V|}$; $s \leftarrow |\mathscr{F}_{|V|}|$;
4 `init_jumper`(J, Γ);
5 $i \leftarrow W^- - 1$; $j \leftarrow 1$; */*Jumping Scan-Phases*/*
6 **while** T **do**
7 **if** *ei-jump*(i, J) **then**
8 **if** $L^{inc} = \emptyset$ **then return** $(\mathscr{W}_0, \mathscr{W}_1, \nu, \sigma_0^*)$;
9 $(i, S) \leftarrow$ `ua-jumps`$(J.i, s, F, J, \Gamma)$;
10 $j \leftarrow 1$;
11 `J-VI`$(i, j, F, J, \Gamma[S])$;
12 `set_vars`$(\mathscr{W}_0, \mathscr{W}_1, \nu, \sigma_0^*, i, j, F, J, \Gamma[S])$;
13 `scl_back_f`(j, F, J);
14 $j \leftarrow j + 1$;

After setting $i \leftarrow W^- - 1$, $j \leftarrow 1$ (line 5), Algorithm 1 enters into a `while` loop (line 6), which lasts until both `ei-jump`$(i, J) = \mathtt{T}$ at line 7, *and* $L^{\text{inc}} = \emptyset$ at line 8, hold; in which case $(\mathscr{W}_0, \mathscr{W}_1, \nu, \sigma_0^*)$ is returned (line 8) and Algorithm 1 halts. Inside the `while` loop, `ei-jump`(i, J) (SubProcedure 6) is invoked (line 7). This checks whether or not to make an EI-Jump; if so, the ending point of the EI-Jump (the new value of i) is stored into $J.i$. This will be the starting point for making a sequence of UA-Jumps, which begins by invoking `ua-jumps`$(J.i, s, F, J, \Gamma)$ at line 9. When the `ua-jumps()` halts, it returns $(\hat{\imath}, S)$, where: $\hat{\imath}$ is the new value of i (line 9), for some $\hat{\imath} \geq J.i$; and S is a set of vertices such that $S = \mathscr{W}_0(\Gamma_{w_{\hat{\imath}-1, s-1}}) \cap \mathscr{W}_1(\Gamma_{w_{\hat{\imath}, s-1}})$. Next, $j \leftarrow 1$ is set (line 10), as Algorithm 1 is now completing the backtracking from $w_{\hat{\imath}, s-1}$ to $w_{\hat{\imath}, 1}$, in order to begin scrolling through $\mathscr{F}_{|V|}$ by running a sequence of `J-VI()` at line 11. Such a sequence of `J-VI()`s will last until the occurence of another EI-Jump at line 7, that in turn will lead to another sequence of UA-Jumps at line 9, and so on. So, a `J-VI()` (SubProcedure 4) is executed on input $(\hat{\imath}, j, F, J, \Gamma[S])$ at line 11. We remark that, during the `J-VI`$(i, j, F, J, \Gamma[S])$, the energy-levels are scaled up, from $\mathbb{Q}$ to $\mathbb{N}$; actually, from $J.f$ to $\lceil D_j \cdot J.f \rceil$, where D_j is the denominator of F_j. Also, `J-VI`$(i, j, F, J, \Gamma[S])$ (SubProcedure 4) is designed so that, when it halts, $L_\top = \mathscr{W}_0(\Gamma_{\mathtt{prev}_{\rho^J}(i,j)}) \cap \mathscr{W}_1(\Gamma_{i,j})$. Then, `set_vars()` is invoked on input $(\mathscr{W}_0, \mathscr{W}_1, \nu, \sigma_0^*, i, j, F, J, \Gamma[S])$ (line 12): this checks whether some value and optimal strategy needs to be assigned to ν and σ_0^* (respectively). Next, all of the energy-levels are scaled back, from $\mathbb{N}$ to $\mathbb{Q}$, and stored back into $J.f$: this is done by invoking $\texttt{scl\_back\_}f(j,F,J)$ (line 13). Finally, $j \leftarrow j+1$ (line 14) is assigned (to step through the sequence $\mathscr{F}_{|V|}$ during the `while` loop at line 7). This concludes `solve_MPG`(), which is the main procedure of Algorithm 1.

Set Values and Optimal Strategy. Let us provide the details of `set_vars`() (SubProcedure 2). It takes $(\mathscr{W}_0, \mathscr{W}_1, \nu, \sigma_0^*, i, j, F, \Gamma)$ in input, where $i \in [W^-, W^+]$ and $j \in [1, s-1]$. At line 1, $D = D_{j-1}$ is the denominator of F_{j-1}. Then, all of the following operations are repeated while $L_\top \neq \emptyset$ (line 2). Firstly, the front element u of $L_\top$ is popped (line 3); recall, it will turn out that $u \in \mathscr{W}_0(\Gamma_{i,j-1}) \cap \mathscr{W}_1(\Gamma_{i,j})$, thanks to the specs of `J-VI`() (SubProcedure 4). For this reason, the optimal value of u in the MPG Γ is set to $\nu(u) \leftarrow i + F[j-1]$ (line 4); and, if $\nu(u) \geq 0$, u is added to the winning region $\mathscr{W}_0$; else, to $\mathscr{W}_1$ (line 5). The correctness of lines 4-5 relies on Theorem 3. If $u \in V_0$ (line 6), it is searched an arc $(u,v) \in E$ that is compatible w.r.t. $D_{j-1} \cdot J.f$ in $\Gamma_{i,j-1}$ (line 8), i.e., it is searched some $v \in N_\Gamma^{\text{out}}(u)$ such that:

$$(D \cdot J.f[u]) \succeq (D \cdot J.f[v]) \ominus \texttt{get\_scl\_}w\big(w(u,v), i, j-1, F\big) \ \text{ (line 8);}$$

By Theorem 4, setting $\sigma_0^*(u) \leftarrow v$ (line 9) brings an optimal positional strategy for Player 0 in the MPG Γ. Here, $\texttt{get\_scl\_}w\big(w, i, j-1, F\big)$ simply returns $D_{j-1} \cdot (w(u,v) - i) - N_{j-1}$, where: N_{j-1} is the numerator of F_{j-1}, and D_{j-1} is its denominator. Thanks to how `J-VI`() (SubProcedure 4) is designed, at this point $J.f$ still stores the energy-levels as they were just *before* the last invocation of `J-VI`() made at line 11 of Algorithm 1; instead, the new energy-levels, those lifted-up during that same `J-VI`(), are stored into L_f. So, at this point, it will turn out that $\forall^{u \in V}\, J.f[u] = f^*_{i,j-1}(u)$.

SubProcedure 2: Set Values and Optimal Strategy

Procedure *set_vars*$(\mathscr{W}_0, \mathscr{W}_1, \nu, \sigma_0^*, i, j, F, J, \Gamma)$
 input : Winning sets $\mathscr{W}_0, \mathscr{W}_1$, values ν, opt. strategy σ_0^*, $i \in [W^-, W^+]$, $j \in [1, s-1]$, ref. F to $\mathscr{F}_{|V|}$, MPG Γ
1 $D \leftarrow$ denominator of $F[j-1]$;
2 **while** $L_\top \neq \emptyset$ **do**
3   $u \leftarrow \texttt{pop\_front}(L_\top)$;
4   $\nu(u) \leftarrow i + F[j-1]$;
5   **if** $\nu(u) \geq 0$ **then** $\mathscr{W}_0 \leftarrow \mathscr{W}_0 \cup \{v\}$; **else** $\mathscr{W}_1 \leftarrow \mathscr{W}_1 \cup \{v\}$;
6   **if** $u \in V_0$ **then**
7   **for** $v \in N_\Gamma^{out}(u)$ **do**
8   **if** $(D \cdot J.f[u]) \succeq (D \cdot J.f[v]) \ominus$ *get_scl_w*$\big(w(u,v), i, j-1, F\big)$ **then**
9         $\sigma_0^*(u) \leftarrow v$; **break**;

This actually concludes the description of `set_vars`() (SubProcedure 2).

Indeed, the role of L_f is precisely that to allow the `J-VI`() to lift-up the energy-levels during the (i,j)-th Scan-Phase, meanwhile preserving (inside $J.f$) those computed at the $(i, j-1)$-th one (because `set_vars`() needs them in order to rely on Theorem 4). As mentioned, when `set_vars`() halts, all the energy-levels are scaled back, from $\mathbb{N}$ to $\mathbb{Q}$, and stored back from L_f into $J.f$ (at line 13 of Algorithm 1, see $\texttt{scl\_back\_}f$() in SubProcedure 3).

We remark at this point that all the arithmetics of Algorithm 1 can be done in $\mathbb{Z}$.

Now, let us detail the remaining subprocedures, those governing the Jumps and those concerning the energy-levels and the `J-VI`(). Since the details of the former rely significantly on those of the latter two, we proceed by discussing firstly how the energy-levels are handled by the `J-VI`() (see SubProcedure 4 and 3).

J-Value-Iteration. `J-VI`() is similar to the Value-Iteration of Brim et al. (2011). Still, there are some distinctive features. The `J-VI`() takes in input two indices $i \in [W^-, W^+]$ and $j \in [1, s-1]$, a reference F to $\mathscr{F}_{|V|}$, (a reference to) the Jumper J, (a reference to) the input arena Γ. Basically, J-VI(i,j,F,J,Γ) aims at computing the least-SEPM of the reweighted EG $\Gamma_{i,j}$. For this, it relies on a (slightly revisited) *energy-lifting* operator $\delta : [V \to \mathscr{C}_\Gamma] \times V \to [V \to \mathscr{C}_\Gamma]$. The array-list employed to keep track of the inconsistent vertices is L^{inc}. It is assumed, as a pre-condition, that L^{inc} is already initialized when J-VI() starts. We will show that this pre-condition holds thanks to how $L^{\text{inc}}_{\text{nxt}}$ is managed. Recall, Algorithm 1 is going to perform a sequence of invocations to J-VI(). During the execution of any such invocation of J-VI(), the role of $L^{\text{inc}}_{\text{nxt}}$ is precisely that of collecting, in advance, the initial list of inconsistent vertices for the *next*[2] J-VI(). Rephrasing, the k-th invocation of J-VI() takes care of initializing L^{inc} for the $k+1$-th invocation of J-VI(), and this is done thanks to $L^{\text{inc}}_{\text{nxt}}$.

Also, the energy-levels are managed in a special way. The *inital* energy-levels are stored inside $J.f$ (as a pre-condition). Again, the k-th invocation of J-VI() takes care of initializing the initial energy-levels for the $k+1$-th one: actually, those computed at the end of the k-th J-VI() will become the initial energy-levels for the $k+1$-th one (subject to a rescaling). In this way, Algorithm 1 will succeed at amortizing the cost of all invocations of J-VI(). As mentioned, since $J.f$ stores rational-scalings, and $\Gamma_{i,j}$ is weighted in $\mathbb{Z}$, the J-VI() needs to scale everything up, from $\mathbb{Q}$ to $\mathbb{N}$, when it reads the energy-levels out from $J.f$. So, $J.f$ is accessed *read-only* during the J-VI(): we want to update the energy-levels by applying δ, but still we need a back-up copy of the initial energy-levels (because they are needed at line 8 of set_vars(), SubProcedure 2). Therefore, a special subprocedure is employed for accessing energy-levels during J-VI(), it is named get_scl_f() (SubProcedure 3); moreover, an array-list L_f is employed, whose aim is that to store the current energy-levels, those lifted-up during the J-VI(). SubProcedure 3 shows get_scl_f(), it takes: $u \in V$, some $j \in [1, s-1]$, a reference F to $\mathscr{F}_{|V|}$, and (a reference to) J.

get_scl_f() goes as follows. If $L_f[u] = \perp$ (line 1), the denominator D of F_j is taken (line 2), and $f \leftarrow \lceil D \cdot J.f[u] \rceil$ is computed (line 3); a (new) entry (v, f) is inserted into L_f (line 4). Finally, in any case, $L_f[v]$ is returned (line 5).

SubProcedure 3: Energy-Levels

```
  SubProcedure get_scl_f(v, j, F, J)
      input: v ∈ V, j ∈ [1, s−1],
             F is a ref. to 𝓕_|V|, J is Jumper.
1     if L_f[v] = ⊥ then
2         D ← denominator of F[j];
3         f ← ⌈D · J.f[v]⌉;
4         insert((v, f), L_f);
5     return L_f[v];

  SubProcedure scl_back_f(j, F, J)
      input: j ∈ [0, s−1], F is a ref. to Farey's terms,
             J is Jumper.
1     D ← denominator of F[j];
2     while L_f ≠ ∅ do
3         (v, f) ← pop_front(L_f);
4         J.f[v] ← f/D;
```

As mentioned, at line 13 of Algorithm 1, $J.f$ will be overwritten by scaling back the values that are stored in L_f. This is done by scl_back_f() (SubProcedure 3): at line 1, D is the denominator of F_j; then, L_f is emptied, one element at a time (line 2); for each $(v, f) \in L_f$ (line 3), the rational f/D is stored back to $J.f[v]$ (line 4). This concludes scl_back_f().

Next, J-VI() takes in input: $i \in [W^-, W^+]$, $j \in [1, s-1]$, a reference F to $\mathscr{F}_{|V|}$, (a reference to) the Jumper J, and (a reference to) the input MPG Γ. At line 1, J-VI() enters into a while loop which lasts while $L^{\text{inc}} \neq \emptyset$. The front vertex $v \leftarrow$ pop_front(L^{inc}) is popped from L^{inc} (line 2). Next, the energy-lifting operator δ is applied to v by invoking apply_$\delta(v, i, j, F, J, \Gamma)$ (line 3).

There inside (at line 1 of apply_δ()), the energy-level of v is lifted-up as follows:

$$f_v \leftarrow \begin{cases} \min\big\{\texttt{get\_scl\_}f\big(v', j, F, J\big) \ominus \texttt{get\_scl\_}w\big(w(v,v'), i, j, F\big) \mid v' \in N^{\text{out}}_\Gamma(v)\big\}, & \text{if } v \in V_0; \\ \max\big\{\texttt{get\_scl\_}f\big(v', j, F, J\big) \ominus \texttt{get\_scl\_}w\big(w(v,v'), i, j, F\big) \mid v' \in N^{\text{out}}_\Gamma(v)\big\}, & \text{if } v \in V_1. \end{cases}$$

[2] i.e., the subsequent invocation (in the above mentioned sequence of J-VI()) that will be performed, either at line 12 of solve_MPG() (Algorithm 1), or at line 3 of ua-jumps() (SubProcedure 7).

Then, f_v is stored inside L_f (notice, not in $J.f$), where it is binded to the key v (line 2). The control turns back to J-VI(). The current energy-level of v is retrieved by $f_v \leftarrow \texttt{get\_scl\_}f(v, j, F, J)$ (line 4). If $f_v \neq \top$ (line 5), then v is inserted into $L^{\text{inc}}_{\text{nxt}}$ (if it isn't already in there) (line 6); moreover, if $v \in V_0$, then $J.\texttt{cnt}[v]$ and $\{J.\texttt{cmp}[(v, v')] \mid v' \in N^{\text{out}}_{\Gamma}(v)\}$ are recalculated from scratch, by invoking $\texttt{init\_cnt\_cmp}(v, i, j, F, J, \Gamma)$ (line 7, see SubProcedure 5). Else, if $f_v = \top$ (line 8), then v is stored into $L_\top$ (line 9); and if $L^{\text{inc}}_{\text{nxt}}[v] \neq \bot$ in addition, then v is removed from $L^{\text{inc}}_{\text{nxt}}$ (line 10).

At this point it is worth introducing the following notation concerning energy-levels.

Definition 2. *For any step of execution ι and for any variable x of Algorithm 1, the state of x at step ι is denoted by x^ι. Then, the* current energy-levels *at step ι are defined as follows:*

$$\forall^{v\in V} f^{c:\iota}(v) \triangleq \begin{cases} L^\iota_f[v], & \textit{if } L^\iota_f[v] \neq \bot; \\ \lceil D_{j^\iota} \cdot J.f^\iota[v] \rceil, & \textit{otherwise}. \end{cases}$$

where D_{j^ι} is the denominator of F_{j^ι}. If ι is implicit, the current energy-levels *are denoted by f^c.*

SubProcedure 4: J-Value-Iteration

Procedure $J\text{-}VI(i, j, F, J, \Gamma)$
    input : $i \in [W^-, W^+]$ and $j \in [1, s-1]$, F is a ref. to Farey's terms, J is Jumper, Γ is an MPG.
1   **while** $L^{inc} \neq \emptyset$ **do**
2       $v \leftarrow \texttt{pop\_front}(L^{\text{inc}})$;
3       $\texttt{apply\_}\delta(v, i, j, F, J, \Gamma)$;
4       $f_v \leftarrow \texttt{get\_scl\_}f(v, j, F, J)$;
5   **if** $f_v \neq \top$ **then**
6   **if** $L^{inc}_{nxt}[v] = \bot$ **then** $\texttt{insert}(v, L^{\text{inc}}_{\text{nxt}})$;
7   **if** $v \in V_0$ **then** $\texttt{init\_cnt\_cmp}(v, i, j, F, J, \Gamma)$;
8   **else**
9           $\texttt{insert}(v, L_\top)$;
10   **if** $L^{inc}_{nxt}[v] \neq \bot$ **then** $\texttt{remove}(v, L^{\text{inc}}_{\text{nxt}})$;
11   **foreach** $u \in N^{in}_{\Gamma}(v)$ **do**
12          $f_u \leftarrow \texttt{get\_scl\_}f(u, j, F, J)$;
13          $\Delta_{u,v} \leftarrow f_v \ominus \texttt{get\_scl\_}w(w(u, v), i, j, F)$;
14   **if** $L^{inc}[u] = \bot$ **and** $f_u < \Delta_{u,v}$ **then**
15   **if** $u \in V_0$ **and** $J.cmp[(u, v)] = T$ **then**
16                  $J.\texttt{cnt}[u] \leftarrow J.\texttt{cnt}[u] - 1$;
17                  $J.\texttt{cmp}[(u, v)] \leftarrow \texttt{F}$;
18   **if** $u \in V_1$ **OR** $J.cnt[u] = 0$ **then** $\texttt{insert}(u, L^{\text{inc}})$;
19 $\texttt{swap}(L^{\text{inc}}, L^{\text{inc}}_{\text{nxt}})$;

SubProcedure $apply\_\delta(v, i, j, F, J, \Gamma)$
    input : $v \in V$, $i \in [W^-, W^+]$, $j \in [1, s-1]$, F is a ref. to Farey, J is Jumper, Γ is an MPG.
1   $f_v \leftarrow \begin{cases} \min\{\texttt{get\_scl\_}f(v', j, F, J) \ominus \texttt{get\_scl\_}w(w(v, v'), i, j, F) \mid v' \in N^{\text{out}}_{\Gamma}(v)\}, & \text{if } v \in V_0; \\ \max\{\texttt{get\_scl\_}f(v', j, F, J) \ominus \texttt{get\_scl\_}w(w(v, v'), i, j, F) \mid v' \in N^{\text{out}}_{\Gamma}(v)\}, & \text{if } v \in V_1. \end{cases}$
2   $\texttt{insert}((v, f_v), L_f)$;

Remark 4. *Recall, the role of L^{inc}_{nxt} and that of the swap() (line 19) is precisely that of initializing, in advance, the list of inconsistent vertices L^{inc} for the* next *J-VI(); because the J-VI() assumes a correct initialization of L^{inc} as a pre-condition.*

We argue in Proposition 6 and Lemma 1 that, when J-VI() halts –say at step h– it is necessary to initialize $J.L^{inc}$ for the next *J-VI() by including (at least) all the $v \in V$ such that: $0 < f^{c:h}(v) \neq \top$.*

Notice, if $L^{\text{inc}}_{\text{nxt}} = \emptyset$ holds just before the swap() at line 19, then $L^{\text{inc}} = \emptyset$ holds soon after; therefore, in that case yet another *EI-Jump* will occur (at line 7 of Algorithm 1) and eventually some other vertices will be inserted into L^{inc} (see the details of SubProcedure 6).

We shall provide the details of init_cnt_inc(v, i, j, F, J) (line 7) very soon hereafter.

But let us first discuss the role that is played by J.cnt and J.cmp during the J-VI().

SubProcedure 5: Counters and Cmp Flags

```
SubProcedure init_cnt_cmp(u, i, j, F, J, Γ)
    input : u ∈ V_0, i ∈ [W^-, W^+], j ∈ [1, s−1], F is a ref. to
            Farey, J is Jumper, Γ is an MPG.
1   c_u ← 0;
2   foreach v ∈ N_Γ^out(u) do
3       f_u ← get_scl_f(u, j, F, J);
4       f_v ← get_scl_f(v, j, F, J);
5       if f_u ⪰ f_v ⊖ get_scl_w(w(u, v), i, j, F) then
6           c_u ← c_u + 1;
7           J.cmp[(u, v)] ← T;
8       else J.cmp[(u, v)] ← F;
9   J.cnt[u] ← c_u;
```

From line 11 to line 18, J-VI() explores $N_\Gamma^{\text{in}}(v)$ in order to find all the $u \in N_\Gamma^{\text{in}}(v)$ that may have become inconsistent soon after the energy-lifting δ that was applied to v (before, at line 3). For each $u \in N_\Gamma^{\text{in}}(v)$ (line 11), the energy-level $f_u \leftarrow$ get_scl_$f(u, j, F, J)$ is considered (line 12), also, $\Delta_{u,v} \leftarrow f_v \ominus w'_{i,j}(u, v)$ is computed (line 13), where $f_v \leftarrow$ get_scl_$f(v, j, F, J)$; if $f_u < \Delta_{u,v}$ (i.e., in case (u, v) is now incompatible w.r.t. f^{c} in $\Gamma_{i,j}$) *and* $L^{\text{inc}}[u] = \perp$ holds (line 14), then:

– If $u \in V_0$ *and* (u, v) was not already incompatible *before* (i.e., if J.cmp$[(u, v)] = $ T at line 15, then: J.cnt$[u]$ is decremented (line 16), and J.cmp$[(u, v)] \leftarrow$ F is assigned (line 17). (This is the role of the J.cnt and J.cmp flags).

– After that, if $u \in V_1$ *or* J.cnt$[u] = 0$, then u is inserted into L^{inc} (line 18).

When the while loop (at line 1) ends, the (references to) L^{inc} and $L^{\text{inc}}_{\text{nxt}}$ are *swapped* (line 19) (one is assigned to reference the other and vice-versa, in $O(1)$ time by interchanging pointers).

The details of init_cnt_cmp(u, i, j, F, J, Γ) (line 7), where $u \in V_0$, are given in SubProcedure 5. At line 1, $c_u \leftarrow 0$ is initialized. For each $v \in N_\Gamma^{\text{out}}(u)$ (line 2), it is checked whether (u, v) is compatible with respect to the current energy-levels; i.e., whether or not $f_u \succeq f_v \ominus w'_{i,j}(u, v)$, holds for $f_u \leftarrow$ get_scl_$f(u, j, F, J) = f^{\text{c}}(u)$ and $f_v \leftarrow$ get_scl_$f(v, j, F, J) = f^{\text{c}}(v)$ (lines 3-5); if (u, v) is found to be compatible, then c_u is incremented (line 6) and J.cmp$[(u, v)] \leftarrow$ T is assigned (line 7); otherwise, (c_u stands still and) it is set J.cmp$[(u, v)] \leftarrow$ F (line 8). At the very end, it is finally set J.cnt$[u] \leftarrow c_u$ (line 9).

Concerning J.cmp and J.cnt, it is now worth defining a formal notion of *coherency*.

Definition 3. *Let ι be any step of execution of Algorithm 1. Let $i \in [W^-, W^+]$, $j \in [0, s-1]$, $u \in V_0$ and $v \in N_\Gamma^{out}(u)$. We say that $J.cmp^\iota[(u, v)]$ is* coherent *w.r.t. $f^{c:\iota}$ in $\Gamma_{i,j}$ when it holds:*

$$J.\mathit{cmp}^\iota[(u, v)] = T \text{ iff } f^{c:\iota}(u) \succeq f^{c:\iota}(v) \ominus w'_{i,j}(u, v).$$

Also, we say that $J.cnt^\iota[u]$ is coherent *w.r.t. $f^{c:\iota}$ in $\Gamma_{i,j}$ when:*

$$J.\mathit{cnt}^\iota[u] = \left|\left\{(u, v) \in E \mid f^{c:\iota}(u) \succeq f^{c:\iota}(v) \ominus w'_{i,j}(u, v)\right\}\right|.$$

We say that $J.cmp^\iota$ is coherent *when $\forall (u \in V_0 \setminus L^{inc^\iota})\ \forall (v \in N_\Gamma^{out}(u))\ J.cmp^\iota[(u, v)]$ is coherent; and we say that $J.cnt^\iota$ is* coherent *when $\forall (u \in V_0 \setminus L^{inc^\iota})\ J.cnt^\iota[u]$ is coherent.*

Finally, when something is not coherent, it is incoherent*. Remark: the step ι can be implicit.*

Remark 5. *In the Value-Iteration (Brim et al., 2011), the consistency checking of $(u, v) \in E$ (line 14) is explicit: an inequality like "$f(u) \succeq f(v) \ominus w(u, v)$" is tested; thus, neither the cmp flags nor an explicit notion of coherency are needed. So, why we introduced cmp flags and coherency? Observe, at line 14 of J-VI(), it doesn't make much sense to check "$f(u) \succeq f(v) \ominus w(u, v)$" in our setting. Consider the following facts: (1) of course the values of $w'_{i,j}$ depend on the index (i, j) of the current Scan-Phase; (2) therefore, going from one Scan-Phase to* *the next one (possibly, by Jumping), some counters may become incoherent, because $w_{i',j'} < w_{i,j}$ if $(i',j') > (i,j)$; but in the Value-Iteration (Brim et al., 2011) the only possible source of incoherency was the application of $\delta(\cdot,v)$; in Algorithm 1, going from one Scan-Phase to the next, we have an additional source of incoherency. (3) still,* `J-VI`() *can't afford to re-initialize* `cnt` $: V \to \mathbb{N}$ *each time that it is needed, as this would cost $\Omega(|E|)$. So, if $(u,v) \in E$ is found incompatible (at line 14 of* `J-VI`()*) after the application of $\delta(\cdot,v)$ (line 3), how do we know whether or not (u,v) was already incompatible* before *the (last) application of $\delta(\cdot,v)$? We suggest to adopt the* `cmp` *flags, one bit per arc is enough.*

To show correctness and complexity, we firstly assume that whenever `J-VI`(i,j,F,J,Γ) is invoked the following three pre-conditions are satisfied:

(PC-1) $L_f = \emptyset$ and $\forall^{v\in V}\, f^{\mathrm{c}}(v) \preceq f^*_{w'_{i,j}}(v)$;

(PC-2) $L^{\mathrm{inc}} = \mathrm{Inc}(f^{\mathrm{c}}, i, j)$;

(PC-3) $J.$`cnt` and $J.$`cmp` are *coherent* w.r.t. f^{c} in $\Gamma_{i,j}$.

After having described the internals of the EI-Jumps, we'll show how to ensure (a slightly weaker, but still sufficient formulation of) (PC-1), (PC-2), (PC-3).

Assuming the pre-conditions, similar arguments as in [Brim et al. (2011), Theorem 4] show that `J-VI`() computes the least-SEPM of the EG $\Gamma_{i,j}$ in time $O(|V|^2|E|W)$ and linear space.

Proposition 5. *Assume that* `J-VI`() *is invoked on input (i,j,F,J,Γ), and that (PC-1), (PC-2), (PC-3) hold at invocation time. Then,* `J-VI`() *halts within the following time bound:*

$$\Theta\Big(\sum_{v\in V} \mathit{deg}_{\Gamma}(v)\cdot \ell^1_{\Gamma_{i,j}}(v)\Big) = O\big(|V|^2|E|W\big),$$

where $0 \leq \ell^1_{\Gamma_{i,j}}(v) \leq (|V|-1)|V|W$ is the number of times that the energy-lifting operator δ is applied to any $v \in V$, at line 3 of `J-VI`() *on input (i,j,F,J,Γ). The working space is $\Theta(|V|+|E|)$.*

When `J-VI`() *halts, f^{c} coincides with the* least-*SEPM of the reweighted EG $\Gamma_{i,j}$.*

Proof. The argument is very similar to that of [Brim et al. (2011), Theorem 4], but there are some subtle differences between the `J-VI`() and the Value-Iteration of Brim, *et al.*:

(1) `J-VI`() employs $J.f$ and L_f to manage the energy-levels; however, one can safely argue by always referring to the current energy-levels f^{c}.

(2) `J-VI`() has no initialization phase; however, notice that the pre-conditions (PC-1), (PC-2), (PC-3) ensure a correct initialization of it.

(3) `J-VI`() employs $J.$`cmp` in order to test the consistency state of the arcs (see line 15 and 17 of `J-VI`()); but it is easy to see that, assuming (PC-3), this is a correct way to go.

Let us provide a sketch of the proof of correctness. As already observed in [Brim et al. (2011), Lemma 7], the energy-lifting operator δ is $\sqsubseteq$-*monotone* (i.e., $\delta(f,v) \sqsubseteq \delta(g,v)$ for all $f \sqsubseteq g$). Next, the following invariant is maintained by `J-VI`() (Subprocedure 4) at line 1.

Inv-JVI. $\forall$(iteration ι of line 1 of `J-VI`(i,j,F,J,Γ)) $\forall(u \in V \setminus J.L^{\mathrm{inc}^{\iota}})$ $\forall(v \in N^{\mathrm{out}}_{\Gamma}(u))$:

(*i*) $\delta(f^{\mathrm{c}:\iota}, u) = f^{\mathrm{c}:\iota}$;

(*ii*) if $u \in V_0 \setminus J.L^{\mathrm{inc}^{\iota}}$, then $J.$`cnt`$^{\iota}[u]$ and $J.$`cmp`$^{\iota}[(u,v)]$ are both coherent w.r.t. $f^{\mathrm{c}:\iota}$ in $\Gamma_{i,j}$.

It is not difficult to prove that *Inv-JVI* holds. The argument is almost the same as in [Brim et al. (2011), Lemma 8]; the only noticeable variations are: (a) the `J-VI`() employs $J.$`cmp` in order to flag the compatibility status of the arcs; (b) the reference energy-level is $f^{\rm c}$; (c) at the first iteration of line 1 of J-VI$()$, the *Inv-JVI* holds thanks to (PC-2) and (PC-3).

Termination is enforced by three facts: (i) every application of the energy-lifting operator (line 3) strictly increases the energy-level of one vertex v; (ii) the co-domain of SEPMs is finite.

Correctness follows by applying the Knaster-Tarski's Fixed-Point Theorem. Indeed, at halting time, since δ is $\sqsubseteq$-monotone, and since (PC-1) and *Inv-JVI* hold, then we can apply Knaster-Tarski's Fixed-Point Theorem to conclude that, when J-VI$()$ halts at step h (say), then $f^{{\rm c}:h}$ is the unique least fixpoint of (simultaneously) all operators $\delta(\cdot,v)$ for all $v\in V$, i.e., $f^{{\rm c}:h}$ is the least-SEPM of the EG $\Gamma_{i,j}$.

So, when J-VI$()$ halts, it holds that $\forall^{v\in V} f^{{\rm c}:h}(v) = f^*_{w'_{i,j}}(v)$.

Concerning the time and space complexity, $\delta(\cdot,v)$ can be computed in time $O(|N^{\rm out}_\Gamma(v)|)$ (line 3) (see apply_$\delta()$ in SubProcedure 4); the updating of J.cnt and J.cmp, which is performed by init_cnt_cmp$()$ (line 7), also takes $O(|N^{\rm out}_\Gamma(v)|)$ time. Soon after that $\delta(\cdot,v)$ has been applied to $v\in V$ (line 3), the whole $N^{\rm in}_\Gamma(v)$ is explored for repairing incoherencies and for finding new inconsistent vertices (which is done from line 11 to line 18): this process takes $O(|N^{\rm in}_\Gamma(v)|)$ time. Therefore, if $\delta(\cdot,v)$ is applied $\ell^1_{\Gamma_{i,j}}(v)$ times to (any) $v\in V$ during the J-VI(i,j,F,J,Γ), the total time is $\Theta\big(\sum_{v\in V} \texttt{deg}_\Gamma(v)\cdot \ell^1_{\Gamma_{i,j}}(v)\big)$. The codomain of any SEPM of $\Gamma_{i,j}$ is at most $(|V|-1)W'$, for $W'=D_jW\leq |V|W$, where the additional factor $D_j\leq |V|$ comes from the scaled weights of $\Gamma_{i,j}$; thus, $\forall^{v\in V} 0\leq \ell^1_{\Gamma_{i,j}}(v)\leq (|V|-1)D_jW\leq (|V|-1)|V|W$. As already mentioned in Section 3, the Farey's term $F[j]$ can be computed at the beginning of J-VI$()$ in $O(1)$ time and space, from $F[j-1]$ and $F[j-2]$. Since $\sum_{v\in V}\texttt{deg}_\Gamma(v)=2|E|$, the running time is also $O(|V|^2|E|W)$. We check that J-VI$()$ works with $\Theta(|V|+|E|)$ space: $L^{\rm inc}$, $L^{\rm inc}_{\rm nxt}$, L_f, and $L_\top$ contain no duplicates, so they take $\Theta(|V|)$ space; the size of $J.f$ and J.cnt is $|V|$, that of J.cmp plus L_ω is $\Theta(|E|)$. □

Indeed, the J-VI$()$ keeps track of two additional array-lists, $L^{\rm inc}_{\rm nxt}$ and $L_\perp$. The role of $L^{\rm inc}_{\rm nxt}$ is to ensure (a slightly weaker formulation of) (PC-2): during the execution of Algorithm 1, the $\texttt{prev}_{\rho^J}(i,j)$-th invocation of J-VI$()$ handles $L^{\rm inc}_{\rm nxt}$ so to ensure that (a slightly weaker, but still sufficient form of) (PC-2) holds for the (i,j)-th invocation. However, the way in which this happens also relies on the internals of the EI-Jumps. Also, the EI-Jumps take care of repairing J.cnt and J.cmp so to ensure (a weaker) (PC-3). The weaker formulation of (PC-2), (PC-3) is discussed in SubSection 4.2. From this perspective, the functioning of J-VI$()$ and that of the EI-Jumps is quite braided. In order to detail these aspects, we need to observe the following fact.

Proposition 6. *Let $i\in[W^-,W^+]$ and $j\in[1,s-1]$. Assume that J-VI(i,j,F,J,Γ) is invoked at some step $\imath$, suppose that $J.L^{inc\,\imath}_{nxt}=\emptyset$, and that (PC-1), (PC-2), (PC-3) hold at step $\imath$.*

Then, the following two facts hold:

1. *At each step $\hat\imath\geq\imath$ of J-VI$()$, done* before *the swap$()$ at line 19: $J.L^{inc\,\hat\imath}\subseteq Inc(f^{c:\hat\imath},i,j)$.*
2. *When J-VI() halts,* after *the swap$()$ at line 19, say at step h, then:*

$$J.L^{inc\,h}=\{v\in V\mid 0<f^{c:h}(v)\neq\top\}.$$

Proof of (1) When J-VI() is invoked, Item 1 holds by (PC-2). Then, J-VI$()$ can insert any $u\in V$ into $L^{\rm inc}$ only at line 18, when exploring $N^{\rm in}_\Gamma(v)$ (from line 11 to line 18), for some $v\in V$. At line 18, $u\in V$ is inserted into $L^{\rm inc}$ *iff* $f_u<\Delta_{u,v}$ (line 14) and either $u\in V_1$ or $J.\texttt{cnt}[u]=0$; i.e., *iff* u is inconsistent w.r.t. $f^{\rm c}$ in $\Gamma_{i,j}$ (indeed, J.cnt is coherent by (PC-3) and the fact that lines 15-17 of J-VI() preserve coherency). As $f^{\mathrm{c}}(u)$ stands still while u is inside L^{inc}, and $f^{\mathrm{c}}(v)$ for any $v \in N_{\Gamma}^{\mathrm{out}}(u)$ can only increase during the J-VI(), then Item 1 holds. □

Proof of (2) Let us focus on the state of $L_{\mathrm{nxt}}^{\mathrm{inc}}$. Initially, $L_{\mathrm{nxt}}^{\mathrm{inc}} = \emptyset$ by hypothesis. During the J-VI(), $L_{\mathrm{nxt}}^{\mathrm{inc}}$ is modified only at line 6 or 10: some $v \in V$ is inserted into $L_{\mathrm{nxt}}^{\mathrm{inc}}$, say at step $\hat{\imath}$, (line 6) *iff* $f_v \neq \top$ (where f_v is the energy-level of v at the time of the insertion $\hat{\imath}$). We argue that $f_v > 0$ holds at $\hat{\imath}$ (line 6): since v was extracted from L^{inc} (line 2), and since all vertices in L^{inc} are inconsistent w.r.t. $f^{\mathrm{c}:\hat{\imath}}$ in $\Gamma_{i,j}$ by Item 1, then $\delta(\cdot, v)$ had really increased $f^{\mathrm{c}}(v)$ (at line 3); thus, it really holds $f_v > 0$ at $\hat{\imath}$. After the insertion, in case $f^{\mathrm{c}}(v)$ becomes $\top$ at some subsequent execution of line 3, v is removed from $L_{\mathrm{nxt}}^{\mathrm{inc}}$ (and inserted into $L_{\top}$), see lines 8-10. Finally, at line 19 of J-VI(), $L_{\mathrm{nxt}}^{\mathrm{inc}}$ and L^{inc} are swapped (line 19). Therefore, at that point, Item 2 holds. □

When J-VI() halts, it is necessary to initialize L^{inc} for the *next* J-VI() by including all the $v \in V$ such that $0 < f^{\mathrm{c}}(v) \neq \top$, because they are all inconsistent; this is shown by Lemma 1.

Lemma 1. *Let $i \in [W^-, W^+]$ and $j \in [1, s-1]$, where $s \triangleq |\mathscr{F}_{|V|}|$. Assume that J-VI() is invoked on input (i, j, F, J, Γ), and that all the pre-conditions (PC-1), (PC-2), (PC-3) are satisfied. Assume that J-VI(i, j, F, J, Γ) halts at step h. Let $i' \in [W^-, W^+]$ and $j' \in [1, s-1]$ be any two indices such that $(i', j') > (i, j)$. If $v \in V$ satisfies $0 < f^{\mathrm{c}:h}(v) \neq \top$, then $v \in Inc(f^{\mathrm{c}:h}, i', j')$.*

Proof. Let $\hat{v} \in V$ be any vertex such that $0 < f^{\mathrm{c}:h}(\hat{v}) \neq \top$. By Proposition 5, $\forall^{v \in V} f^{\mathrm{c}:h}(v) = f^*_{w'_{i,j}}(v)$. Since $f^*_{w'_{i,j}}(\hat{v})$ is the least-SEPM of $\Gamma_{i,j}$, then it is the unique least fixed-point of simultaneously all operators $\{\delta(\cdot, v)\}_{v \in V}$ by Knaster-Tarski; therefore, the following holds:

$$f^{\mathrm{c}:h}(\hat{v}) = f^*_{w'_{i,j}}(\hat{v}) = \begin{cases} \min\{f^*_{w'_{i,j}}(v') \ominus w'_{i,j}(\hat{v}, v') \mid v' \in N_{\Gamma}^{\mathrm{out}}(\hat{v})\}, & \text{if } \hat{v} \in V_0 \\ \max\{f^*_{w'_{i,j}}(v') \ominus w'_{i,j}(\hat{v}, v') \mid v' \in N_{\Gamma}^{\mathrm{out}}(\hat{v})\}, & \text{if } \hat{v} \in V_1 \end{cases}$$

Since $0 < f^{\mathrm{c}:h}(\hat{v}) \neq \top$, it is safe to discard the $\ominus$ operator in the equality above. Moreover, since $(i', j') > (i, j)$, then $w'_{i,j} > w'_{i',j'}$. Therefore, the following inequality holds:

$$\begin{aligned} f^{\mathrm{c}:h}(\hat{v}) &= \begin{cases} \min\{f^{\mathrm{c}:h}(v') - w'_{i,j}(\hat{v}, v') \mid v' \in N_{\Gamma}^{\mathrm{out}}(\hat{v})\}, & \text{if } \hat{v} \in V_0 \\ \max\{f^{\mathrm{c}:h}(v') - w'_{i,j}(\hat{v}, v') \mid v' \in N_{\Gamma}^{\mathrm{out}}(\hat{v})\}, & \text{if } \hat{v} \in V_1 \end{cases} \\ &< \begin{cases} \min\{f^{\mathrm{c}:h}(v') - w'_{i',j'}(\hat{v}, v') \mid v' \in N_{\Gamma}^{\mathrm{out}}(\hat{v})\}, & \text{if } \hat{v} \in V_0 \\ \max\{f^{\mathrm{c}:h}(v') - w'_{i',j'}(\hat{v}, v') \mid v' \in N_{\Gamma}^{\mathrm{out}}(\hat{v})\}, & \text{if } \hat{v} \in V_1 \end{cases} \end{aligned}$$

So, restoring $\ominus$, we have $f^{\mathrm{c}:h}(\hat{v}) \prec \begin{cases} f^{\mathrm{c}:h}(v') \ominus w'_{i',j'}(\hat{v}, v') \text{ for all } v' \in N_{\Gamma}^{\mathrm{out}}(\hat{v}), & \text{if } \hat{v} \in V_0 \\ f^{\mathrm{c}:h}(v') \ominus w'_{i',j'}(\hat{v}, v') \text{ for some } v' \in N_{\Gamma}^{\mathrm{out}}(\hat{v}), & \text{if } \hat{v} \in V_1 \end{cases}$

Therefore, $v \in \mathrm{Inc}(f^{\mathrm{c}:h}, i', j')$. □

Although, when the $\mathtt{prev}_{\rho^J}(i, j)$-th J-VI() halts, it is correct –and necessary– to initialize L^{inc} for the (i, j)-th J-VI() by including all those $v \in V$ such that $0 < f^{\mathrm{c}}(v) \neq \top$ (because they are all inconsistent w.r.t. to f^{c} in $\Gamma_{i,j}$ by Lemma 1), still, we observe that this is not sufficient. Consider the following two facts (I-1) and (I-2):

(I-1) It may be that, when the $\mathtt{prev}_{\rho^J}(i, j)$-th J-VI() halts, it holds for all $v \in V$ that either $f^{\mathrm{c}}(v) = 0$ or $f^{\mathrm{c}}(v) = \top$. In that case, L^{inc} would be empty (if nothing more than what prescribed by Proposition 6 is done). We need to prevent this from happening, so to avoid *vain* Scan-Phases.

(I-2) When going, say, from the $(i-1, j)$-th to the $(i,1)$-th Scan-Phase, there might be some $(u,v) \in E$ such that: $f^c(u) = 0 = f^c(v)$ and $w(u,v) = i$; those (u,v) may become incompatible w.r.t. f^c in $\Gamma_{i,1}$ (because $i-1$ had been increased to i), possibly breaking the compatibility (and thus the coherency) of (u,v). These incompatible arcs are not taken into account by Proposition 6, nor by Lemma 1. Thus a special care is needed in order to handle them.

Energy-Increasing-Jumps. To resolve the issues raised in I-1 and I-2, the *EI-Jumps* will come into play. The pseudocode of the EI-Jumps is provided in SubProcedure 6. The `ei-jump`(i,J) really makes a jump only when $L^{\text{inc}} = \emptyset$ holds invocation. Basically, if $L^{\text{inc}} = \emptyset$ (line 1) we aim at avoiding *vain* Scan-Phases, i.e., (I-1); still, we need to take care of some additional (possibly) incompatible arcs, i.e., (I-2). Recall, L^{inc} is initialized by the `J-VI`() itself according to Proposition 6. Therefore, at line 1, $L^{\text{inc}} = \emptyset$ *iff* either $J.f(v) = 0$ *or* $J.f(v) = \top$ for every $v \in V$.

SubProcedure 6: EI-Jump

```
    Procedure ei-jump(i, J)
        input  : Jumper J.
        output: T if an EI-Jump occurs; else, F.
1       if L^inc = ∅ then
2           L^inc ← L^inc_cpy; L^inc_cpy ← ∅ ;
3           J.i ← i + 1;
4           if L_ω ≠ ∅ then
5               (w, L_α) ← read_front(L_ω);
6               if w = J.i then
7                   pop_front(L_ω);
8                   repair(L_α, J);
9           while L^inc = ∅ and L_ω ≠ ∅ do
10              (w, L_α) ← pop_front(L_ω);
11              J.i ← w;
12              repair(L_α, J);
13          return T;
14      else return F;
    SubProcedure repair(L_α, J)
        input  : A list of arcs L_α, reference to Jumper J.
1       foreach (u, v) ∈ L_α do
2           if J.f[u] = 0 and J.f[v] = 0 and L^inc[u] = ⊥ then
3               if u ∈ V_0 then
4                   J.cnt[u] ← J.cnt[u] − 1;
5                   J.cmp[(u, v)] ← F;
6                   if J.cnt[u] = 0 then
7                       insert(u, L^inc);
8               if u ∈ V_1 then insert(u, L^inc);
```

To begin with, if $L^{\text{inc}} = \emptyset$ (line 1), copy $L^{\text{inc}} \leftarrow L^{\text{inc}}_{\text{cpy}}$, then, erase $L^{\text{inc}}_{\text{cpy}} \leftarrow \emptyset$ (line 2): this is related to the steps of backtracking that are performed by the UA-Jumps, we will give more details on this later on. Next, we increment i to $J.i \leftarrow i+1$ (line 3). Then, if $L_\omega \neq \emptyset$ at line 4, we read (read-only) the front entry $(\hat{w}, L_{\hat{\alpha}})$ of L_ω (line 5); only if $\hat{w} = J.i$ (line 6), we pop $(\hat{w}, L_{\hat{\alpha}})$ out of L_ω (line 7), and we invoke `repair`$(L_{\hat{\alpha}}, J)$ (line 8) to repair the coherency state of all those arcs (i.e., all and only those in $L_{\hat{\alpha}}$) that we mentioned in (I-2). We will detail `repair`() shortly, now let us proceed with `ei-jump`(). At line 9, *while* $L^{\text{inc}} = \emptyset$ *and* $L_\omega \neq \emptyset$: the front $(\bar{w}, L_{\bar{\alpha}})$ is popped from L_ω (line 10) and $J.i \leftarrow \bar{w}$ is assigned (line 11). The ending-point of the EI-Jump will now reach $\bar{w}$ (at least). A moment's reflection reveals that, jumping up to $\bar{w}$, some arcs $(u,v) \in E$ such that $f^c(u) = 0 = f^c(v)$ (which were compatible w.r.t. the (i,j)-th Scan-Phase, just *before* the jump) may become incompatible for the $(\bar{w},1)$-th Scan-Phase (which is now candidate to happen), because $\bar{w} > i$. What are these new incompatible arcs? Since L_ω was sorted in increasing order, they're all *and only* those of weight $w(u,v) = \bar{w} = J.i$; i.e., those in the $L_{\bar{\alpha}}$ that is binded to $\bar{w}$ in L_ω. To repair coherency, `repair`$(L_{\bar{\alpha}}, J)$ (line 12) is invoked. This repeats until $L^{\text{inc}} \neq \emptyset$ *or* $L_\omega = \emptyset$. Then, `ei-jump`() returns `T` (at line 13).

If $L^{\text{inc}} \neq \emptyset$ at line 1, then `F` is returned (line 14); so, in that case, *no* EI-Jump will occur.

Let us detail the `repair`(L_α, J). On input (L_α, J), for each arc $(u,v) \in L_\alpha$ (line 1), if $J.f[u] = 0 = J.f[v]$ and $L^{\text{inc}}[u] = \bot$ (line 2), the following happens. If $u \in V_1$, then u is promptly inserted (in front of) L^{inc} (line 8); else, if $u \in V_0$, $J.$`cnt`$[u]$ is decremented by one unit (line 4); also, it is flagged $J.$`cmp`$[(u,v)] \leftarrow$ `F` (line 5). After that, if $J.$`cnt`$[u] = 0$ (line 6), then u is inserted in front of L^{inc} (line 7). The following proposition holds for the `ei-jump`() (SubProcedure 6).

Proposition 7. *The* `ei-jump`() *(SubProcedure 6) halts in finite time. The* total *time spent for all* *invocations of* `ei-jump`() *(that are made, at line 7, during the main* `while` *loop of Algorithm 1) is* $\Theta(t_{\ell_7}+|E|)$*, where* t_{ℓ_7} *is the total number of iterations of line 7 that are made by Algorithm 1. The* `ei-jump`() *works with* $\Theta(|V|+|E|)$ *space.*

Proof. The `for-each` loop in `repair`() is bounded: each arc (u,v) of L_α is visited exactly once, spending $O(1)$ time per each. The `while` loop in `ei-jump`() (lines 9-12) is also bounded: it consumes the elements (w,L_α) of L_ω, spending $O(|L_\alpha|)$ time per cycle. There are no other loops in `ei-jump`(), so it halts in finite time. Now, consider the following three facts: (i) `ei-jump`() is invoked by `solve_MPG()` (Algorithm 1) once per each iteration of the main `while` loop at line 7. Assume there are t_{ℓ_7} such iterations overall. (ii) either `ei-jump`() returns immediately or it visits k arcs $(u,v)\in E$ in time $\Theta(k)$, for some $1\leq k\leq |E|$; (iii) each arc $(u,v)\in E$ is visited by `ei-jump`() at most once during the whole execution of Algorithm 1, because the elements of L_ω are consumed and there are no duplicates in there. Altogether, (i), (ii) and (iii) imply the $\Theta(t_{\ell_7}+|E|)$ total running time. Moreover, `ei-jump`() works with $\Theta(|V|+|E|)$ space. Indeed L^{inc} contains no duplicated vertices, so: $|L^{\text{inc}}|\leq |V|$, $|L_\omega|=|E|$, the size of $J.f$ and that of $J.$`cnt` is $|V|$, and the size of $J.$`cmp` is $|E|$. □

The description of Algorithm 1 ends by detailing the UA-Jumps.

Unitary-Advance-Jumps. Recall, UA-Jumps are adopted so to scroll through $\mathscr{F}_{|V|}$ only *when* (and *where*) it is really necessary; that is only $|V|$ times at most, because each time at least one vertex will take a value. The pseudocode is shown in Fig. 7.

The UA-Jumps begin soon after that `ei-jump()` returns `T` at line 8 of Algorithm 1. The starting point of the UA-Jumps (i.e., the initial value of *i*) is provided by `ei-jump()` (line 7 of Algorithm 1): it is stored into $J.i$ and passed in input to `ua-jumps`$(J.i,s,F,J,\Gamma)$ (at line 9 of Algorithm 1). Starting from $i=J.i$, basically the `ua-jumps`() repeats a sequence of invocations to `J-VI`(), on input $(i,s-1),(i+1,s-1),(i+2,s-1),\cdots,(\hat{i},s-1)$; until $L_\top\triangleq\mathscr{W}_0(\Gamma_{\hat{i}-1,s-1})\cap\mathscr{W}_1(\Gamma_{\hat{i},s-1})\neq\emptyset$ holds for some $\hat{i}\geq i$. When $L_\top\neq\emptyset$, the `ua-jumps`() *backtracks* the Scan-Phases from the $(\hat{i},s-1)$-th to the $(\hat{i},1)$-th one, by invoking `backtrack_ua-jump`(i,s,F,J,Γ), and then it halts; soon after, Algorithm 1 will begin scrolling through $\mathscr{F}_{|V|}$ by invoking another sequence of `J-VI`() (this time at line 11 of Algorithm 1) on input $(\hat{i},1),(\hat{i},2),(\hat{i},3),\ldots$ (which is controlled by the `while` loop at line 6 of Algorithm 1). More details concerning the UA-Jumps now follow.

So, `ua-jumps`() (SubProcedure 7) performs a sequence of UA-Jumps (actually, at least one). The invocation to `J-VI`$(\hat{i},s-1,F,J,\Gamma)$ repeats for $\hat{i}\geq i$ (lines 1-2), until $L_\top\neq\emptyset$ (line 6). There, $L_\top$ contains all and only those $v\in V$ whose energy-level became $f(v)=\top$ during the last performed `J-VI`() (line 2); so, at line 3, it is $L_\top=\mathscr{W}_0(\Gamma_{\hat{i}-1,s-1})\cap\mathscr{W}_1(\Gamma_{\hat{i},s-1})$. At this point, if $L_\top=\emptyset$ (line 3), the procedure prepares itself to make another UA-Jump: $\hat{i}\leftarrow\hat{i}+1$ is set (line 4), and then `rejoin_ua-jump`$(\hat{i},s,F,J)$ is invoked (line 5). Else, if $L_\top\neq\emptyset$ (line 6), it is invoked `backtrack_ua-jump`$(\hat{i},s,F,J,\Gamma)$ (line 7), and then (i,S) is returned (line 8), where $S\triangleq L_\top=\mathscr{W}_0(\Gamma_{\hat{i}-1,s-1})\cap\mathscr{W}_1(\Gamma_{\hat{i},s-1})$ was assigned at line 4 of `backtrack_ua-jump`().

The `rejoin_ua-jump`(i,s,F,J) firstly copies the energy-levels stored in L_f back to $J.f$, by invoking `scl_back_`$f(s-1,F,J)$ (line 1). Secondly, at lines 4-6, by operating in the same way as `ei-jump`() does (see lines 4-8 of `ei-jump`(), SubProcedure 6), it repairs the coherency state of $J.$`cnt` and $J.$`cmp` w.r.t. all those arcs $(u,v)\in E$ such that $w(u,v)=i$ and $J.f[u]=0=J.f[v]$.

Let us detail the `backtrack_ua-jump`(). Basically, it aims at preparing a correct state so to allow Algorithm 1 to step through $\mathscr{F}_{|V|}$. Stepping through $\mathscr{F}_{|V|}$ essentially means to execute a sequence of `J-VI`() at line 11 of Algorithm 1, until $L^{\text{inc}}=\emptyset$. A moment's reflection reveals that this sequence of J-VI() can run just on the sub-arena of Γ that is induced by $S \triangleq L_\top = \mathscr{W}_0(\Gamma_{i-1,s-1}) \cap \mathscr{W}_1(\Gamma_{i,s-1})$ (see line 4 of backtrack_ua-jump()); there is no real need to lift-up again (actually, slowly than before) all the energy-levels of the component induced by $V \setminus L_\top$: those energy-levels can all be confirmed now that the UA-Jumps are finishing, and they can all stand still while Algorithm 1 is stepping through $\mathscr{F}_{|V|}$ at line 11, until another EI-Jump occurs.

For this reason, backtrack_ua-jump$(\hat{\imath}, s, F, J, \Gamma)$ works as follows.

SubProcedure 7: UA-Jumps

```
SubProcedure ua-jumps(i, s, F, J, Γ)
    input : i ∈ [W⁻, W⁺], s = |𝓕_{|V|}|, F is a ref. to
            𝓕_{|V|}, Jump J, input MPG Γ.
1   repeat
2       J-VI(i, s−1, F, J, Γ); /* UA-Jump */
3       if L_⊤ = ∅ then
4           i ← i + 1;
5           rejoin_ua-jump(i, s, F, J);
6   until L_⊤ ≠ ∅
7   S ← backtrack_ua-jump(i, s, F, J, Γ);
8   return (i, S);

SubProcedure rejoin_ua-jump(i, s, F, J)
    input : i ∈ [W⁻, W⁺], F is a ref. to 𝓕_{|V|}, Jump J.
1   scl_back_f(s−1, F, J);
2   if L_ω ≠ ∅ then
3       (w, L_α) ← read_front(L_ω);
4       if w = i then
5           pop_front(L_ω);
6           repair(L_α, J); // see SubProc. 6

SubProcedure backtrack_ua-jump(i, s, F, J, Γ)
    input : i ∈ [W⁻, W⁺], s = |𝓕_{|V|}|, Jump J, MPG Γ.
1   L^inc_cpy ← L^inc; L^inc ← ∅;
2   L_f[u] ← { ⊥      , if u ∈ L_⊤;
              { L_f[u] , if u ∈ V \ L_⊤.
3   scl_back_f(s−1, F, J);
4   S ← L_⊤;
5   while L_⊤ ≠ ∅ do
6       u ← pop_front(L_⊤)
7       if u ∈ V_0 then
8           init_cnt_cmp(u, i, 1, F, J, Γ[S]);
9           if J.cnt[u] = 0 then
10              insert(u, L^inc);
11      if u ∈ V_1 then
12          foreach v ∈ N^out_{Γ[S]}(u) do
13              f_u ← get_scl_f(u, 1, F, J);
14              f_v ← get_scl_f(v, 1, F, J);
15              w' ← get_scl_w(w(u, v), i, 1, F);
16              if f_u ≺ f_v ⊖ w' then
17                  insert(u, L^inc); break;
18  return S;
```

Firstly, we copy $L^{\text{inc}}_{\text{cpy}} \leftarrow L^{\text{inc}}$, then we erase $L^{\text{inc}} \leftarrow \emptyset$ (line 1). This is sort of a back-up copy, notice that $L^{\text{inc}}_{\text{cpy}}$ will be restored back to L^{inc} at line 2 of ei-jump() (SubProcedure 6): when Algorithm 1 will finish to step through $\mathscr{F}_{|V|}$, it will hold $L^{\text{inc}} = \emptyset$ at line 1 of ei-jump() (SubProcedure 6), so at that point the state of L^{inc} will need to be restored by including (at least) all those vertices that are now assigned to $L^{\text{inc}}_{\text{cpy}}$ at line 1 of backtrack_ua-jump(). Next, all the energy-levels of $V \setminus L_\top$ are confirmed and saved back to $J.f$; this is done: (i) by setting,

$$L_f[u] \leftarrow \begin{cases} \bot & \text{, if } u \in L_\top; \\ L_f[u] & \text{, if } u \in V \setminus L_\top. \end{cases} \quad \text{(line 2)}$$

and (ii) by invoking scl_back_$f(s-1, F, J)$ (line 3). The energy-levels of all $v \in L_\top$ are thus restored as they were at the end of the $(\hat{\imath}-1, s-1)$-th invocation of J-VI() at line 2 of ua-jumps(). Next, it is assigned $S \leftarrow L_\top$ at line 4. Then, backtrack_ua-jump() takes care of preparing a correct state of L^{inc}, $J.$cnt, $J.$cmp for letting Algorithm 1 stepping through $\mathscr{F}_{|V|}$.

While $L_\top \neq \emptyset$ (line 5), we pop the front element of $L_\top$, i.e., $u \leftarrow$ pop_front$(L_\top)$ (line 6):

– If $u \in V_0$ (line 7), then we compute $J.$cnt$[u]$ and we also compute for every $v \in N^{\text{out}}_{\Gamma[S]}(u)$ a coherent $J.$cmp$[(u,v)]$ w.r.t. f^{c} in $\Gamma[S]_{\hat{\imath},1}$, by init_cnt_cmp$(u, \hat{\imath}, 1, F, J, \Gamma[S])$ (line 8); finally, if $J.$cnt$[u] = 0$ (line 9), we insert u into L^{inc} (line 10).

– Else, if $u \in V_1$ (line 11), we explore $N^{\text{out}}_{\Gamma[S]}(u)$ looking for some incompatible arc (lines 12-17). For each $v \in N^{\text{out}}_{\Gamma[S]}(u)$ (line 12), if $f_u \prec f_v \ominus w'_{\hat{\imath},1}(u,v)$ (i.e., if (u,v) is incompatible w.r.t. f^{c} in $\Gamma_{\hat{\imath},1}$), where $f_u \leftarrow$ get_scl_$f(u, 1, F, J)$ and $f_v \leftarrow$ get_scl_$f(v, 1, F, J)$, then, we insert u into L^{inc} at line 17 (also breaking the for-each cycle).

This concludes the description of the UA-Jumps. Algorithm 1 is completed.

4.2. Correctness of Algorithm 1

This subsection presents the proof of correctness for Algorithm 1. It is organized as follows. Firstly, we show that `J-VI()` (SubProcedure 4) works fine even when assuming a relaxed form of the pre-conditions (PC-2) and (PC-3). Secondly, we identify an additional set of pre-conditions under which the `ei-jump()` (SubProcedure 6) is correct. Thirdly, we prove that under these pre-conditions `ua-jumps()` (SubProcedure 7) is also correct. Finally, we show that these pre-conditions are all satisfied during the execution of Algorithm 1, and that the latter is thus correct.

Correctness of `J-VI()` (SubProcedure 4)

To prove the correctness of `J-VI()`, the (PC-1), (PC-2), (PC-3) have been assumed in Lemma 1. It would be fine if they were met whenever Algoritm 1 invokes `J-VI()`. Unfortunately, (PC-2) and (PC-3) may not hold. Still, we shall observe that a weaker formulation of them, denoted by (w-PC-2) and (w-PC-3), really hold; and these will turn out to be enough for proving correctness.

Definition 4. *Let $i \in [W^-, W^+]$ and $j \in [1, s-1]$. Fix some step of execution ι of Algorithm 1. The pre-conditions (w-PC-2) and (w-PC-3) are defined at step ι as follows.*

(w-PC-2) $L^{inc^\iota} \subseteq Inc(f^{c:\iota}, i, j)$.

(w-PC-3) $\forall (u \in V \setminus L^{inc^\iota})\ \forall (v \in N^{out}_\Gamma(u))$*:*

If $u \in V_0$, the following three properties hold on $J.\texttt{cnt}^\iota$ and $J.\texttt{cmp}^\iota$:

1. *If $J.\texttt{cmp}^\iota[(u,v)] = F$, then (u,v) is incompatible w.r.t. f^c in $\Gamma_{i,j}$;*
2. *If $J.\texttt{cmp}^\iota[(u,v)] = T$ and (u,v) is incompatible w.r.t. f^c in $\Gamma_{i,j}$, then $v \in L^{inc^\iota}$.*
3. $J.\texttt{cnt}^\iota[u] = \left|\left\{v \in N^{out}_\Gamma(u) \mid J.\texttt{cmp}^\iota[(u,v)] = T\right\}\right|$ *and* $J.\texttt{cnt}^\iota[u] > 0$.

If $u \in V_1$, and $(u,v) \in E$ is incompatible w.r.t. f^c in $\Gamma_{i,j}$, then $v \in L^{inc^\iota}$.

If (w-PC-3) holds on $J.\texttt{cnt}^\iota$ and $J.\texttt{cmp}^\iota$, they are said weak-coherent *w.r.t. f^c in $\Gamma_{i,j}$.*

We will also need the following Lemma 2, it asserts that $\psi_\rho : (i,j) \to f^*_{i,j}$ is monotone non-decreasing; the proof already appears in [Comin and Rizzi (2016), Lemma 8, Item 1].

Lemma 2. *Let $i, i' \in [W^-, W^+]$ and $j, j' \in [1, s-1]$ be any two indices such that $(i,j) < (i',j')$. Then, $\forall^{v\in V} f^*_{i,j}(v) \preceq f^*_{i',j'}(v)$.*

Proposition 8 shows that (PC-1), (w-PC-2), (w-PC-3) suffices for the correctness of `J-VI()`.

Proposition 8. *The `J-VI()` (SubProcedure 4) is correct (i.e., Propositions 5 and 6 still hold) even if (PC-1), (w-PC-2), (w-PC-3) are assumed instead of (PC-1), (PC-2), (PC-3).*

In particular, suppose that `J-VI()` is invoked on input (i,j,F,J,Γ), say at step ι, and that all of the pre-conditions (PC-1), (w-PC-2), (w-PC-3) hold at ι. When `J-VI`(i,j,F,J,Γ) halts, say at step h, then all of the following four propositions hold:

1. *$f^{c:h}$ is the least-SEPM of the EG $\Gamma_{i,j}$;*
2. *$J.\texttt{cnt}^h$, $J.\texttt{cmp}^h$ are both coherent w.r.t. $f^{c:h}$ in $\Gamma_{i,j}$;*
3. *$L^{inc^h} = \{v \in V \mid 0 < f^{c:h}(v) \neq \top\}$;*
4. *$L^h_\top = V_{f^{c:\iota}} \cap V \setminus V_{f^{c:h}}$.*

Proof. Basically, we want to prove that Propositions 5 and 6 still hold.

Suppose ${L^{\rm inc}}^{\iota} = \emptyset$. Let $u \in V_0$. By (w-PC-3) and ${L^{\rm inc}}^{\iota} = \emptyset$, for every $v \in N_\Gamma^{\rm out}(u)$, $J.\mathtt{cmp}^{\iota}[(u,v)]$ is coherent w.r.t. f^{ι} in $\Gamma_{i,j}$; thus, $J.\mathtt{cnt}^{\iota}[u]$ is also coherent w.r.t. $f^{c:\iota}$ in $\Gamma_{i,j}$. Therefore, (PC-3) holds. Now, let $u \in V_1$. By (w-PC-3) and ${L^{\rm inc}}^{\iota} = \emptyset$, for every $v \in N_\Gamma^{\rm out}(u)$ it holds that (u,v) is compatible w.r.t. $f^{c:\iota}$ in $\Gamma_{i,j}$; thus, u is consistent w.r.t. $f^{c:\iota}$ in $\Gamma_{i,j}$. In addition, by (w-PC-3) again, $J.\mathtt{cnt}^{\iota}[u] > 0$ holds for every $u \in V_0$. Therefore, every $u \in V$ is consistent w.r.t. $f^{c:\iota}$ in $\Gamma_{i,j}$; so, (PC-2) holds. Since (PC-1,2,3) hold, then Propositions 5 and 6 hold.

Now, suppose ${L^{\rm inc}}^{\iota} \neq \emptyset$. Since $J.\mathtt{cnt}^{\iota}$ and $J.\mathtt{cmp}^{\iota}$ may be incoherent – *at time* ι –, there might be some $\hat{u} \in V \setminus {L^{\rm inc}}^{\iota}$ which is already inconsistent w.r.t. $f^{c:\iota}$ in $\Gamma_{i,j}$ (i.e., even if $u \notin {L^{\rm inc}}^{\iota}$).

Still, we claim that, during J-VI()'s execution (say at some steps ι', ι'', i.e., eventually), for every $u \in V_0$ and $v \in N_\Gamma^{\rm out}(u)$, both $J.\mathtt{cmp}^{\iota'}[(u,v)]$ and $J.\mathtt{cnt}^{\iota''}[u]$ will *become* coherent (at ι', ι'' respectively); and we also claim that any $u \in V_1$ which was inconsistent at ι will be (eventually, say at step ι''') inserted into $L^{\rm inc}$. Indeed, at that point (say, at $\hat{\iota} = \max\{\iota', \iota'', \iota'''\}$), *all* (and only those) $\hat{u} \in V$ that were already inconsistent at invocation time ι, or that became inconsistent during J-VI()'s execution (until step $\hat{\iota}$), they will be really inserted into $L^{\rm inc}$.

To prove it, let $\hat{u} \in V \setminus {L^{\rm inc}}^{\iota}$ and $\hat{v} \in N_\Gamma^{\rm out}(\hat{u})$ be any two (fixed) vertices such that either:

$\hat{u} \in V_0$ and $J.\mathtt{cmp}^{\iota}[(\hat{u},\hat{v})] = \mathtt{F}$: Then, by (w-PC-3), $(\hat{u},\hat{v})$ is incompatible w.r.t. $f^{c:\iota}$ in $\Gamma_{i,j}$.

$\hat{u} \in V_0$ and $J.\mathtt{cmp}^{\iota}[(\hat{u},\hat{v})] = \mathtt{T}$ but $(\hat{u},\hat{v})$ is incompatible w.r.t. $f^{c:\iota}$ in $\Gamma_{i,j}$:

Then, by (w-PC-3), $\hat{v} \in {L^{\rm inc}}^{\iota}$. Since J-VI() aims precisely at emptying $L^{\rm inc}$, $\hat{v}$ is popped from ${L^{\rm inc}}^{\iota'}$ (line 2 of SubProcedure 4) – say at some step ι' of J-VI()'s execution. Soon after that, $N_\Gamma^{\rm in}(\hat{v})$ is explored (lines 11-18 of SubProcedure 4); so $\hat{u}$ is visited, then $(\hat{u},\hat{v})$ is found incompatible (i.e., $f_{\hat{u}} < \Delta_{\hat{u},\hat{v}}$ at line 14, after ι'). Since $\hat{u} \in V_0 \setminus {L^{\rm inc}}^{\iota'}$, and $J.\mathtt{cmp}^{\iota'}[(\hat{u},\hat{v})] = \mathtt{T}$, then at some step $\iota'' > \iota'$ the counter $J.\mathtt{cnt}^{\iota''}$ is decremented by one unit and therefore $J.\mathtt{cmp}^{\iota''}[(u,v)] \leftarrow \mathtt{F}$ is assigned (at lines 16-17). This proves that $J.\mathtt{cmp}[(\hat{u},\hat{v})]$ becomes coherent eventually (i.e., at ι''). Now, given $\hat{u}$, the same argument holds for any other $v \in N_\Gamma^{\rm out}(\hat{u})$; therefore, when $J.\mathtt{cmp}[(\hat{u},v)]$ will finally become coherent for every $v \in N_\Gamma^{\rm out}(\hat{u})$, then $J.\mathtt{cnt}[\hat{u}]$ will be coherent as well by (w-PC-3). Thus, by (w-PC-3), coherency of both $J.\mathtt{cnt}$ and $J.\mathtt{cmp}$ holds eventually, say at $\hat{\iota}$. At that point, all $u \in V_0$ that were inconsistent at ι, or that have become inconsistent during the execution (up to $\hat{\iota}$), they necessarily have had to be inserted into $L^{\rm inc}$ (at line 18 of J-VI(), SubProcedure 4), because their (coherent) counter $J.\mathtt{cnt}[u]$ must reach 0 (at $\hat{\iota}$), which allows J-VI() to recognize u as inconsistent at lines 14-18. Notice that the coherency of $J.\mathtt{cnt}$ and $J.\mathtt{cmp}$ is kept satisfied from $\hat{\iota}$ onwards: when some $v \in V$ is popped out of $L^{\rm inc}$ (line 2), then $J.\mathtt{cnt}$ and $J.\mathtt{cmp}$ are recalculated from scratch (line 7), and it is easy to check that init_cnt_cmp() (SubProcedure 5) is correct; then $J.\mathtt{cnt}$, $J.\mathtt{cmp}$ may be modified subsequently, at lines 16-17 (SubProcedure 4), but it's easy to check that lines 14-17 preserve coherency; so, coherency will be preserved until J-VI() halts.

$\hat{u} \in V_1$ and $(\hat{u},\hat{v})$ is incompatible w.r.t. $f^{c:\iota}$ in $\Gamma_{i,j}$:

Then, by (w-PC-3), $\hat{v} \in {L^{\rm inc}}^{\iota}$. As before, since the J-VI() aims precisely at emptying $L^{\rm inc}$, $\hat{v}$ is popped from $L^{\rm inc}$ (line 2 of SubProcedure 4); at some step of J-VI()'s execution. Soon after that, $N_\Gamma^{\rm in}(\hat{v})$ is explored (lines 11-18 of SubProcedure 4). As soon as $\hat{u}$ is visited, $(\hat{u},\hat{v})$ is found incompatible (i.e., $f_{\hat{u}} < \Delta_{\hat{u},\hat{v}}$ at line 14). Since $\hat{u} \in V_1 \setminus L^{\rm inc}$, then $\hat{u}$ is promptly inserted into $L^{\rm inc}$ (line 18). In this way, all those $u \in V_1$ that were inconsistent at the time of J-VI()'s invocation, or that become inconsistent during the execution, they necessarily have had to be inserted into L^{inc} (line 18 of SubProcedure 4).

This analysis is already sufficient for asserting that Proposition 5 holds, even assuming only (PC-1), (w-PC-3): indeed, the Inv-JVI invariant mentioned in its proof will hold, eventually, and then the Knaster-Tarski's Fixed-Point Theorem applies. This also proves Items (1) and (2).

Moreover, by (w-PC-2) and by arguments above, at each step $\bar{\iota}$ of J-VI(), if $v \in L^{\text{inc}^{\bar{\iota}}}$ then v is really inconsistent w.r.t. $f^{\text{c}:\bar{\iota}}$ in $\Gamma_{i,j}$, i.e., $L^{\text{inc}^{\bar{\iota}}} \subseteq \text{Inc}(f^{\text{c}:\bar{\iota}}, i, j)$. Thus, every time that some v is popped from L^{inc} at line 2, then $\delta(f^{\text{c}}, v)$ really increases $f^{\text{c}}(v)$ at line 3; therefore, $f^{\text{c}}(v) > 0$ holds whenever v is inserted into $L^{\text{inc}}_{\text{nxt}}$ at line 6 of J-VI() (SubProcedure 4); this implies that Proposition 6 holds, assuming (PC-1), (w-PC-2), (w-PC-3), and proves Item (3). To conclude, we show Item (4). Notice, $L_\top$ is modified only at line 9 of J-VI() (SubProcedure 4); in particular, some $v \in V$ is inserted into $L_\top$ at line 9, say at step $\hat{\iota}$, if and only if $f^{\text{c}:\hat{\iota}}(v) = \top$. Since the energy-levels can only increase during the execution of J-VI(), then $L^h_\top \subseteq V \setminus V_{f^{\text{c}:h}} = \{u \in V \mid f^{\text{c}:h}(u) = \top\}$. Since at each step $\bar{\iota}$ of J-VI() it holds $L^{\text{inc}^{\bar{\iota}}} \subseteq \text{Inc}(f^{\text{c}:\bar{\iota}}, i, j)$, then whenever some $v \in V$ is inserted into $L_\top$ at line 9, it must be that $f^{\text{c}:\iota}(v) < \top$ where ι is the invocation time (otherwise, v would not have been inconsistent at step $\bar{\iota}$); thus, $L^h_\top \subseteq V_{f^{\text{c}:\iota}} = \{v \in V \mid f^{\text{c}:\iota}(v) < \top\}$. Therefore, $L^h_\top \subseteq V_{f^{\text{c}:\iota}} \cap V \setminus V_{f^{\text{c}:h}}$. Vice versa, let $v \in V_{f^{\text{c}:\iota}} \cap V \setminus V_{f^{\text{c}:h}}$; the only way in which J-VI() can increase the energy-level of v from step ι to step h is by applying $\delta(f^{\text{c}}, v)$ at line 3; as soon as $f^{\text{c}}(v) = \top$ (and this will happen, eventually, since $v \in V_{f^{\text{c}:\iota}} \cap V \setminus V_{f^{\text{c}:h}}$), then v is inserted into $L_\top$ at line 9. Thus, $V_{f^{\text{c}:\iota}} \cap V \setminus V_{f^{\text{c}:h}} \subseteq L^h_\top$. Therefore, $L^h_\top = V_{f^{\text{c}:\iota}} \cap V \setminus V_{f^{\text{c}:h}}$; and this proves Item (4). □

Correctness of EI-Jump (SubProcedure 6)

To begin, it is worth asserting some preliminary properties of ei-jump() (SubProcedure 6).

Lemma 3. *Assume* ei-jump(i,J) *(SubProcedure 6) is invoked by Algorithm 1 at line 7, say at step ι, and for some $i \in [W^- - 1, W^+]$ (i.e., for $i = i^\iota$). Assume $L^{inc^\iota} = \emptyset$ and $L^\iota_\omega \neq \emptyset$; and say that* ei-jump$(i,J)$ *halts at step h. Then, the following two properties hold.*

1. *The front element $(\bar{w}, L_\alpha)$ of L^ι_ω satisfies $\bar{w} = \min\{w_e \mid e \in E, w_e > i\}$;*
2. *It holds that $J.i^h \geq \bar{w} > i$.*

Proof. At the first invocation of ei-jump(i,J) (SubProcedure 6), made at line 7 of Algorithm 1, it holds $i = W^- - 1$ (by line 5 of Algorithm 1). Since $L^{\text{inc}^\iota} = \emptyset$, then ei-jump() first assigns $J.i \leftarrow i+1 = W^-$ at line 3. Since $L_{\mathbf{w}}$ was sorted in increasing order at line 12 of init_jumper() (SubProcedure 1), the front entry of $L_{\mathbf{w}}$ has key $w = W^-$, and all of the subsequent entries of $L_{\mathbf{w}}$ are binded to greater keys. Actually, ei-jump() consumes the front entry (W^-, L_α) of $L_{\mathbf{w}}$ at line 7; and W^- is assigned to $J.i$ (line 3). These observations imply both Item 1 and Item 2. Now, consider any invocation of ei-jump(i,J) (SubProcedure 6) which is not the first, but any subsequent one. Let us check that the front element $(\bar{w}, L_\alpha)$ of $L^\iota_{\mathbf{w}}$ satisfies $\bar{w} = \min\{w_e \mid e \in E, w_e > i^\iota\}$. Consider each line of Algorithm 1 at which the value i^ι could have ever been assigned to i; this may happen only as follows:

– At line 3 of ei-jump() (SubProcedure 6), i.e., $J.i \leftarrow i+1$ $(= i^\iota)$. But then the front element $(\hat{w}, L_\alpha)$ of $L_{\mathbf{w}}$ is also checked at lines 5-6 (because $L_{\mathbf{w}} \neq \emptyset$): and $\hat{w}$ is popped from $L_{\mathbf{w}}$ at line 7, in case $\hat{w} = J.i$ $(= i^\iota)$ holds at line 6.

– The same happens at lines 2-5 of rejoin_ua-jump() (SubProcedure 7); just notice that in that case i was incremented just before at line 4 of ua-jumps() (SubProcedure 7).

– At lines 9-10 of `ei-jump`() (SubProcedure 6), whenever the front element $(\hat{w}, L_\alpha)$ of $L_{\mathbf{w}}$ is popped, then $J.i \leftarrow \hat{w}$ is assigned.

Therefore, in any case, the following holds:

When the variable i got any of its possible values, say $\hat{\imath}$ (including i^ι), the front entry $(\hat{w}, L_\alpha)$ of $L_{\mathbf{w}}$ had always been checked, and then popped from $L_{\mathbf{w}}$ if $\hat{w} = \hat{\imath}$.

Recall, $L_{\mathbf{w}}$ was sorted in increasing order at line 12 of `init_jumper`() (SubProcedure 1).

Therefore, when `ei-jump`(i^ι, J) is invoked at step ι, all of the entries (w, L_α) of $L_{\mathbf{w}}$ such that $w \leq i^\iota$ must already have been popped from $L_{\mathbf{w}}$ before step ι.

Therefore, $\bar{w} = \min\{w_e \mid e \in E, w_e > i^\iota\}$, if $\bar{w}$ is the key (weight) of the front entry of $L^\iota_{\mathbf{w}}$.

Next, since $L^{\text{inc}^\iota} = \emptyset$ and $L^\iota_{\mathbf{w}} \neq \emptyset$ by hypothesis, and by line 9 of `ei-jump`(), at least one further element (w, L_α) of $L_{\mathbf{w}}$ must be popped from $L^\iota_{\mathbf{w}}$, either at line 7 or line 10 of `ei-jump`(), soon after ι. Consider the last element, say w', which is popped after ι and before h. Then, $J.i^h \leftarrow w'$ is assigned either at line 3 or line 11 of `ei-jump`(). Notice, $w' \geq \bar{w} > i^\iota$. Thus, $J.i^h \geq \bar{w} > i^\iota$. □

The following proposition essentially asserts that `ei-jump`() (SubProcedure 6) is correct. To begin, notice that, when `ei-jump`(i, J) is invoked at line 7 of Algorithm 1, then $i \in [W^- - 1, W^+]$. Also recall that any invocation of `ei-jump`(i, J) halts in finite time by Proposition 7.

Proposition 9. *Consider any invocation of `ei-jump`(i, J) (SubProcedure 6) that is made at line 7 of Algorithm 1, say at step ι, and for some $i \in [W^- - 1, W^+]$. Further assume that $L^{inc^\iota} = \emptyset$ and that `ei-jump`() halts at step h.*

Suppose the following pre-conditions are all satisfied at invocation time ι, for $s = |\mathscr{F}_{|V|}|$:

(eij-PC-1) $f^{c:\iota}$ *is the least-SEPM of* $\Gamma_{i,s-1}$*; thus,* $Inc(f^{c:\iota}, i, s-1) = \emptyset$. *Also,* $L^\iota_f = \emptyset$.
(eij-PC-2) $\{v \in V \mid 0 < f^{c:\iota}(v) \neq \top\} = \emptyset$;
(eij-PC-3) $L^{inc^\iota}_{cpy} \subseteq Inc(f^{c:\iota}, i', j')$ *for every* $(i', j') > (i, s-1)$;
(eij-PC-4) $J.$`cnt`$^\iota$ *and* $J.$`cmp`$^\iota$ *are both coherent w.r.t.* $f^{c:\iota}$ *in* $\Gamma_{i,s-1}$.

Finally, let $i' \in [W^-, W^+]$, $j' \in [1, s-1]$ be any indices such that $(i, s-1) < (i', j') \leq (J.i^h, 1)$. Then, the following holds.

1. *Suppose that $L^\iota_{\mathbf{w}} \neq \emptyset$. Let $(\hat{w}, L_{\hat{\alpha}})$ be any entry of $L^\iota_{\mathbf{w}}$ such that $\hat{w} = J.i^{\iota'} = i'$ holds either at line 6 or line 11 of `ei-jump`(i, J), for some step $\iota' > \iota$. When the `repair`$(L_{\hat{\alpha}}, J)$ halts soon after, either at line 8 or 12 (respectively), say at some step $\iota'' > \iota'$, both $J.$`cnt`$^{\iota''}$ and $J.$`cmp`$^{\iota''}$ are coherent w.r.t. $f^{c:\iota''}$ ($= f^{c:\iota}$) in $\Gamma_{i',j'}$.*
2. *If $(i', j') < (J.i^h, 1)$, then $Inc(f^{c:\iota}, i', j') = \emptyset$;*
3. *It holds that either $L^{inc^h} \neq \emptyset$ or both $L^{inc^h} = \emptyset$ and $L^h_\omega = \emptyset$.*
   Anyway, $L^{inc^h} = Inc(f^{c:h}, i^h, 1)$.

Notice that f^{c} stands still during `ei-jump`() (SubProcedure 6), i.e., $f^{\text{c}:\iota} = f^{\text{c}:\iota'} = f^{\text{c}:\iota''} = f^{\text{c}:h}$, for steps $\iota, \iota', \iota'', h$ defined as in Proposition 9. In the proofs below, we can simply refer to f^{c}.

Proof of Item (1). Let $u \in V_0$ and $v \in N^{\text{out}}_\Gamma(u)$, let i', j' be fixed indices such that $(i, s-1) < (i', j') \leq (J.i^h, 1)$. By (*eij-PC-2*), either $f^{\text{c}}(u) = \top$ or $f^{\text{c}}(u) = 0$, either $f^{\text{c}}(v) = \top$ or $f^{\text{c}}(v) = 0$.

- If $f^{\text{c}}(u) = \top$, then $(u, v) \in E$ is compatible w.r.t. f^{c} in $\Gamma_{i,s-1}$. So, $J.$`cmp`$^\iota[(u, v)] =$ `T` holds by (*eij-PC-4*). Since $f^{\text{c}}(u) = \top$, `ei-jump`() can't modify $J.$`cmp`$[(u, v)]$; see line 2 of `repair`() (SubProcedure 6). So, $J.$`cmp`$^{\iota''}[(u, v)] =$ `T` is still coherent w.r.t. f^{c} in $\Gamma_{i',j'}$.

- If $f^c(u) = 0$ and $f^c(v) = \top$, then $(u,v) \in E$ is incompatible w.r.t. f^c in $\Gamma_{i,s-1}$. So, $J.\texttt{cmp}^{\iota}[(u,v)] = \texttt{F}$ holds by *(eij-PC-4)*; and it will hold for the whole execution of `ei-jump()`, because `ei-jump()` never changes $J.\texttt{cmp}$ from F to T. Thus, $J.\texttt{cmp}^{\iota''}[(u,v)] = \texttt{F}$ is still coherent w.r.t. f^c in $\Gamma_{i',j'}$.
- Assume $f^c(u) = 0$ and $f^c(v) = 0$.

  Again, $J.\texttt{cmp}^{\iota}[(u,v)]$ is coherent w.r.t. f^c in $\Gamma_{i,s-1}$ by *(eij-PC-4)*. We have two cases:

  - If $J.\texttt{cmp}^{\iota}[(u,v)] = \texttt{F}$, then (u,v) is incompatible w.r.t. f^c in $\Gamma_{i,s-1}$, i.e., $f^c(u) < f^c(v) - (w(u,v) - i - F_{s-1})$. Since $f^c(u) = f^c(v) = 0$ and $F_{s-1} = 1$, then:

    $$0 = f^c(u) < f^c(v) - w(u,v) + i + F_{s-1} = -w(u,v) + i + 1.$$

    Therefore, $w(u,v) \le i$, because $w(u,v) \in \mathbb{Z}$. Since $(i, s-1) < (i', j')$, then $i < i'$, so $w(u,v) < i'$; this means that (u,v) is still incompatible w.r.t. f^c in $\Gamma_{i',j'}$. Meanwhile, $J.\texttt{cmp}[(u,v)] = \texttt{F}$ stands still (because `ei-jump()` never changes $J.\texttt{cmp}$ from F to T), therefore, $J.\texttt{cmp}^{\iota''}[(u,v)] = \texttt{F}$.
  - If $J.\texttt{cmp}^{\iota}[(u,v)] = \texttt{T}$, then (u,v) is compatible w.r.t. f^c in $\Gamma_{i,s-1}$ by *(eij-PC-4)*. So,
    * If (u,v) is compatible w.r.t. f^c in $\Gamma_{i',j'}$, i.e., $f^c(u) \ge f^c(v) - (w(u,v) - i' - F_{j'})$, then, since $f^c(u) = f^c(v) = 0$, we have:

      $$0 = f^c(u) \ge f^c(v) - w(u,v) + i' + F_{j'} = -w(u,v) + i' + F_{j'}.$$

      Then, $w(u,v) > i'$, because $j' \in [1, s-1]$ (so, $F_{j'} > 0$) and $w(u,v) \in \mathbb{Z}$. Consider what happens in `ei-jump()` at ι'. Since $L_{\mathbf{w}}$ was sorted in increasing order, and since $w(u,v) > i'$, then the entry $(w(u,v), L_\alpha)$ is still inside $L_{\mathbf{w}}$ at ι' (indeed, at step ι', the front entry of $L_{\mathbf{w}}$ has key value i' by hypothesis). Therefore, neither the subsequent invocation of `repair()` (line 8 or line 12 of `ei-jump()`), nor any of the previous invocations of `repair()` (before ι'), can alter the state of $J.\texttt{cmp}([u,v])$ from T to F, just because $(u,v) \in L_\alpha$ is still inside $L_{\mathbf{w}}$ at ι'; so, $J.\texttt{cmp}([u,v]) = \texttt{T}$ stands still, thus, $J.\texttt{cmp}^{\iota''}[(u,v)] = \texttt{T}$.
    * If $(u,v) \in E$ is incompatible w.r.t. f^c in $\Gamma_{i',j'}$, i.e., $f^c(u) < f^c(v) - (w(u,v) - i' - F_{j'})$, then, since $f^c(u) = f^c(v) = 0$, we have:

      $$0 = f^c(u) < f^c(v) - w(u,v) + i' + F_{j'} = -w(u,v) + i' + F_{j'}.$$

      Thus, $w(u,v) \le i'$, because $f^c(u) = f^c(v) = 0$ and $j' \in [1, s-1]$ (so $F_{j'} > 0$). On the other side, since $(u,v) \in E$ is compatible w.r.t. f^c in $\Gamma_{i,s-1}$, at this point the reader can check that $w(u,v) > i$. Then, by Item 1 of Lemma 3, and since $L_{\mathbf{w}}$ was sorted in increasing order, the entry $(w(u,v), L_\alpha)$ is still inside $L_{\mathbf{w}}$ at ι. Therefore, since $w(u,v) \le i'$, there must be some step $\hat{\iota}$ (such that $\iota < \hat{\iota} \le \iota'$) at which the entry $(w(u,v), L_\alpha)$ must have been considered, either at line 5 or line 10 of `ei-jump`(i,J), and thus popped from $L_{\mathbf{w}}$. Soon after $\hat{\iota}$, the subsequent invocation of `repair()` (either at line 8 or line 12 of `ei-jump()`) changes the state of $J.\texttt{cmp}^{\hat{\iota}}[(u,v)]$ from T to F (line 5 of `repair()`), and it decrements $J.\texttt{cnt}^{\hat{\iota}}[(u,v)]$ by one unit (line 4 of `repair()`). Thus $J.\texttt{cmp}$ gets *repaired* so that to be coherent w.r.t. f^c in $\Gamma_{w(u,v),j'}$. Now, by Item 2 of Lemma 3, $J.i$ can only increase during the execution of `ei-jump()`. So, from that point on, (u,v) remains incompatible w.r.t. f^{c} in $\Gamma_{J.i,j'}$ for every $w(u,v) \leq J.i \leq i'$. On the other hand, $J.\mathtt{cmp}^{\hat{\iota}}[(u,v)] = \mathtt{F}$ stands still, since $\mathtt{ei\text{-}jump}()$ (SubProcedure 6) never changes it from F to T. So, $J.\mathtt{cmp}^{\iota''}[(u,v)] = \mathtt{F}$.

This proves that, in any case, $J.\mathtt{cmp}^{\iota''}[(u,v)]$ is coherent w.r.t. f^{c} in $\Gamma_{i',j'}$.

This also proves that $J.\mathtt{cnt}^{\iota''}$ is coherent w.r.t. f^{c} in $\Gamma_{i',j'}$: indeed, $J.\mathtt{cnt}^{\iota}$ was coherent w.r.t. f^{c} in $\Gamma_{i,s-1}$ by (*eij-PC-4*); then $J.\mathtt{cnt}$ was decremented by one unit (line 4 of $\mathtt{repair}()$) each time that $J.\mathtt{cmp}$ was repaired (line 5 of $\mathtt{repair}()$), as described above; therefore, at step ι'', the coherency of $J.\mathtt{cnt}^{\iota''}$ follows by that of $J.\mathtt{cmp}^{\iota''}$.

□

Proof of Item (2). Let $u \in V$. We want to prove that u is consistent w.r.t. f^{c} in $\Gamma_{i',j'}$ for every $i' \in [W^-, W^+]$ and $j' \in [1, s-1]$ such that $(i, s-1) < (i', j') < (J.i^h, 1)$ (if any).

By (*eij-PC-2*), either $f^{\mathrm{c}}(u) = 0$ or $f^{\mathrm{c}}(u) = \top$. If $f^{\mathrm{c}}(u) = \top$, the claim holds trivially. Assume $f^{\mathrm{c}}(u) = 0$. By (*eij-PC-1*), u is consistent w.r.t. f^{c} in $\Gamma_{i,s-1}$. Assume $\hat{w} = J.i^{\iota'} = i'$, either at line 6 or line 11 of $\mathtt{ei\text{-}jump}(i,J)$, for some step $\iota' > \iota$. Assume $\mathtt{repair}(L_{\hat{\alpha}}, J)$ halts soon after at line 8 or 12 (respectively), for some step ι'' where $\iota'' > \iota'$. We claim u is consistent w.r.t. f^{c} in $\Gamma_{i',j'}$.

If $u \in V_0$, By (*eij-PC-4*), $J.\mathtt{cnt}^{\iota}[u]$ is coherent w.r.t. f^{c} in $\Gamma_{i,s-1}$. Thus, since u is consistent w.r.t. f^{c} in $\Gamma_{i,s-1}$, it holds that $J.\mathtt{cnt}^{\iota}[u] > 0$. Now, since $(i',j') < (J.i^h, 1)$, then $i' < J.i^h$, thus $L^{\mathrm{inc}\,\iota''} = \emptyset$. Therefore, $J.\mathtt{cnt}^{\iota''}[u] > 0$ (otherwise u would have been inserted into L^{inc} within ι'' at line 7 of $\mathtt{repair}()$). By Item 1 of Proposition 9, $J.\mathtt{cnt}^{\iota''}[u]$ is coherent w.r.t. f^{c} in $\Gamma_{i',j'}$. Since $J.\mathtt{cnt}^{\iota''}[u] > 0$ and $J.\mathtt{cnt}[u]^{\iota''}$ is coherent, u is consistent w.r.t. f^{c} in $\Gamma_{i',j'}$.

If $u \in V_1$, Since $(i',j') < (J.i^h, i)$, then $i' < J.i^h$, thus $L^{\mathrm{inc}\,\iota''} = \emptyset$. Let $v \in N_\Gamma^{\mathrm{out}}(u)$. By (*eij-PC-2*), either $f^{\mathrm{c}}(v) = 0$ or $f^{\mathrm{c}}(v) = \top$. Since $f^{\mathrm{c}}(u) = 0$ by assumption, $u \in V_1$, and u is consistent w.r.t. f^{c} in $\Gamma_{i,s-1}$ by (*eij-PC-1*), then $f^{\mathrm{c}}(v) = 0$.

Now, we argue that $w(u,v) > i'$.

Assume, for the sake of contradiction, that $w(u,v) \leq i'$. On one side, since u is consistent w.r.t. f^{c} in $\Gamma_{i,s-1}$ and since $f^{\mathrm{c}}(u) = f^{\mathrm{c}}(v) = 0$, then:

$$0 = f^{\mathrm{c}}(u) \geq f^{\mathrm{c}}(v) - w(u,v) + i + F_{s-1} = -w(u,v) + i + 1.$$

Thus, $w(u,v) \geq i+1 > i$. Therefore, by Lemma 3, the entry $(w(u,v), L_\alpha)$ is still inside $L_{\mathbf{w}}$ at step ι. On the other side, since $w(u,v) \leq i'$, it is easy to see at this point that within (or soon after) step ι' the entry $(w(u,v), L_{\alpha'})$ must have been popped from $L_{\mathbf{w}}$ either at line 7 or at line 10 of $\mathtt{ei\text{-}jump}()$ (SubProcedure 6). So, $(w(u,v), L_{\alpha'})$ must have been popped from $L_{\mathbf{w}}$ after ι and within ι' (or soon after ι' at line 7). But soon after that, since $u \in V_1$, the subsequent invocation of $\mathtt{repair}()$ would insert u into L^{inc} at line 8, because $f^{\mathrm{c}}(u) = f^{\mathrm{c}}(v) = 0$ and $L^{\mathrm{inc}}[u] = \emptyset$ at line 2. Therefore $L^{\mathrm{inc}\,\iota''} \neq \emptyset$, which is a contradiction. Therefore, $w(u,v) > i'$. Since $v \in N_\Gamma^{\mathrm{out}}(u)$ was chosen arbitrarily, $\forall^{v \in N_\Gamma^{\mathrm{out}}(u)}\ w(u,v) > i'$.

Since $\forall^{v \in N_\Gamma^{\mathrm{out}}(u)}\ w(u,v) > i'$ and $f^{\mathrm{c}}(u) = f^{\mathrm{c}}(v) = 0$, then u is consistent w.r.t. f^{c} in $\Gamma_{i',j'}$.

So, u is consistent w.r.t. f^{c} in $\Gamma_{i',j'}$. Since $u \in V$ was chosen arbitrarily, then $\mathrm{Inc}(f^{\mathrm{c}:\iota}, i', j') = \emptyset$. □

Proof of Item (3). By line 9 of `ei-jump`() (SubProcedure 6), when the `while` loop at lines 9-12 halts, then `ei-jump`() halts soon after at line 13, say at step h, and it must be that either ${L^{\text{inc}}}^h \neq \emptyset$ or both ${L^{\text{inc}}}^h = \emptyset$ and $L^h_{\mathbf{w}} = \emptyset$. Now, we want to prove that ${L^{\text{inc}}}^h = \text{Inc}(f^c, J.i^h, 1)$.

- Firstly, ${L^{\text{inc}}}^h \subseteq \text{Inc}(f^c, J.i^h, 1)$:

Assume $u \in {L^{\text{inc}}}^h$. We have three cases to check:

– If $u \in {L^{\text{inc}}_{\text{cpy}}}^{\iota}$, notice that $J.i^h > i$ by Lemma 3, then $u \in \text{Inc}(f^c, J.i^h, 1)$ holds by (eij-PC-3);

– If $u \in V_0 \setminus {L^{\text{inc}}_{\text{cpy}}}^{\iota}$, then $J.\texttt{cnt}^h[u] = 0$ by lines 6-7 of `repair`() (SubProcedure 6). By Item 1 of Proposition 9, $J.\texttt{cnt}^h[u]$ is coherent w.r.t. f^c in $\Gamma_{J.i^h,1}$. Therefore, $u \in \text{Inc}(f^c, J.i^h, 1)$.

– If $u \in V_1 \setminus {L^{\text{inc}}_{\text{cpy}}}^{\iota}$, then $\exists^{v \in N^{\text{out}}_{\Gamma}(u)}\ f^c(u) = f^c(v) = 0$ and $w(u,v) = J.i^h$ by lines 2-8 of `repair`(). Therefore, $u \in \text{Inc}(f^c, J.i^h, 1)$.

This proves, ${L^{\text{inc}}}^h \subseteq \text{Inc}(f^c, J.i^h, 1)$.

- Secondly, ${L^{\text{inc}}}^h \supseteq \text{Inc}(f^c, J.i^h, 1)$:

Let $u \in \text{Inc}(f^c, J.i^h, 1)$. By *(eij-PC-2)*, either $f^c(u) = 0$ or $f^c(u) = \top$. Since $u \in \text{Inc}(f^c, J.i^h, 1)$, then $f^c(u) = 0$. Now, by *(eij-PC-1)*, u is consistent w.r.t. f^c in $\Gamma_{i,s-1}$. So, let $(u, \hat{v}) \in E$ be any arc which is compatible w.r.t. f^c in $\Gamma_{i,s-1}$.

If $u \in V_0$, then, *at least one* such a compatible $\hat{v} \in N^{\text{out}}_{\Gamma}(u)$ exists (because $u \in V_0$ is consistent w.r.t. f^c in $\Gamma_{i,s-1}$). By (*eij-PC-2*), either $f^c(\hat{v}) = 0$ or $f^c(\hat{v}) = \top$. Since $f^c(u) = 0$ and $(u, \hat{v})$ is compatible w.r.t. f^c in $\Gamma_{i,s-1}$, then $f^c(\hat{v}) = 0$. Since $f^c(u) = f^c(\hat{v}) = 0$ and $u \in \text{Inc}(f^c, J.i^h, 1)$, then $w(u, \hat{v}) \leq J.i^h$.

We claim that at some step of execution of line 7 or line 10 in `ei-jump`() (SubProcedure 6), say at step ι' for $\iota < \iota' < h$, the entry $(w(u,\hat{v}), L_\alpha)$ is popped from $L_{\mathbf{w}}$.

Since $f^c(u) = f^c(\hat{v}) = 0$, and $(u, \hat{v})$ is compatible w.r.t. f^c in $\Gamma_{i,s-1}$, then $w(u, \hat{v}) > i$. Thus, by Lemma 3, when `ei-jump`() is invoked (i.e., at step ι), the front entry $(\bar{w}, L_{\bar{\alpha}})$ of $L^{\iota}_{\mathbf{w}}$ satisfies $\bar{w} = \min\{w_e \mid e \in E, w_e > i\} \leq w(u, \hat{v})$. So, $\bar{w} \leq w(u, \hat{v}) \leq J.i^h$. Thus, at some step of execution ι' for $\iota < \iota' < h$, the entry $(w(u,\hat{v}), L_\alpha)$ must be popped from $L_{\mathbf{w}}$.

Soon after that, `repair`(L_α, J) is invoked: there, since $f^c(u) = f^c(\hat{v}) = 0$ and $u \in V_0$, then $J.\texttt{cnt}[u]$ is decremented by one unit at line 4 of `repair`().

Indeed, this happens (after ι but before h), *for every* $v \in N^{\text{out}}_{\Gamma}(u)$ such that $(u, v) \in E$ is compatible w.r.t. f^c in $\Gamma_{i,s-1}$. Thus, eventually and before h, it will hold $J.\texttt{cnt}[u] = 0$. At that point, u will be inserted into L^{inc} at line 7 of `repair`(); and soon after, `ei-jump`() halts (since $L^{\text{inc}} \neq \emptyset$ at line 9). So, $u \in {L^{\text{inc}}}^h$. This holds for every $u \in \text{Inc}(f^c, J.i^h, 1) \cap V_0$. Thus, $\text{Inc}(f^c, J.i^h, 1) \cap V_0 \subseteq {L^{\text{inc}}}^h$.

If $u \in V_1$, then, *all* $\hat{v} \in N^{\text{out}}_{\Gamma}(u)$ are such that $(u, \hat{v})$ is compatible w.r.t. f^c in $\Gamma_{i,s-1}$, because $u \in V_1$ is consistent w.r.t. f^c in $\Gamma_{i,s-1}$ by (*eij-PC-1*). The argument proceeds almost in the same way as before. By *(eij-PC-2)*, either $f^c(\hat{v}) = 0$ or $f^c(\hat{v}) = \top$. Since $f^c(u) = 0$ and $(u, \hat{v})$ is compatible w.r.t. f^c in $\Gamma_{i,s-1}$, then $f^c(\hat{v}) = 0$. Since $f^c(u) = f^c(\hat{v}) = 0$ and $u \in \text{Inc}(f^c, J.i^h, 1)$, then $w(u, \hat{v}) \leq J.i^h$. By arguing as above, we see that at some step of execution of line 3 in (the considered invocation of) `ei-jump`() (SubProcedure 6), say at step ι' for $\iota < \iota' < h$, the entry $(w(u, \hat{v}), L_\alpha)$ is popped from $L_{\mathbf{w}}$. Soon after that, `repair`(L_α, J) is invoked: there, since $f^{\rm c}(u) = f^{\rm c}(\hat{v}) = 0$ and $u \in V_1$, then u is inserted into $L^{\rm inc}$; soon after, `ei-jump()` halts (since $L^{\rm inc} \neq \emptyset$); so, $u \in {L^{\rm inc}}^h$. This holds for every $u \in {\rm Inc}(f^{\rm c}, J.i^h, 1) \cap V_1$; so, ${\rm Inc}(f^{\rm c}, J.i^h, 1) \cap V_1 \subseteq {L^{\rm inc}}^h$.

Therefore, ${\rm Inc}(f^{\rm c}, J.i^h, 1) = {L^{\rm inc}}^h$. □

Correctness of `ua-jumps()` *(SubProcedure 7)*

Proposition 10. *Consider any invocation of* `ei-jump()` *(SubProcedure 6) that is made at line 7 of Algorithm 1. Assume that the pre-conditions (eij-PC-1), (eij-PC-2), (eij-PC-3), (eij-PC-4), are all satisfied at invocation time. Further assume that $L^{inc} \neq \emptyset$ at line 8 of Algorithm 1, so that* `ua-jumps()` *is invoked soon after at line 9. Then, consider any invocation of* $\texttt{J-VI}(i, s-1, F, J, \Gamma)$ *(SubProcedure 4) that is made at line 2 of* `ua-jumps()` *(SubProcedure 7), for some $i \in [W^-, W^+]$, where $s = |\mathscr{F}_{|V|}|$. Then, the following properties hold.*

1. *The* (PC-1), (w-PC-2), (w-PC-3) *are all satisfied by that invocation of* $\texttt{J-VI}(i, s-1, F, J, \Gamma)$.
2. *When the* $\texttt{J-VI}(i, s-1, F, J, \Gamma)$ *halts, say at step h, then the following holds:*

$$L^h_\top = \mathscr{W}_0(\Gamma_{i-1,s-1}) \cap \mathscr{W}_1(\Gamma_{i,s-1}).$$

3. *Assume that* $\texttt{backtrack\_ua-jumps}(i, s, F, J, \Gamma)$ *is invoked at line 7 of* `ua-jumps()`*, say at step ι, and assume that it halts at step h.*
   (a) *At line 1 of* $\texttt{backtrack\_ua-jumps}(i, s, F, J, \Gamma)$*, it holds:*

$$L^{inc}_{cpy} = \{v \in V \mid 0 < f^{c:\iota}(v) \neq \top\}.$$

   (b) *Consider the two induced games $\Gamma[L^\iota_\top]$ and $\Gamma[V \setminus L^\iota_\top]$. The following holds:*
      i. $\forall (v \in L^\iota_\top)\ f^{c:h}(v) = f^*_{w'_{i-1,s-1}}(v)$;
      ii. $\forall (v \in V \setminus L^\iota_\top)\ f^{c:h}(v) = f^*_{w'_{i,s-1}}(v)$;
      iii. $\forall (u \in L^\iota_\top \cap V_0) \forall (v \in N^{out}_{\Gamma[L^\iota_\top]}(u))\ J.\texttt{cmp}^h(u, v)$ *is coherent w.r.t.* $f^{c:h}$ *in* $\Gamma[L^\iota_\top]_{i,1}$;
      iv. $\forall (v \in L^\iota_\top \cap V_0)\ J.\texttt{cnt}^h(v)$ *is coherent w.r.t.* $f^{c:h}$ *in* $\Gamma[L^\iota_\top]_{i,1}$;
      v. $\forall (j' \in [1, s-1])\ Inc(f^{c:h}, i, j') \setminus L^\iota_\top = \emptyset$.
4. *Any invocation of* `ua-jumps()` *(SubProcedure 7) (line 7, Algorithm 1) halts in finite time.*

Proof of Item (1). By induction on the number $k \in \mathbb{N}$ of invocations of `J-VI()` that are made at line 2 of `ua-jumps()`.

Base Case: $k = 1$. Consider the first invocation of `J-VI()` at line 2 of `ua-jumps()`, say it happens at step ι. Just before step ι, Algorithm 1 invoked `ei-jump()` at line 7. By hypothesis, (*eij-PC-1*), (*eij-PC-2*), (*eij-PC-3*), (*eij-PC-4*) are all satisfied at that time. Then:

– (*PC-1*): It is easy to check from the definitions that (*eij-PC-1*) directly implies (*PC-1*).

– (*w-PC-2*): By Item 3 of Proposition 9, it holds that ${L^{\rm inc}}^\iota = {\rm Inc}(f^{{\rm c}:\iota}, i, 1)$. Since ${\rm Inc}(f^{{\rm c}:\iota}, i, 1) \subseteq {\rm Inc}(f^{{\rm c}:\iota}, i, s-1)$, then (*w-PC-2*) holds.

– (*w-PC-3*): Let $u \in V \setminus {L^{\rm inc}}^\iota$ and $v \in N^{\rm out}_\Gamma(u)$. We need to check the following two cases.

If $u \in V_0$, by Item 1 of Proposition 9, both $J.\texttt{cmp}^\iota$ and $J.\texttt{cnt}^\iota$ are coherent w.r.t. $f^{{\rm c}:\iota}$ in $\Gamma_{i,1}$. Therefore, (*w-PC-3*) holds when $u \in V_0$.

If $u \in V_1$ and (u,v) is incompatible w.r.t. $f^{c:\iota}$ in $\Gamma_{i,1}$, then, $u \in \text{Inc}(f^{c:\iota}, i, 1)$. By Item 3 of Proposition 9, $\text{Inc}(f^{c:\iota}, i, 1) = L^{\text{inc}^\iota}$. Thus, $u \in L^{\text{inc}^\iota}$, so (*w-PC-3*) holds when $u \in V_1$.

Therefore, (*w-PC-3*) holds when $k = 1$.

Inductive Step: $k > 1$. Consider the k-th invocation of `J-VI`() for $k > 1$, at line 2 of `ua-jumps`(). Say it happens at step ι. Since $k > 1$, just before step ι, Algorithm 1 performed the $(k-1)$-th invocation of `J-VI`() at line 2 of `ua-jumps`(). Say it happened at step ι_0. By induction hypothesis, at step ι_0 the (*PC-1*), (*w-PC-2*), (*w-PC-3*) were all satisfied. Therefore, the $(k-1)$-th invocation of `J-VI`() at line 2 halted in a correct manner, as prescribed by Proposition 8. Soon after that, Algorithm 1 invoked `rejoin_ua-jump`() at line 5 of `ua-jumps`().

($\star$) The key is that `rejoin_ua-jump`(), apart from copying the energy-levels of L_f back to $J.f$ (with `scl_back_`$f(s-1,F,J)$ at line 1), it takes care of repairing (at line 6) the coherency state of $J.\texttt{cnt}[u]$ and $J.\texttt{cmp}[(u,v)]$ for all those $(u,v) \in E$ such that: $u \in V_0$, $w(u,v) = i$ and $J.f[u] = J.f[v] = 0$ (if any); moreover, it checks the compatibility state of all those arcs $(u,v) \in E$ such that: $u \in V_1$, $w(u,v) = i$ and $J.f[u] = J.f[v] = 0$ (if any). In doing so, if any $u \in V \setminus L^{\text{inc}}$ is recognized to be inconsistent w.r.t. f^c in $\Gamma_{i,s-1}$, then u is (correctly) inserted into L^{inc}. See the pseudo-code of `repair`() in SubProcedure 6.

With ($\star$) in mind, we can check that (*PC-1*), (*w-PC-2*), (*w-PC-3*) are all satisfied at step ι.

– (*PC-1*). Since `rejoin_ua-jump`() empties L_f (by `scl_back_`f() at line 1), then $L^\iota_f = \emptyset$. Next, we argue $f^c \preceq f^*_{w'_{i,s-1}}$. By induction hypothesis, when the $(k-1)$-th invocation of `J-VI`() at line 2 of `ua-jumps`() halts, Proposition 8 holds, therefore, $f^c = f^*_{w'_{i-1,s-1}}$. Since $F_{s-1} = 1$, then $w'_{i-1,s-1} = w_{i-1,s-1}$ and $w'_{i,s-1} = w_{i,s-1}$. Therefore, the following holds for every $v \in V$:

$$
\begin{aligned}
f^c(v) =& f^*_{w'_{i-1,s-1}}(v) && \text{[by induction hypothesis and Proposition 8]} \\
=& f^*_{i-1,s-1}(v) && [\text{by } w'_{i-1,s-1} = w_{i-1,s-1}] \\
\preceq& f^*_{i,s-1}(v) && [\text{by } w_{i-1,s-1} > w_{i,s-1} \text{ and Lemma 2}] \\
=& f^*_{w'_{i,s-1}}(v) && [\text{by } w_{i,s-1} = w'_{i,s-1}]
\end{aligned}
$$

In summary, $\forall^{v \in V} f^c(v) \preceq f^*_{w'_{i,s-1}}(v)$. This proves (PC-1).

– (w-PC-2). By induction hypothesis and Proposition 6, all vertices that were already inside L^{inc} at the end of the $(k-1)$-th invocation of `J-VI`(), at line 2 of `ua-jumps`(), they were all inconsistent w.r.t. $J.f^c$ in $\Gamma_{i-1,s-1}$, so they are still inconsistent w.r.t. $J.f^c$ in $\Gamma_{i,s-1}$, because $w'_{i,s-1} < w'_{i-1,s-1}$. In addition, the repairing process performed by `rejoin_ua-jumps`(), as mentioned in ($\star$), can only add inconsistent vertices to L^{inc}. Therefore, (w-PC-2) holds.

– (w-PC-3). Let $u \in V \setminus L^{\text{inc}^\iota}$ and $v \in N^{\text{out}}_\Gamma(u)$. We need to check the following two cases.

Case $u \in V_0$. In order to prove Item 1 of (w-PC-3), we need to check three cases.

1. If $J.\texttt{cmp}^\iota[u,v] = \texttt{F}$, we argue that $(u,v) \in E$ is incompatible w.r.t. f^c in $\Gamma_{i,s-1}$. Indeed, one of the following two cases (i) or (ii) holds:
(i) $J.\texttt{cmp}[(u,v)] = \texttt{F}$ was already so at the end of the $(k-1)$-th invocation of `J-VI`(). By induction hypothesis and by Item 2 of Proposition 8, then (u,v) was incompatible w.r.t. f^c in $\Gamma_{i-1,s-1}$. So, (u,v) is incompatible w.r.t. f^c in $\Gamma_{i,s-1}$ (as $w'_{i,s-1} < w'_{i-1,s-1}$).
(ii) at the end of the $(k-1)$-th invocation of `J-VI`(), it was $J.\texttt{cmp}[(u,v)] = \texttt{T}$. But then, `rejoin_ua-jump`$(i,s,F,J)$ repaired it by setting $J.\texttt{cmp}[(u,v)] = \texttt{F}$ at line 5 of

$\texttt{repair}()$; notice that this (correctly) happens *iff* $w(u,v)=i$ and $J.f[u]=J.f[v]=0$, so that (u,v) is really incompatible w.r.t. $f^{\rm c}$ in $\Gamma_{i,s-1}$.
Therefore, in any case, Item 1 of (w-PC-3) holds.

2. If $J.\texttt{cmp}^{\iota}[(u,v)]=\texttt{T}$, we argue that either (u,v) is compatible w.r.t. $f^{\rm c}$ in $\Gamma_{i,s-1}$ or $v\in L^{\rm inc^{\iota}}$. Indeed, assume $J.\texttt{cmp}^{\iota}[(u,v)]=\texttt{T}$ and that (u,v) is incompatible w.r.t. $f^{\rm c}$ in $\Gamma_{i,s-1}$. Since $J.\texttt{cmp}^{\iota}[(u,v)]=\texttt{T}$, then it was as such even when the $(k-1)$-th invocation of $\texttt{J-VI}()$ halted at line 2 of $\texttt{ua-jumps}()$. By induction hypothesis and by Item 2 of Proposition 8, (u,v) was compatible w.r.t. $f^{\rm c}$ in $\Gamma_{i-1,s-1}$. But (u,v) is now incompatible w.r.t. $f^{\rm c}$ in $\Gamma_{i,s-1}$, and still $J.\texttt{cmp}^{\iota}[(u,v)]=\texttt{T}$. Thus, the last invocation of $\texttt{repair}()$, within the last $\texttt{rejoin\_ua-jump}()$, has not recognized (u,v) as incompatible (otherwise, it would be $J.\texttt{cmp}^{\iota}[(u,v)]=\texttt{F}$). Therefore, it must be that $f^{\rm c}(u)>0$ or $f^{\rm c}(v)>0$: otherwise, if $f^{\rm c}(u)=f^{\rm c}(v)=0$, since (u,v) is compatible w.r.t. $f^{\rm c}$ in $\Gamma_{i-1,s-1}$ but incompatible in $\Gamma_{i,s-1}$, and $w(u,v)\in\mathbb{Z}$, then $w(u,v)=i$ (contradicting the fact that the last invocation of $\texttt{repair}()$ has not recognized (u,v) as incompatible). Moreover, since (u,v) is incompatible w.r.t. $f^{\rm c}$ in $\Gamma_{i,s-1}$, then $f^{\rm c}(u)\neq\top$; and since (u,v) was compatible w.r.t. $f^{\rm c}$ in $\Gamma_{i-1,s-1}$ and $f^{\rm c}(u)\neq\top$, then $f^{\rm c}(v)\neq\top$. Now, when the $(k-1)$-th invocation of $\texttt{J-VI}()$ halts, $L^{\rm inc}=\{q\in V\mid 0<f^{\rm c}(q)\neq\top\}$ holds by induction hypothesis and Item 3 of Proposition 8. Since $u\notin L^{\rm inc^{\iota}}$ and $f^{\rm c}(u)\neq\top$, then $f^{\rm c}(u)=0$. Thus, since either $f^{\rm c}(u)>0$ or $f^{\rm c}(v)>0$, it holds that $f^{\rm c}(v)>0$. So, it is $0<f^{\rm c}(v)\neq\top$. Therefore, $v\in L^{\rm inc^{\iota}}$.

3. By induction hypothesis and by Proposition 8, when the $(k-1)$-th invocation of $\texttt{J-VI}()$ halts at line 2 of $\texttt{ua-jumps}()$, say at step ι_0, $f^{\rm c}$ is the least-SEPM of the EG $\Gamma_{i-1,s-1}$ and $J.\texttt{cnt}^{\iota_0}$, $J.\texttt{cmp}^{\iota_0}$ are both coherent w.r.t. $f^{\rm c}$ in $\Gamma_{i-1,s-1}$. Therefore,

$$\begin{aligned}J.\texttt{cnt}^{\iota_0}[u] &= \left|\left\{v\in N^{\rm out}_{\Gamma}(u)\mid f^{\rm c}(u)\succeq f^{\rm c}(v)\ominus w'_{i-1,s-1}(u,v)\right\}\right| && \text{[by coherency of } J.\texttt{cnt}^{\iota_0}] \\ &= \left|\left\{v\in N^{\rm out}_{\Gamma}(u)\mid J.\texttt{cmp}^{\iota_0}[(u,v)]=\texttt{T}\right\}\right|. && \text{[by coherency of } J.\texttt{cmp}^{\iota_0}]\end{aligned}$$

Moreover, since $f^{\rm c}$ is least-SEPM of $\Gamma_{i-1,s-1}$, then u is consistent w.r.t. $f^{\rm c}$ in $\Gamma_{i-1,s-1}$; thus $J.\texttt{cnt}^{\iota_0}[u]>0$. Then, after ι_0 and before ι, $\texttt{ua-jumps}()$ increments i by one unit at line 4 and it invokes $\texttt{rejoin\_ua-jump}()$ at line 5. There, $\texttt{repair}()$ can (possibly) alter the state of both $J.\texttt{cnt}$ and $J.\texttt{cmp}$ at lines 4-5. Whenever the state of $J.\texttt{cmp}$ is modified from T to F, then $J.\texttt{cnt}$ is decremented by one unit; moreover, whenever $J.\texttt{cnt}[u]=0$, then $\texttt{repair}()$ takes care of inserting u into $L^{\rm inc}$. Therefore, $J.\texttt{cnt}^{\iota}[u]=\left|\left\{v\in N^{\rm out}_{\Gamma}(u)\mid J.\texttt{cmp}^{\iota}[(u,v)]=\texttt{T}\right\}\right|$; since $u\notin L^{\rm inc^{\iota}}$, then $J.\texttt{cnt}^{\iota}[u]>0$.

Case $u\in V_1$. Let $v\in N^{\rm out}_{\Gamma}(u)$ be such that (u,v) is incompatible w.r.t. $f^{\rm c}$ in $\Gamma_{i,j}$. We claim $v\in L^{\rm inc^{\iota}}$. By induction hypothesis and by Proposition 8, when the $(k-1)$-th invocation of $\texttt{J-VI}()$ halts at line 2 of $\texttt{ua-jumps}()$, say at step ι_0, $f^{\rm c}$ is the least-SEPM of the EG $\Gamma_{i-1,s-1}$ and $L^{\rm inc^{\iota_0}}=\{q\in V\mid 0<f^{\rm c}(q)\neq\top\}$. Thus, since $u\in V_1$, the arc (u,v) is compatible w.r.t. $f^{\rm c}$ in $\Gamma_{i-1,s-1}$. Moreover, since $u\notin L^{\rm inc^{\iota}}$ by hypothesis, then $u\notin L^{\rm inc^{\iota_0}}$, thus $f^{\rm c}(u)=0$ or $f^{\rm c}(u)=\top$; but since u is incompatible w.r.t. $f^{\rm c}$ in $\Gamma_{i,j}$, it is $f^{\rm c}(u)=0$. Now, if $0<f^{\rm c}(v)\neq\top$, then $v\in L^{\rm inc^{\iota_0}}\subseteq L^{\rm inc^{\iota}}$, so we are done. Otherwise, since (u,v) is compatible w.r.t. $f^{\rm c}$ in $\Gamma_{i-1,s-1}$ and $f^{\rm c}(u)=0$, then $f^{\rm c}(v)\neq\top$. So, $f^{\rm c}(v)=0$. Since $f^{\rm c}(u)=f^{\rm c}(v)=0$ and (u,v) is compatible w.r.t. $f^{\rm c}$ in $\Gamma_{i-1,s-1}$, but incompatible w.r.t. $f^{\rm c}$ in $\Gamma_{i,s-1}$, and $w(u,v)\in\mathbb{Z}$, then $w(u,v)=i$. Whence, soon after ι_0, when $\texttt{ua-jumps}()$ invokes $\texttt{rejoin\_ua-jump}()$ at line 5; there inside, $\texttt{repair}()$ takes care of inserting v into $L^{\rm inc}$. Therefore, $v\in L^{\rm inc^{\iota}}$.

This concludes the inductive step, and thus the proof of Item 1 of Proposition 10. $\square$

Proof of Item (2). Consider the first invocation of `J-VI`() at line 2 of `ua-jumps`(), say it happens at step ι. Notice that, just before step ι, the `ei-jump`() was invoked by Algorithm 1 at line 7, say at step ι_0. By hypothesis, (*eij-PC-1*), (*eij-PC-2*), (*eij-PC-3*), (*eij-PC-4*) were all satisfied at step ι_0. By (*eij-PC-1*) and Item 2 of Proposition 9, $f^{c:\iota_0}$ is the least-SEPM of $\Gamma_{i-1,s-1}$; therefore, $V_{f^{c:\iota_0}} = \mathscr{W}_0(\Gamma_{i-1,s-1})$ by Proposition 3. By Item 1 of Proposition 10 and Item 1 of Proposition 8, when the first invocation of `J-VI`() halts, say at step h, then $f^{c:h}$ is the least-SEPM of $\Gamma_{i,s-1}$; therefore, $V \setminus V_{f^{c:h}} = \mathscr{W}_1(\Gamma_{i,s-1})$ by Proposition 3. Moreover, by Item 4 of Proposition 8, $L^h_\top = V_{f^{c:\iota}} \cap V \setminus V_{f^{c:h}}$. Therefore, $L^h_\top = \mathscr{W}_0(\Gamma_{i-1,s-1}) \cap \mathscr{W}_1(\Gamma_{i,s-1})$.

Next, consider the k-th invocation of `J-VI()`, for $k > 1$, at line 2 of `ua-jumps`(). By Item 1 of Proposition 10 and Item 1 of Proposition 8, the following two hold: (i) when the $(k-1)$-th invocation of `J-VI`() halts, at line 2 of `ua-jumps`(), say at step ι, then $f^{c:\iota}$ is the least-SEPM of $\Gamma_{i-1,s-1}$; and $V_{f^{c:\iota}} = \mathscr{W}_0(\Gamma_{i-1,s-1})$ by Proposition 3. (ii) when the k-th invocation of `J-VI`() halts, at line 2 of `ua-jumps`(), say at step h, then $f^{c:h}$ is the least-SEPM of $\Gamma_{i,s-1}$; and $V \setminus V_{f^{c:h}} = \mathscr{W}_1(\Gamma_{i,s-1})$ by Proposition 3. Notice that, when the k-th invocation of `J-VI`() takes place, soon after ι, the current energy-levels are still $f^{c:\iota}$ (i.e., they are not modified by `rejoin_ua-jumps`() at line 5 of `ua-jumps`()). Moreover, by Item 4 of Proposition 8, $L^h_\top = V_{f^{c:\iota}} \cap V \setminus V_{f^{c:h}}$. Therefore, by (i) and (ii), it holds $L^h_\top = \mathscr{W}_0(\Gamma_{i-1,s-1}) \cap \mathscr{W}_1(\Gamma_{i,s-1})$. $\square$

Proof of Item (3). We need to check the following two items (a) and (b).

(a) Consider the state of $L^{\text{inc}}_{\text{cpy}}$ at line 1 of `backtrack_ua-jumps`(i,s,F,J,Γ). By Item 1 of Proposition 10, and by Item 3 of Proposition 8, when the last invocation of `J-VI`() halts at line 2 of `ua-jumps`(), say at step h_0, it holds $J.L^{\text{inc}^{h_0}} = \{v \in V \mid 0 < f^{c:h_0}(v) \neq \top\}$. By the copy operation which is performed at line 1 of `backtrack_ua-jumps`(), then $J.L^{\text{inc}}_{\text{cpy}} = J.L^{\text{inc}^{h_0}} = \{v \in V \mid 0 < f^{c:h_0}(v) \neq \top\}$. This proves (a).

(b) Let us focus on the two induced games $\Gamma[L^\iota_\top]$ and $\Gamma[V \setminus L^\iota_\top]$.

i. $\forall (v \in L^\iota_\top)\, f^{c:h}(v) = f^*_{w'_{i-1,s-1}}(v)$: indeed, by arguing similarly as in the proof of Item 2 of Proposition 10, $\forall^{v\in V}\, J.f^\iota[v] = f^*_{w'_{i-1,s-1}}(v)$. Notice that `backtrack_ua-jump`() modifies the energy-levels only at lines 2-3, where the following assignment is performed:

$$L^h_f[u] \leftarrow \begin{cases} \bot & \text{, if } u \in L^\iota_\top; \\ L^\iota_f[u] & \text{, if } u \in V \setminus L^\iota_\top. \end{cases}$$

and `scl_back_`f() is invoked (respectively). Since $\forall (v \in L^\iota_\top)\, L^h_f[u] = \bot$, soon after the invocation of `scl_back_`f() at line 3, it must be that $\forall (v \in L^\iota_\top)\, f^{c:h}(v) = J.f^\iota[v] = f^*_{w'_{i-1,s-1}}(v)$. This proves (i).

ii. $\forall (v \in V \setminus L^\iota_\top)\, f^{c:h}(v) = f^*_{w'_{i,s-1}}(v)$: indeed, by Item 1 of Proposition 10 and Item 1 of Proposition 8, $\forall (v \in V)\, f^{c:\iota}(v) = f^*_{w'_{i,s-1}}(v)$. As mentioned, `backtrack_ua-jump`() modifies the energy-levels only at lines 2-3, where $L^h_f[u]$ is assigned (as above in (i)). Since $\forall (v \in V \setminus L^\iota_\top)\, (L^h_f[u] = L^\iota_f[u]$ and $f^{c:\iota}(v) = f^*_{w'_{i,s-1}}(v))$, then (ii) holds.

iii. $\forall (u \in L^{\iota}_{\top} \cap V_0) \forall (v \in N^{\text{out}}_{\Gamma[L^{\iota}_{\top}]}(u))$ $J.\texttt{cmp}^h(u,v)$ is coherent w.r.t. $f^{c:h}$ in $\Gamma[L^{\iota}_{\top}]_{i,1}$: indeed, at lines 4-7 of `backtrack_ua-jump()`, for each $u \in L^{\iota}_{\top} \cap V_0$, it is invoked the `init_cnt_cmp`$(u,i,1,F,J,\Gamma[L^{\iota}_{\top}])$ (line 7). Therefore, (iii) holds.

iv. $\forall (v \in L^{\iota}_{\top} \cap V_0)$ $J.\texttt{cnt}^h(v)$ is coherent w.r.t. $f^{c:h}$ in $\Gamma[L^{\iota}_{\top}]_{i,1}$: same argument as in (iii).

v. $\forall (j' \in [1,s-1])$ $\text{Inc}(f^{c:h},i,j') \setminus L^{\iota}_{\top} = \emptyset$: indeed, let $u \in V \setminus L^{\iota}_{\top}$ and let $j' \in [1,s-1]$ be fixed arbitrarily. We want to show that $u \notin \text{Inc}(f^{c:h},i,j')$. Since $f^*_{w'_{i,s-1}}$ is the least-SEPM of $\Gamma_{i,s-1}$, then $u \notin \text{Inc}(f^*_{w'_{i,s-1}},i,s-1)$. We have two cases.

Case $u \in V_0 \setminus L^{\iota}_{\top}$ Since $u \in V_0 \setminus \text{Inc}(f^*_{w'_{i,s-1}},i,s-1)$, for *some* $v \in N^{\text{out}}_{\Gamma}(u)$ it holds that:

$$f^*_{w'_{i,s-1}}(u) \succeq f^*_{w'_{i,s-1}}(v) \ominus w'_{i,s-1}(u,v). \qquad (*_0)$$

By Item (i) of Proposition 10, it holds that $\forall (v \in L^{\iota}_{\top})$ $f^{c:h}(v) = f^*_{w'_{i-1,s-1}}(v)$. By Item (ii) of Proposition 10, it holds that $\forall (v \in V \setminus L^{\iota}_{\top})$ $f^{c:h}(v) = f^*_{w'_{i,s-1}}(v)$; so $f^{c:h}(u) = f^*_{w'_{i,s-1}}(u)$. By Lemma 2, $f^*_{w'_{i-1,s-1}} \preceq f^*_{w'_{i,s-1}}$. Then, since $f^{c:h}(u) = f^*_{w'_{i,s-1}}(u)$, since $f^{c:h}(v) \in \{f^*_{w'_{i-1,s-1}}(v), f^*_{w'_{i,s-1}}(v)\}$ and $f^*_{w'_{i-1,s-1}} \preceq f^*_{w'_{i,s-1}}$, from $(*_0)$ we obtain the following inequality:

$$f^{c:h}(u) \succeq f^{c:h}(v) \ominus w'_{i,s-1}(u,v).$$

Now, since $w'_{i,j'}(u,v) \geq w'_{i,s-1}(u,v)$, it also holds $f^{c:h}(u) \succeq f^{c:h}(v) \ominus w'_{i,j'}(u,v)$. This proves that $u \notin \text{Inc}(f^{c:h},i,j')$.

Case $u \in V_1 \setminus L^{\iota}_{\top}$ Since $u \in V_1 \setminus \text{Inc}(f^*_{w'_{i,s-1}},i,s-1)$, for *all* $v \in N^{\text{out}}_{\Gamma}(u)$ it holds that:

$$f^*_{w'_{i,s-1}}(u) \succeq f^*_{w'_{i,s-1}}(v) \ominus w'_{i,s-1}(u,v).$$

By arguing as in the previous case, we obtain that for every $v \in N^{\text{out}}_{\Gamma}(u)$ the following holds: $f^{c:h}(u) \succeq f^{c:h}(v) \ominus w'_{i,s-1}(u,v)$. This proves that $u \notin \text{Inc}(f^{c:h},i,j')$.

□

Proof of Item (4). The fact that `ua-jumps()` halts in finite time follows directly from Item 1 of Proposition 10 and the definition of `rejoin_ua-jump()` and that of `backtrack_ua-jump()`. □

Correctness of `solve_MPG()` (Algorithm 1)

As shown next, it turns out that (PC-1), (w-PC-2), (w-PC-3) are all satisfied by Algorithm 1.

Proposition 11. *Let $i \in [W^- - 1, W^+]$ and $j \in [1,s-1]$. The following two propositions hold.*

1. *Consider any invocation of `ei-jump`(i,J) (SubProcedure 6) at line 7 of Algorithm 1 such that $L^{inc} = \emptyset$. Then, (eij-PC-1), (eij-PC-2), (eij-PC-3), (eij-PC-4) are all satisfied w.r.t. Γ.*
2. *Consider any invocation of `J-VI`$(i,j,F,J,\Gamma[S])$ at line 11 of Algorithm 1. Then, (PC-1), (w-PC-2), (w-PC-3) are all satisfied w.r.t. the sub-arena $\Gamma[S]$.*

Proof. We prove Item 1 and 2 jointly, arguing by induction on the number k_1 of invocations of `ei-jump`() at line 7 of Algorithm 1 *and* the number k_2 of invocations of `J-VI`() at line 11.

Base Case: $k_1 = 1$ *and* $k_2 = 0$. So, the first subprocedure to be invoked is `ei-jump`(i,J) at line 7 of Algorithm 1, say at step ι. Notice that: $i^{\iota} = W^- - 1$; $\forall(v \in V)\ f^{c:\iota}(v) = 0$; $\forall(v \in V_0)\ J.\mathtt{cnt}^{\iota}[v] = |N^{\text{out}}_{\Gamma}(v)|$ and $\forall(u \in V_0)\forall(v \in N^{\text{out}}_{\Gamma}(u))\ J.\text{cmp}^{\iota}[u,v] = \mathtt{T}$; $L^{\iota}_f = L^{\text{inc}^{\iota}} = L^{\text{inc}\ \iota}_{\text{cpy}} = \emptyset$. Also notice that for every $u \in V$ and $v \in N^{\text{out}}_{\Gamma}(u)$ the following holds:

$$w'_{i^{\iota},s-1}(u,v) = w'_{W^- -1,s-1}(u,v) = w(u,v) - W^- \geq 0.$$

With this, it is straightforward to check that *(eij-PC-1)*, *(eij-PC-2)*, *(eij-PC-3)*, *(eij-PC-4)* hold.

Inductive Step: $k_1 = 1$ *and* $k_2 \geq 1$, *or* $k_1 > 1$. We need to check three cases.

1. Assume that `J-VI`$(i,j,F,J,\Gamma[S])$ is invoked at line 11 of Algorithm 1, say at step ι_1, soon after that `ua-jumps`() halted at line 9 of Algorithm 1. So, we aim at showing Item 2. Notice that: $j^{\iota_1} = 1$ holds (by line 10 of Algorithm 1). Let us check the (PC-1), (w-PC-2), (w-PC-3) w.r.t. $\Gamma[S]$. By line 4 of `backtrack_ua-jump`(), $S = L^{\iota}_{\top}$ for some step $\iota < \iota_1$.
   – *PC-1*: By line 2 of `backtrack_ua-jump`(), it holds $\forall(u \in L^{\iota}_{\top})\ L^{\iota}_f[u] = \perp$. By Item [3, (b), (i)] of Proposition 10, $\forall(v \in L^{\iota}_{\top})\ f^{c:\iota_1}(v) = f^*_{w'_{i-1,s-1}}(v)$. By Lemma 2, $f^*_{w'_{i-1,s-1}} \preceq f^*_{w'_{i,j^{\iota_1}}}$. Therefore, $\forall(v \in L^{\iota}_{\top})\ f^{c:\iota_1}(v) \preceq f^*_{w'_{i,j^{\iota_1}}}(v)$. This proves that (PC-1) holds w.r.t. $\Gamma[L^{\iota}_{\top}] = \Gamma[S]$.
   – *w-PC-2:* By lines 5-17 of `backtrack_ua-jumps`(), and since `init_cnt_cmp`() is correct, $L^{\text{inc}^{\iota_1}} \subseteq \text{Inc}(f^{c:\iota_1}, i, 1)$. Since $j^{\iota_1} = 1$, then (w-PC-2) holds w.r.t. $\Gamma[L^{\iota}_{\top}] = \Gamma[S]$.
   – *w-PC-3:* By induction hypothesis and by Item [3, (b), (iii) and (iv)] of Proposition 10, $J.\mathtt{cnt}^{\iota_1}$ and $J.\mathtt{cmp}^{\iota_1}$ are coherent w.r.t. $f^{c:\iota_1}$ in $\Gamma[L^{\iota}_{\top}]_{i,1}$; also, if $u \in V_1 \cap L^{\iota}_{\top}$ and $u \in \text{Inc}(f^{c:\iota_1}, i, 1)$, then $u \in L^{\text{inc}^{\iota_1}}$ by lines 11-17 of `backtrack_ua-jump`(). Since $j^{\iota_1} = 1$, this proves that (PC-3) holds w.r.t. $\Gamma[L^{\iota}_{\top}] = \Gamma[S]$, so (w-PC-3) holds as well.
2. Assume that `J-VI`$(i,j,F,J,\Gamma[S])$ is invoked at line 11 of Algorithm 1, say at step ι_2, soon after that a previous invocation of `J-VI`$(i,j-1,F,J,\Gamma[S])$ halted at line 11 say at step ι_1. Notice that $j \in [2, s-1]$ in that case. Let us check (PC-1), (w-PC-2), (w-PC-3) w.r.t. $\Gamma[S]$. By line 4 of `backtrack_ua-jump`(), $S = L^{\iota}_{\top}$ for some step $\iota < \iota_1$.
   – *(PC-1)*: By lines 2-3 of `scl_back`() (which was executed at line 13 of Algorithm 1, just before ι_2), it holds $\forall(u \in L^{\iota}_{\top})\ L^{\iota_1}_f[u] = \perp$. By induction hypothesis and by Item 1 of Proposition 8, the following holds:

$$\forall(v \in L^{\iota}_{\top})\ f^{c:\iota_2}(v) = f^*_{w'_{i,j^{\iota_1}}}(v) = f^*_{w'_{i,j^{\iota_2}-1}}(v).$$

   By Lemma 2, $f^*_{w'_{i,j^{\iota_2}-1}} \preceq f^*_{w'_{i,j^{\iota_2}}}$. Therefore, $\forall(v \in L^{\iota}_{\top})\ f^{c:\iota_2}(v) \preceq f^*_{w'_{i,j^{\iota_2}}}(v)$. Whence, (PC-1) holds w.r.t. $\Gamma[L^{\iota}_{\top}] = \Gamma[S]$.
   – *(w-PC-2):* By induction hypothesis and by Item 3 of Proposition 8, then:

$$L^{\text{inc}^{\iota_2}} = \{v \in V \mid 0 < f^{c:\iota_2}(v) \neq \top\}.$$

   Thus, by Lemma 1, $L^{\text{inc}^{\iota_2}} \subseteq \text{Inc}(f^{c:\iota_2}, i, j^{\iota_2})$. So, (w-PC-2) holds w.r.t. $\Gamma[L^{\iota}_{\top}] = \Gamma[S]$.
   – *(w-PC-3):* Let $u \in V \setminus L^{\text{inc}^{\iota_2}}$, and let $v \in N^{\text{out}}_{\Gamma[L^{\iota}_{\top}]}(v)$.

If $u \in V_0$, we need to check the state of $J.\mathtt{cmp}^{\iota_2}[(u,v)]$ and $J.\mathtt{cnt}^{\iota_2}[u]$.

1. If $J.\mathtt{cmp}^{\iota_2}[u,v] = \mathtt{F}$, we argue that $(u,v) \in E$ is incompatible w.r.t. $f^{c:\iota_2}$ in $\Gamma[S]_{i,j}$. Indeed, it was already $J.\mathtt{cmp}^{\iota_2}[(u,v)] = \mathtt{F}$ when the previous $\mathtt{J\text{-}VI}()$ (that invoked at step ι_1) halted. Then, by induction hypothesis and by Item 3 of Proposition 8, (u,v) is incompatible w.r.t. $f^{c:\iota_2}$ in $\Gamma[S]_{i,j-1}$. Thus, (u,v) is incompatible w.r.t. $f^{c:\iota_2}$ also in $\Gamma[S]_{i,j}$ (because $w'_{i,j} < w'_{i,j-1}$). So, $J.\mathtt{cmp}^{\iota_2}[(u,v)]$ is coherent w.r.t. $f^{c:\iota_2}$ in $\Gamma[S]_{i,j}$.
2. If $J.\mathtt{cmp}^{\iota_2}[(u,v)] = \mathtt{T}$, we argue that either (u,v) is compatible w.r.t. $f^{c:\iota_2}$ in $\Gamma[S]_{i,j}$ or it holds that $v \in L^{\text{inc}\,\iota_2}$. Indeed, assume that $J.\mathtt{cmp}^{\iota_2}[(u,v)] = \mathtt{T}$ and that (u,v) is incompatible w.r.t. $f^{c:\iota_2}$ in $\Gamma[S]_{i,j}$. Since $J.\mathtt{cmp}^{\iota_2}[(u,v)] = \mathtt{T}$, then it was as such even when the previous $\mathtt{J\text{-}VI}()$ (that invoked at step ι_1) halted. So, (u,v) was compatible w.r.t. $f^{c:\iota_2}$ in $\Gamma[S]_{i,j-1}$. Since $u \notin L^{\text{inc}\,\iota_2}$, then $f^{c:\iota_2}(u) = 0$ (indeed, if $f^{c:\iota_2}(u) = \top$, then (u,v) would have been compatible). Therefore, it is not possible that $f^{c:\iota_2}(v) = 0$; since, otherwise, from the fact that $f^{c:\iota_2}(u) = f^{c:\iota_2}(v) = 0$ and (u,v) is compatible w.r.t. $f^{c:\iota_2}$ in $\Gamma[S]_{i,j-1}$, it would be $w'(u,v)_{i,j-1} \geq 0$; and since $w(u,v) \in \mathbb{Z}$ and $0 < F_{j-1} < F_j \leq 1$ where $j \in [2, s-1]$, it would be $w'_{i,j}(u,v) \geq 0$ as well, so (u,v) would be compatible w.r.t. f^{c} in $\Gamma[S]_{i,j}$. Also, it is not possible that $f^{c:\iota_2}(v) = \top$, since otherwise (u,v) would have been incompatible w.r.t. $f^{c:\iota_2}$ in $\Gamma[S]_{i,j-1}$ (because $f^{c:\iota_2}(u) = 0$). Therefore, $0 < f^{c:\iota_2}(v) < \top$. Then, induction hypothesis and by Item 3 of Proposition 8, $v \in L^{\text{inc}\,\iota_2}$.
3. By induction hypothesis and by Proposition 8, $f^{c:\iota_2}$ is the least-SEPM of $\Gamma[S]_{i,j-1}$ and $J.\mathtt{cnt}^{\iota_2}$, $J.\mathtt{cmp}^{\iota_2}$ are both coherent w.r.t. $f^{c:\iota_2}$ in $\Gamma[S]_{i,j-1}$. Therefore,

$$\begin{aligned} J.\mathtt{cnt}^{\iota_2}[u] &= \left|\left\{v \in N^{\text{out}}_{\Gamma[S]}(u) \mid f^{c:\iota_2}(u) \succeq f^{c:\iota_2}(v) \ominus w'_{i,j-1}(u,v)\right\}\right| && \text{[by coherency of } J.\mathtt{cnt}^{\iota_2}] \\ &= \left|\left\{v \in N^{\text{out}}_{\Gamma[S]}(u) \mid J.\mathtt{cmp}^{\iota_2}[(u,v)] = \mathtt{T}\right\}\right|. && \text{[by coherency of } J.\mathtt{cmp}^{\iota_2}] \end{aligned}$$

Moreover, since $f^{c:\iota_2}$ is least-SEPM of $\Gamma[S]_{i,j-1}$, then u is consistent w.r.t. $f^{c:\iota_2}$ in $\Gamma[S]_{i,j-1}$, thus $J.\mathtt{cnt}^{\iota_2}[u] > 0$.

If $u \in V_1$, assume (u,v) is incompatible w.r.t. $f^{c:\iota_2}$ in $\Gamma[S]_{i,j}$, for some $v \in N^{\text{out}}_{\Gamma[S]}(u)$.

We want to prove $v \in L^{\text{inc}\,\iota_2}$. By induction hypothesis and by Proposition 8, $f^{c:\iota_2}$ is the least-SEPM of $\Gamma[S]_{i,j-1}$ and $L^{\text{inc}\,\iota_2} = \{q \in V \mid 0 < f^{c:\iota_2}(q) \neq \top\}$. Thus, since $v \in V_1$, (u,v) is compatible w.r.t. $f^{c:\iota_2}$ in $\Gamma[S]_{i,j-1}$. Moreover, since $u \notin L^{\text{inc}\,\iota_2}$ by hypothesis, then $f^{c:\iota_2}(u) = 0$ or $f^{c}(u) = \top$; but since (u,v) is incompatible w.r.t. $f^{c:\iota_2}$ in $\Gamma[S]_{i,j}$, it is $f^{c:\iota_2}(u) = 0$. Therefore, it is not possible that $f^{c:\iota_2}(v) = 0$; since, otherwise, from the fact that $f^{c:\iota_2}(u) = f^{c:\iota_2}(v) = 0$ and (u,v) is compatible w.r.t. $f^{c:\iota_2}$ in $\Gamma[S]_{i,j-1}$, it would be:

$$w'(u,v)_{i,j-1} = w(u,v) - i - F_{j-1} \geq 0,$$

since $w(u,v) \in \mathbb{Z}$ and $0 < F_{j-1} < F_j \leq 1$ where $j \in [2, s-1]$, it would be $w'_{i,j}(u,v) \geq 0$, so (u,v) would have been compatible w.r.t. f^{c} in $\Gamma[S]_{i,j}$. Also, it is not possible that $f^{c:\iota_2}(v) = \top$, since otherwise (u,v) would have been incompatible w.r.t. $f^{c:\iota_2}$ in $\Gamma[S]_{i,j}$ (because $f^{c:\iota_2}(u) = 0$). Therefore, $0 < f^{c:\iota_2}(v) < \top$. Then, induction hypothesis and by Item 3 of Proposition 8, $v \in L^{\text{inc}\,\iota_2}$.

3. Assume that $\mathtt{ei\text{-}jump}(i, J)$ is invoked at line 7 of Algorithm 1, say at step ι_1, and that $L^{\text{inc}\,\iota_1} = \emptyset$. Then, the following properties hold.
– (eij-PC-1) $f^{c:\iota_1}$ is the least-SEPM of $\Gamma_{i,s-1}$: indeed, consider the previous invocation $\mathtt{J\text{-}VI}(i, j^{\iota_0}, F, J, \Gamma[S^{\iota_0}])$ at line 11 of Algorithm 1, say it was invoked at step ι_0 before ι_1. By induction hypothesis and by Item 1 of Proposition 8, $f^{c:\iota_1}$ is the least-SEPM of $\Gamma[S^{\iota_0}]_{i,j^{\iota_0}}$.

Since $L^{\mathrm{inc}^{\iota_1}} = \emptyset$ by assumption, by induction hypothesis and Item 3 of Proposition 8, then $\{v \in S^{\iota_0} \mid 0 < f^{\mathrm{c}:\iota_1}(v) \neq \top\} = L^{\mathrm{inc}^{\iota_1}} = \emptyset$. We claim that $\forall (u \in S^{\iota_0})\ f^{\mathrm{c}:\iota_1}(u) = \top$. Indeed, the following holds.

If $u \in V_0$, since $f^{\mathrm{c}:\iota_1}$ is the least-SEPM of $\Gamma[S^{\iota_0}]_{i,j^{\iota_0}}$, there exists $v \in N^{\mathrm{out}}_{\Gamma[S^{\iota_0}]}(u)$ such that (u,v) is compatible w.r.t. $f^{\mathrm{c}:\iota_1}$ in $\Gamma[S^{\iota_0}]_{i,j^{\iota_0}}$. So, it is not possible that $f^{\mathrm{c}:\iota_1}(u) = 0$: otherwise, it would be $f^{\mathrm{c}:\iota_1}(v) = 0$ as well (because either $f^{\mathrm{c}:\iota_1}(v) = 0$ or $f^{\mathrm{c}:\iota_1}(v) = \top$), and since $w(u,v) \in \mathbb{Z}$ and $0 < F_{j^{\iota_0}} \leq 1$ where $j^{\iota_0} \in [1, s-1]$, then (u,v) would be compatible w.r.t. $f^{\mathrm{c}:\iota_1}$ even in $\Gamma[S^{\iota_0}]_{i,s-1}$, thus $f^*_{w'_{i,s-1}}(u) = 0$. But this contradicts the fact that, by induction hypothesis, Item 1 of Proposition 10 and Item 1 of Proposition 8, $f^*_{w'_{i,s-1}}(u) = \top$. Therefore, $f^{\mathrm{c}:\iota_1}(u) = \top$.

If $u \in V_1$, since $f^{\mathrm{c}:\iota_1}$ is the least-SEPM of $\Gamma[S^{\iota_0}]_{i,j^{\iota_0}}$, for every $v \in N^{\mathrm{out}}_{\Gamma[S^{\iota_0}]}(u)$, the arc (u,v) is compatible w.r.t. $f^{\mathrm{c}:\iota_1}$ in $\Gamma[S^{\iota_0}]_{i,j^{\iota_0}}$. Now, by arguing as in the previous case (i.e., $u \in V_0$), it holds that $\forall (u \in S^{\iota_0})\ f^{\mathrm{c}:\iota_1}(u) = \top$.

Thus, $\forall (u \in S^{\iota_0})\ f^{\mathrm{c}:\iota_1}(u) = \top = f^*_{i,s-1}(u)$. By induction hypothesis and Item [3, (b), (ii)] of Proposition 10, $\forall (u \in V \setminus S^{\iota_0})\ f^{\mathrm{c}:\iota_1}(u) = f^*_{w'_{i,s-1}}(u)$. So, $f^{\mathrm{c}:\iota_1} = f^*_{w'_{i,s-1}}$.

– (eij-PC-2) $L^{\mathrm{inc}^{\iota_1}} = \{v \in S^{\iota_0} \mid 0 < f^{\mathrm{c}:\iota_1}(v) \neq \top\}$: this holds by induction hypothesis and by Item 3 of Proposition 8.

– (eij-PC-3) $L^{\mathrm{inc}\ \iota}_{\mathrm{cpy}} \subseteq \mathrm{Inc}(f^{\mathrm{c}:\iota}, i', j')$ for every $(i', j') > (i, s-1)$: this holds by induction hypothesis plus Item [3, (a)] of Proposition 10 and Lemma 1.

– (eij-PC-4): consider the previous invocation of $\texttt{J-VI}(i, j^{\iota_0}, F, J, \Gamma[S^{\iota_0}])$ at line 11 of Algorithm 1, say at step ι_0, just before ι_1. By induction hypothesis and by Item 2 of Proposition 8, for every $u \in V_0 \cap S^{\iota_0}$, $J.\texttt{cnt}^{\iota_1}[u]$ and $J.\texttt{cmp}^{\iota_1}[(u,\cdot)]$ are both coherent w.r.t. $f^{\mathrm{c}:\iota_1}$ in $\Gamma[S^{\iota_0}]_{i,j^{\iota_0}}$; also, for every $u \in V_0 \setminus S^{\iota_0}$, $J.\texttt{cnt}^{\iota_1}[u]$ and $J.\texttt{cmp}^{\iota_1}[(u,\cdot)]$ are both coherent w.r.t. $f^*_{w'_{i,s-1}}$ in $\Gamma_{i,s-1}$. Since (eij-PC-1) holds, then $f^{\mathrm{c}:\iota_1} = f^*_{w'_{i,s-1}}$. So, (eij-PC-4) holds.

□

Lemma 4. *Let $\hat{v} \in V$, assume $\texttt{val}^{\Gamma}(\hat{v}) = \hat{\imath} - F_{\hat{\jmath}-1}$, for some $\hat{\imath} \in [W^-, W^+]$ and $\hat{\jmath} \in [1, s-1]$. Then, eventually, Algorithm 1 invokes $\texttt{J-VI}(\hat{\imath}, \hat{\jmath}, F, J, \Gamma[S])$ at line 11, for some $S \subseteq V$.*

Proof. For the sake of contradiction, for any $S \subseteq V$, assume that $\texttt{J-VI}(\hat{\imath}, \hat{\jmath}, F, J, \Gamma[S])$ is never invoked at line 11 of Algorithm 1. At each iteration of the main `while` loop of Algorithm 1 (lines 6-14), j is incremented (line 14); meanwhile, the value of i stands still until (eventually) `ei-jump`() and (possibly) `ua-jumps`() increase it (also resetting $j \leftarrow 1$). Therefore, since $\texttt{J-VI}(\hat{\imath}, \hat{\jmath}, F, J, \Gamma[S])$ is never invoked at line 11, there are $i_0 \in [W^-, W^+]$ and $j_0 \in [1, s-1]$, where $(i_0, j_0) < (\hat{\imath}, \hat{\jmath})$, such that one of the following two hold:

- Either $\texttt{ei-jump}(i_0, J)$ (line 7) is invoked and, when it halts say at step h, it holds $J.i^h > \hat{\imath}$.

  In that case, by Item 1 of Proposition 11 and Item 2 of Proposition 9, $f^*_{\hat{\imath},\hat{\jmath}-1}(\hat{v}) = f^*_{\hat{\imath},\hat{\jmath}}(\hat{v})$. On the other hand, since $\texttt{val}^{\Gamma}(\hat{v}) = \hat{\imath} - F_{\hat{\jmath}-1}$, then $\hat{v} \in \mathscr{W}_0(\Gamma_{\hat{\imath},\hat{\jmath}-1}) \cap \mathscr{W}_1(\Gamma_{\hat{\imath},\hat{\jmath}})$ by Theorem 3; so, by Propositions 3, $f^*_{\hat{\imath},\hat{\jmath}-1}(\hat{v}) \neq \top$ and $f^*_{\hat{\imath},\hat{\jmath}}(\hat{v}) = \top$. So, $\top \neq f^*_{\hat{\imath},\hat{\jmath}-1}(\hat{v}) = \top$; this is absurd.

- Or $\texttt{ua-jumps}(i_0, s, F, J, \Gamma)$ (line 9) is invoked and, when it halts say at step h, $J.i^h > \hat{\imath}$.

In that case, during the execution of $\texttt{ua-jumps}(i_0, s, F, J, \Gamma)$, at some step $\hat{\iota}$, it is invoked $\texttt{J-VI}(\hat{i}, s-1, F, J, \Gamma)$ (line 2 of $\texttt{ua-jumps}()$); and when it halts, say at step $\hat{\iota}_h$, by Item 2 of Proposition 10 and by line 6 of $\texttt{ua-jumps}()$, then $L_\top^{\hat{\iota}_h} = \mathscr{W}_0(\Gamma_{\hat{i}-1,s-1}) \cap \mathscr{W}_1(\Gamma_{\hat{i},s-1}) = \emptyset$. Still, $\texttt{val}^\Gamma(\hat{v}) = \hat{i} - F_{\hat{j}-1}$, then $\hat{v} \in \mathscr{W}_0(\Gamma_{\hat{i},\hat{j}-1}) \cap \mathscr{W}_1(\Gamma_{\hat{i},\hat{j}})$ by Theorem 3. Since $\rho = \{w_{i,j}\}_{i,j}$ is monotone decreasing, then $\mathscr{W}_0(\Gamma_{\hat{i},\hat{j}-1}) \subseteq \mathscr{W}_0(\Gamma_{\hat{i}-1,s-1})$ and $\mathscr{W}_1(\Gamma_{\hat{i},\hat{j}}) \subseteq \mathscr{W}_1(\Gamma_{\hat{i},s-1})$. Then, $\hat{v} \in \mathscr{W}_0(\Gamma_{\hat{i},\hat{j}-1}) \cap \mathscr{W}_1(\Gamma_{\hat{i},\hat{j}}) \subseteq \mathscr{W}_0(\Gamma_{\hat{i}-1,s-1}) \cap \mathscr{W}_1(\Gamma_{\hat{i},s-1}) = \emptyset$, but this is absurd.

In either case, we arrive at some contradiction.

Therefore, eventually, Algorithm 1 invokes $\texttt{J-VI}()$ at line 11 on input $(\hat{i}, \hat{j})$. □

Theorem 5. *Given any input MPG $\Gamma = (V, E, w, \langle V_0, V_1 \rangle)$, Algorithm 1 halts in finite time.*

If $(\mathscr{W}_0, \mathscr{W}_1, \nu, \sigma_0^*)$ is returned, then: $\mathscr{W}_0$ is the winning set of Player 0 in Γ, $\mathscr{W}_1$ is that of Player 1, $\forall^{v\in V}\, \nu(v) = \texttt{val}^\Gamma(v)$, σ_0^* is an optimal positional strategy for Player 0 in Γ.

Proof. Firstly, we argue that Algorithm 1 halts in finite time. Recall, by Propositions [8, 9, 10, 11], it holds that any invocation of $\texttt{J-VI}()$, $\texttt{ei-jump}()$, $\texttt{ua-jumps}()$ (respectively) halts in finite time. It is easy to check at this point that (by lines 14 of Algorithm 1, line 4 and 8 of $\texttt{ei-jump}()$, line 5 of $\texttt{ua-jumps}()$, and since L_ω was sorted in increasing order), whenever $\texttt{J-VI}()$ is invoked at line 11 of Algorithm 1 – at any two consequential steps ι_0, ι_1 (i.e., such that $\iota_0 < \iota_1$) – then $(i^{\iota_0}, j^{\iota_0}) < (i^{\iota_1}, j^{\iota_1})$. Also, by Propositions 8-11, whenever $\texttt{J-VI}()$ is invoked at line 11 of Algorithm 1, if it halts say at step ι_h, then $f^{c:\iota_h}$ is the least-SEPM of $\Gamma_{i^{\iota_h}, j^{\iota_h}}$; so, eventually, say when $(i^{\hat{\iota}_h}, j^{\hat{\iota}_h})$ are sufficiently large, then $\forall^{u\in V}\, f^{c:\hat{\iota}_h}(u) = \top$; and, by Item 3 of Proposition 8, $L^{\mathrm{inc}^{\hat{\iota}_h}} = \{v \in V \mid 0 < f^{c:\hat{\iota}_h} \neq \top\}$, so, $L^{\mathrm{inc}^{\hat{\iota}_h}} = \emptyset$. Consider the first invocation of $\texttt{ei-jump}()$ (line 7 of Algorithm 1) that is made soon after this $\hat{\iota}_h$, and say it halts at step h. By Item 3 of Proposition 9, $L^{\mathrm{inc}^h} = \mathrm{Inc}(f^{c:h}, J.i^h, 1)$. Since $\forall^{u\in V}\, f^{c:\hat{\iota}_h}(u) = f^{c:h}(u) = \top$, then $\mathrm{Inc}(f^{c:h}, J.i^h, 1) = \emptyset$. Therefore, $L^{\mathrm{inc}^h} = \emptyset$. Therefore, Algorithm 1 halts at line 8 soon after h.

Secondly, we argue that Algorithm 1 returns $(\mathscr{W}_0, \mathscr{W}_1, \nu, \sigma_0^*)$ correctly.

On one side, $\mathscr{W}_0, \mathscr{W}_1, \nu, \sigma_0^*$ are accessed only when $\texttt{set\_vars}()$ is invoked (line 12 of Algorithm 1). Just before that, at line 11, some $\texttt{J-VI}()$ must have been invoked; say it halts at step h. By Items 1 and 2 of Proposition 10, $L_\top^h = \mathscr{W}_0(\Gamma_{i^h, j^h-1}) \cap \mathscr{W}_1(\Gamma_{i^h, j^h})$. Therefore, by Theorem 3, ν is assigned correctly; so, $\mathscr{W}_0, \mathscr{W}_1$ are also assigned correctly. At this point, by Theorem 4, also σ_0^* is assigned correctly.

Conversely, let $\hat{v} \in V$ and assume $\texttt{val}^\Gamma(v) = \hat{i} - F_{\hat{j}-1}$ for some $\hat{i} \in [W^-, W^+]$ and $\hat{j} \in [1, s-1]$. By Lemma 4, eventually, Algorithm 1 invokes $\texttt{J-VI}(\hat{i}, \hat{j}, F, J, \Gamma)$ at line 11. By Items 1 and 2 of Proposition 10, when $\texttt{J-VI}(\hat{i}, \hat{j}, F, J, \Gamma)$ halts, say at step h, it holds that $L_\top^h = \mathscr{W}_0(\Gamma_{\hat{i},\hat{j}-1}) \cap \mathscr{W}_1(\Gamma_{\hat{i},\hat{j}})$. Therefore, soon after at line 12, $\texttt{set\_vars}()$ assigns to $\mathscr{W}_0, \mathscr{W}_1, \nu, \sigma_0^*$ a correct state. □

4.3. Complexity of Algorithm 1

The complexity of Algorithm 1 follows, essentially, from the fact that [*Inv-EI*] is satisfed.

Proposition 12. *Algorithm 1 satisfies [*Inv-EI*]: whenever a Scan-Phase is executed (each time that a Value-Iteration is invoked), an energy-level $f(v)$ strictly increases for at least one $v \in V$. So, the energy-lifting operator δ is applied (successfully) at least once per each $\texttt{J-VI}()$.*

Proof. By lines 1 and 9 of `ei-jump`() (SubProcedure 6), lines 1-6 of `ua-jumps`(), and line 8 of Algorithm 1, whenever `J-VI`() is invoked either at line 11 of Algorithm 1 or at line 2 of `ua-jumps`() (SubProcedure 7), say at step ι, then $L^{\text{inc}^{\iota}} \neq \emptyset$. Moreover, by Proposition 11, by Item 3 of Propositions 8 and Lemma 1, $L^{\text{inc}^{\iota}} \subseteq \text{Inc}(f^{\text{c}:\iota}, i^{\iota}, j^{\iota})$. Therefore, during each `J-VI`() that is possibly invoked by Algorithm 1, at least one application of δ is performed (line 2-3 of `J-VI()`) (because $L^{\text{inc}^{\iota}} \neq \emptyset$) and every single application of $\delta(f^{\text{c}}, v)$ that is made during `J-VI`(), say at step $\hat{\iota}$, for any $v \in V$, really increases $f^{\text{c}:\hat{\iota}}(v)$ (because $L^{\text{inc}^{\hat{\iota}}} \subseteq \text{Inc}(f^{\text{c}:\hat{\iota}}, i^{\hat{\iota}}, j^{\hat{\iota}})$). □

Theorem 6. *Given an input MPG $\Gamma = (V, E, w, \langle V_0, V_1 \rangle)$, Algorithm 1 halts within the following time bound:*

$$O(|E| \log |V|) + \Theta\Big(\sum_{v \in V} deg_{\Gamma}(v) \cdot \ell^{1}_{\Gamma}(v)\Big) = O(|V|^2 |E| W),$$

The working space is $\Theta(|V| + |E|)$.

Proof. The initialization of L_{ω} takes $O(|E| \log |V|)$ time, i.e., the cost for sorting $\{w_e \mid e \in W\}$. Each single application of δ, which is possibly done during any execution of `J-VI`() throughout Algorithm 1, it takes time $\Theta(\texttt{deg}_{\Gamma}(v))$. Indeed, by Proposition 5, the total aggregate time spent for all applications of δ in Algorithm 1 is $\Theta\big(\sum_{v \in V} \texttt{deg}_{\Gamma}(v) \cdot \ell^{1}_{\Gamma}(v)\big)$. It is not difficult to check from the description of Algorithm 1, at this point, that the time spent between any two subsequent applications of δ can increase the total time amount $\sum_{v \in V} \texttt{deg}_{\Gamma}(v) \cdot \ell^{1}_{\Gamma}(v)$ of Algorithm 1 only by a constant factor. Notice that the aggregate total cost of all the invocations of `repair`() is $O(|E|)$. In Section 3 it was shown how to generate $\mathscr{F}_{|V|}$ iteratively, one term after another, in $O(1)$ time-delay and $O(1)$ total space, as in Pawlewicz and Pătraşcu (2009). It is also easy to check, at this point, that Algorithm 1 works with $\Theta(|V| + |E|)$ space. □

4.4. An Experimental Evaluation of Algorithm 1

This section describes an empirical evaluation of Algorithm 1. All algorithms and procedures employed in this practical evaluation have been implemented in C/C++ and executed on a Linux machine having the following characteristics:

– Intel Core i5-4278U CPU @ 2.60GHz x2;
– 3.8GB RAM;
– Ubuntu 15.10 Operating System.

Source codes and scripts are (will be soon, w.r.t. time of submission) fully available online.

The main goal of this experiment was: (i) to determine the average computation time of Algorithm 1, with respect to randomly-generated MPGs, in order to give an idea of the practical behavior it; (ii) to offer an experimental comparison between Algorithm 1 and the algorithm which is offered in Comin and Rizzi (2016), i.e., Algorithm 0, in order to give evidence and experimental confirmation of the algorithmic improvements made over Comin and Rizzi (2016). Here we propose a summary of the obtained results presenting a brief report about, Test 1, Test 2.

In all of our tests, in order to generate a suitable dataset of MPGs, our choice has been to use the `randomgame` procedure of `pgsolver` suite (pgsolver, 2013), that can produce random arenas instances for any given number of nodes. We exploited `randomgame` as follows:

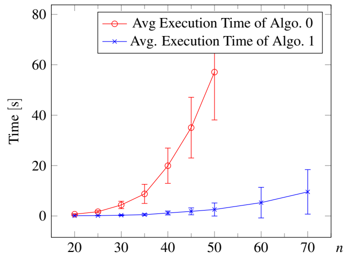
1. First, `randomgame` was used to generate random directed graphs, with out-degree taken uniformly at random in $[1, |V|]$;

$\|V\|$	μ (sec)	σ
20	0.69	0.21
25	1.69	0.43
30	4.37	1.47
35	8.77	3.79
40	19.95	6.99
45	35.06	12.0
50	57.10	18.9

(a) Test 1, μ, Algo-0.

$\|V\|$	μ (sec)	σ
20	0.09	0.05
25	0.13	0.07
30	0.30	0.24
35	0.52	0.39
40	1.18	0.77
45	1.83	1.37
50	2.56	2.59
60	5.30	6.08
70	9.57	8.83

(b) Test 1, μ, Algo-1.



(c) Interpolation of average execution times in Test 1 for Algo-0 (red, mark=o) and Algo-1 (blue, mark=x).

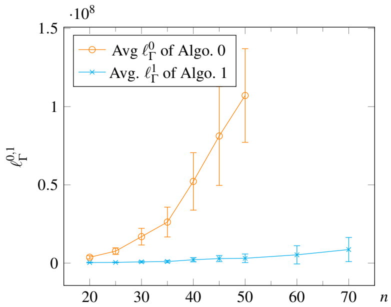
Figure 3: Results of Test 1 on Average Execution Time

$\|V\|$	μ (ℓ^0_Γ)	σ
20	3.71E+06	1.05E+06
25	7.67E+06	2.09E+06
30	1.69E+07	5.35E+06
35	2.62E+07	9.52E+06
40	5.22E+07	1.84E+07
45	8.12E+07	3.16E+07
50	1.07E+08	2.99E+07

(a) Test 1, ℓ^0_Γ, Algo-0.

$\|V\|$	μ (ℓ^1_Γ)	σ
20	3.53E+05	2.04E+05
25	4.45E+05	2.17E+05
30	7.92E+05	5.83E+05
35	9.91E+05	7.53E+05
40	2.13E+06	1.34E+06
45	2.88E+06	1.98E+06
50	3.08E+06	2.72E+06
60	5.28E+06	5.82E+06
70	8.66E+06	7.71E+06

(b) Test 1, ℓ^1_Γ, Algo-1.



(c) Interpolation of avgerage values of $\ell^0_\Gamma, \ell^1_\Gamma$ in Test 1 for Algo-0 (orange, mark=o) and Algo-1 (cyan, mark=x).

Figure 4: Results of Test 1 on $\ell^0_\Gamma, \ell^1_\Gamma$

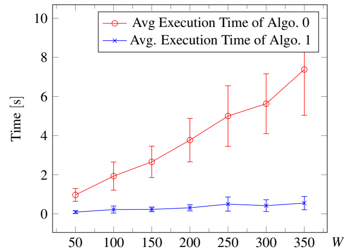
2. Then, the resulting graphs were translated into MPGs by weighting each arc with an integer randomly chosen in the interval $[-W, W]$, where W was chosen accordingly to the test type;

W	μ (sec)	σ
50	0.97	0.33
100	1.93	0.72
150	2.66	0.80
200	3.77	1.11
250	5.00	1.55
300	5.63	1.53
350	7.38	2.34

(a) Test 2, μ, Algo-0.

W	μ (sec)	σ
50	0.09	0.07
100	0.22	0.18
150	0.23	0.12
200	0.31	0.16
250	0.50	0.36
300	0.42	0.30
350	0.55	0.34

(b) Test 2, μ, Algo-1.



(c) Interpolation of average execution times in Test 2 for Algo-0 (red, mark=o) and Algo-1 (blue, mark=x).

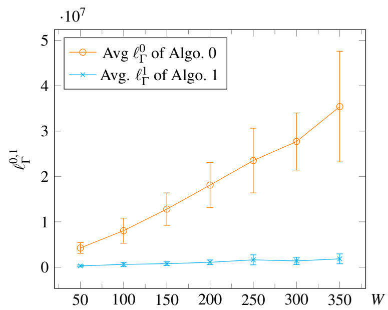
Figure 5: Results of Test 2 on Average Execution Time

W	μ (ℓ^0_Γ)	σ
50	4.24E+06	1.18E+06
100	8.05E+06	2.77E+06
150	1.28E+07	3.57E+06
200	1.81E+07	4.98E+06
250	2.35E+07	7.13E+06
300	2.77E+07	6.29E+06
350	3.54E+07	1.22E+07

(a) Test 2, ℓ^0_Γ, Algo-0.

W	μ (ℓ^1_Γ)	σ
50	2.84E+05	2.02E+05
100	6.21E+05	4.62E+05
150	7.75E+05	4.24E+05
200	1.08E+06	5.22E+05
250	1.62E+06	1.10E+06
300	1.38E+06	8.07E+05
350	1.84E+06	1.09E+06

(b) Test 2, ℓ^1_Γ, Algo-1.



(c) Interpolation of avgerage values of $\ell^0_\Gamma, \ell^1_\Gamma$ in Test 2 for Algo-0 (orange, mark=o) and Algo-1 (cyan, mark=x).

Figure 6: Results of Test 2 on $\ell^0_\Gamma, \ell^1_\Gamma$

With such settings, the resulting MPGs are characterized by $|V|$ and W.

In Test 1 the average computation time was determined for different orders of $|V|$. For each $n \in \{20, 25, 30, 35, 40, 45, 50\}$, 25 MPGs instances with maximum weight $W = 100$ were generated by `randomgame`. Each instance had been solved both with Algorithm 0 and Algorithm 1. In addition, to experiment a little further on Algorithm 1, for each $n \in \{60, 70\}$, 25 MPGs instances with maximum weight (fixed to) $W = 100$ were also generated by `randomgame` and solved only with Algorithm 1. The results of the test are summarized in Fig. 3, where each execution mean time is depicted as a point with a vertical bar representing its confidence interval determined according to its std-dev. As shown by Fig. 3, Test 1 gives experimental evidence of the supremacy of Algorithm 1 over Algorithm 0. In order to provide a better insight on the behavior of the algorithms, a comparison between the values of ℓ^0_Γ and ℓ^1_Γ is offered in Fig. 4. Test 1 confirms that $\ell^1_\Gamma \ll \ell^0_\Gamma$ (by a factor $\geq 10^2$ when $|V| \geq 50$) on randomly generated MPGs. The numerical results of Table 3a-3b suggest that the std-dev of both the avgerage running time of Algorithm 1 and of ℓ^1_Γ is greater (in proportion) than that of Algorithm 0 and ℓ^0_Γ; but thinking about it this actually turns out to be a benefit: as a certain proportion of MPGs instances can now exhibit quite a smaller value of ℓ^1_Γ, then the running time improves, but the std-dev fluctuates more meanwhile.

In Test 2 the average computation time was determined for different orders of W. For each $W \in \{50,100,150,200,250,300,350\}$, 25 MPGs instances with maximum weight W, and $|V| = 25$ (fixed), were generated by `randomgame`. Each instance had been solved both with Algorithm 0 and Algorithm 1. The results of the test are summarized in Fig. 5 and Fig. 6, where each execution mean time and $\ell^{0,1}_\Gamma$ is depicted as a point with a vertical bar representing its confidence interval determined according to its std-dev.

In summary our experiments suggest that, even in practice, Algorithm 1 is significantly faster than the Algorithm 0 devised in (Comin and Rizzi, 2015, 2016).

5. An Energy-Lattice Decomposition of $\texttt{opt}_\Gamma\Sigma^M_0$

Recall the example arena Γ_{ex} shown in Fig. 1. It is easy to see that $\forall^{v\in V}\texttt{val}^{\Gamma_{\text{ex}}}(v) = -1$. Indeed, Γ_{ex} contains only two cycles, i.e., $C_L = [A,B,C,D]$ and $C_R = [F,G]$, also notice that $w(C_L)/C_L = w(C_R)/C_R = -1$. The least-SEPM f^* of the reweighted EG Γ^{w+1}_{ex} can be computed by running a Value Iteration (Brim et al., 2011). Taking into account the reweighting $w \rightsquigarrow w+1$, as in Fig. 7: $f^*(A) = f^*(E) = f^*(G) = 0$, $f^*(B) = f^*(D) = f^*(F) = 4$, and $f^*(C) = 8$.

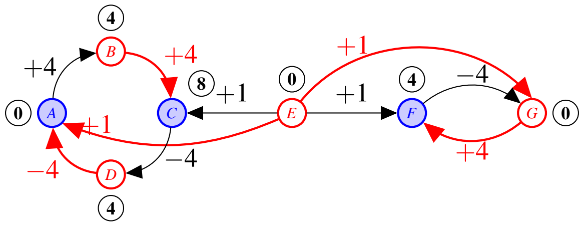


Figure 7: The least-SEPM f^* of Γ^{w+1}_{ex} (energy-levels are depicted in circled boldface). All and only those arcs of Player 0 that are compatible with f^* are $(B,C),(D,A),(E,A),(E,G),(G,F)$ (thick red arcs).

So Γ_{ex} (Fig. 7) implies the following.

Proposition 13. *The converse statement of Theorem 4 does not hold. There exist infinitely many MPGs Γ having at least one $\sigma_0 \in \texttt{opt}_\Gamma\Sigma^M_0$ which is not compatible with the least-SEPM of Γ.*

Proof. Consider the Γ_{ex} of Fig. 7, and the least-SEPM f^* of the EG Γ^{w+1}_{ex}. The only vertex at which Player 0 really has a choice is E. Every arc going out of E is optimal in the MPG Γ_{ex}: whatever arc $(E,X) \in \mathsf{E}$ (for any $X \in \{A,C,F,G\}$) Player 0 chooses at E, the resulting payoff equals $\texttt{val}^{\Gamma_{\text{ex}}}(E) = -1$. Let f^* be the least-SEPM of f^* in Γ^{w+1}_{ex}. Observe, (E,C) and

(E,F) are not compatible with f^* in Γ_{ex}^{w+1}, only (E,A) and (E,G) are. For instance, the positional strategy $\sigma_0 \in \Sigma_0^M$ defined as $\sigma_0(E) \triangleq F$, $\sigma_0(B) \triangleq C$, $\sigma_0(D) \triangleq A$, $\sigma_0(G) \triangleq F$ ensures a payoff $\forall^{v\in V}\texttt{val}^{\Gamma_{\text{ex}}}(v) = -1$, but it is not compatible with the least-SEPM f^* of Γ_{ex}^{w+1} (because $f^*(E) = 0 < 3 = f^*(F) \ominus w(E,F)$). It is easy to turn the Γ_{ex} of Fig. 7 into a family on infinitely many similar examples. □

We now aim at strengthening the relationship between $\text{opt}_\Gamma\Sigma_0^M$ and the Energy-Lattice $\mathscr{E}_\Gamma$. For this, we assume *wlog* $\exists^{\nu\in\mathbb{Q}}\forall^{v\in V}\texttt{val}^\Gamma(v) = \nu$. This follows from Theorem 1, which allows one to partition Γ into several domains $\Gamma_i \triangleq \Gamma_{|C_i}$ each one satisfying: $\exists^{\nu_i\in\mathbb{Q}}\forall^{v\in C_i}\texttt{val}^{\Gamma_i}(v) = \nu_i$. By Theorem 1 we can study $\texttt{opt}_{\Gamma_i}\Sigma_0^M$, independently w.r.t. $\texttt{opt}_{\Gamma_j}\Sigma_0^M$ for $j \neq i$.

We say that an MPG Γ is ν*-valued iff* $\exists^{\nu\in\mathbb{Q}}\forall^{v\in V}\texttt{val}^\Gamma(v) = \nu$.

Given an MPG Γ and $\sigma_0 \in \Sigma_0^M(\Gamma)$, recall, $G(\Gamma,\sigma_0) \triangleq (V,E',w')$ is obtained from G^Γ by deleting all and only those arcs that are not part of σ_0, i.e.,

$$E' \triangleq \big\{(u,v) \in E \mid u \in V_0 \text{ and } v = \sigma_0(u)\big\} \cup \big\{(u,v) \in E \mid u \in V_1\big\},$$

where each $e \in E'$ is weighted as in Γ, i.e., $w' : E' \to \mathbb{Z} : e \mapsto w_e$.

When $G = (V,E,w)$ is a weighted directed graph, a *feasible-potential (FP)* for G is any map $\pi : V \to \mathscr{C}_G$ such that $\forall^{u\in V}\forall^{v\in N^{\text{out}}(u)}\pi(u) \succeq \pi(v) \ominus w(u,v)$. The *least*-FP $\pi^* = \pi_G^*$ is the (unique) FP such that, for any other FP π, it holds $\forall^{v\in V}\pi^*(v) \preceq \pi(v)$. Given G, the Bellman-Ford algorithm can be used to produce π_G^* in $O(|V||E|)$ time. Let $\pi^*_{G(\Gamma,\sigma_0)}$ be the *least-FP* of $G(\Gamma,\sigma_0)$. Notice, for every $\sigma_0 \in \Sigma_0^M$, the least-FP $\pi^*_{G(\Gamma,\sigma_0)}$ is actually a SEPM for the EG Γ; still it can differ from the least-SEPM of Γ, due to σ_0. We consider the following family of strategies.

Definition 5 ($\Delta_0^M(f,\Gamma)$-Strategies)**.** *Let $\Gamma = \langle V,E,w,(V_0,V_1)\rangle$ and let $f : V \to \mathscr{C}_\Gamma$ be a SEPM for the EG Γ. Let $\Delta_0^M(f,\Gamma) \subseteq \Sigma_0^M(\Gamma)$ be the family of all and only those positional strategies of Player 0 in Γ such that $\pi^*_{G(\Gamma,\sigma_0)}$ coincides with f pointwisely, i.e.,*

$$\Delta_0^M(f,\Gamma) \triangleq \Big\{\sigma_0 \in \Sigma_0^M(\Gamma) \mid \forall^{v\in V}\, \pi^*_{G(\Gamma,\sigma_0)}(v) = f(v)\Big\}.$$

We now aim at exploring further on the relationship between $\mathscr{E}_\Gamma$ and $\texttt{opt}_\Gamma\Sigma_0^M$, via $\Delta_0^M(f,\Gamma)$.

Definition 6 (The Energy-Lattice of $\texttt{opt}_\Gamma\Sigma_0^M$)**.** *Let Γ be a ν-valued MPG. Let $\mathscr{X} \subseteq \mathscr{E}_{\Gamma^{w-\nu}}$ be a sub-lattice of SEPMs of the reweighted EG $\Gamma^{w-\nu}$.*

*We say that $\mathscr{X}$ is an "*Energy-Lattice *of $\texttt{opt}_\Gamma\Sigma_0^M$" iff $\forall^{f\in\mathscr{X}}\Delta_0^M(f,\Gamma^{w-\nu}) \neq \emptyset$ and the following disjoint-set decomposition holds:*

$$\texttt{opt}_\Gamma\Sigma_0^M = \bigsqcup_{f\in\mathscr{X}} \Delta_0^M(f,\Gamma^{w-\nu}).$$

Lemma 5. *Let Γ be a ν-valued MPG, and let $\sigma_0^* \in \texttt{opt}_\Gamma\Sigma_0^M$. Then, $G(\Gamma^{w-\nu},\sigma_0^*)$ is* conservative *(i.e., it contains* no *negative cycle).*

Proof. Let $C \triangleq (v_1\ldots,v_k,v_1)$ by any cycle in $G(\Gamma^{w-\nu},\sigma_0^*)$. Since we have $\sigma_0^* \in \texttt{opt}_\Gamma\Sigma_0^M$ and $\forall^{v\in V}\texttt{val}^\Gamma(v) = \nu$, thus $w(C)/k = \frac{1}{k}\sum_{i=1}^k w(v_i,v_{i+1}) \geq \nu$ (for $v_{k+1} \triangleq v_1$) by Proposition 1, so that, assuming $w' \triangleq w - \nu$, then: $w'(C)/k = \frac{1}{k}\sum_{i=1}^k \big(w(v_i,v_{i+1}) - \nu\big) = w(C)/k - \nu \geq \nu - \nu = 0$. □

Some aspects of the following Proposition 14 rely heavily on Theorem 4: the compatibility relation comes again into play. Moreover, we observe that Proposition 14 is equivalent to the following fact, which provides a sufficient condition for a positional strategy to be optimal. Consider a ν-valued MPG Γ, for some $\nu \in \mathbb{Q}$, and let $\sigma_0^* \in \texttt{opt}_\Gamma\Sigma_0^M$. Let $\hat{\sigma}_0 \in \Sigma_0^M(\Gamma)$ be any (not necessarily optimal) positional strategy for Player 0 in the MPG Γ. Suppose the following holds:

$$\forall^{v\in V}\, \pi^*_{G(\Gamma^{w-\nu},\hat{\sigma}_0)}(v) = \pi^*_{G(\Gamma^{w-\nu},\sigma_0^*)}(v).$$

Then, by Proposition 14, $\hat{\sigma}_0$ is an optimal positional strategy for Player 0 in the MPG Γ.

We are thus relying on the same *compatibility* relation between Σ_0^M and SEPMs in reweighted EGs which was at the *base* of Theorem 4, aiming at extending Theorem 4 so to describe the whole $\texttt{opt}_\Gamma\Sigma_0^M$ (and not just the join part of it).

Proposition 14. *Let the MPG Γ be ν-valued, for some $\nu \in \mathbb{Q}$.*
There is at least one Energy-Lattice of $opt_\Gamma\Sigma_0^M$:

$$\mathscr{B} \triangleq \{\pi^*_{G(\Gamma^{w-\nu},\sigma_0)} \mid \sigma_0 \in opt_\Gamma\Sigma_0^M\}.$$

Proof. The only non-trivial point to check being: $\bigsqcup_{f\in\mathscr{B}} \Delta_0^M(f,\Gamma^{w-\nu}) \subseteq \texttt{opt}_\Gamma\Sigma_0^M$.

For this, we shall rely on Theorem 4. Let $\hat{f} \in \mathscr{B}$ and $\hat{\sigma}_0 \in \Delta_0^M(\hat{f},\Gamma^{w-\nu})$ be fixed (arbitrarily). Since $\hat{f} \in \mathscr{B}$, then $\hat{f} = \pi^*_{G(\Gamma^{w-\nu},\sigma_0^*)}$ for some $\sigma_0^* \in \texttt{opt}_\Gamma\Sigma_0^M$. Therefore, the following holds:

$$\pi^*_{G(\Gamma^{w-\nu},\hat{\sigma}_0)} = \hat{f} = \pi^*_{G(\Gamma^{w-\nu},\sigma_0^*)}.$$

Clearly, $\hat{\sigma}_0$ is compatible with $\hat{f}$ in the EG $\Gamma^{w-\nu}$, because $\hat{f} = \pi^*_{G(\Gamma^{w-\nu},\hat{\sigma}_0)}$. By Lemma 5, since σ_0^* is optimal, then $G(\Gamma^{w-\nu},\sigma_0^*)$ is conservative. Therefore:

$$V_{\hat{f}} = V_{\pi^*_{G(\Gamma^{w-\nu},\sigma_0^*)}} = V.$$

Notice, $\hat{\sigma}_0$ satisfies exactly the hypotheses required by Theorem 4. Therefore, $\hat{\sigma}_0 \in \texttt{opt}_\Gamma\Sigma_0^M$. This proves (*).This also shows $\texttt{opt}_\Gamma\Sigma_0^M = \bigsqcup_{f\in\mathscr{B}} \Delta_0^M(f,\Gamma^{w-\nu})$, and concludes the proof. □

Proposition 15. *Let the MPG Γ be ν-valued, for some $\nu \in \mathbb{Q}$. Let $\mathscr{B}_1$ and $\mathscr{B}_2$ be two Energy-Lattices for $opt_\Gamma\Sigma_0^M$. Then, $\mathscr{B}_1 = \mathscr{B}_2$.*

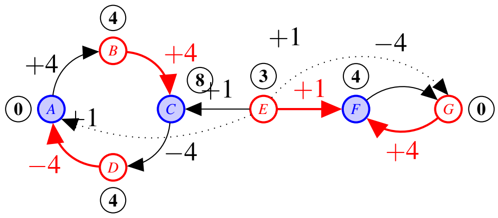
Proof. By symmetry, it is sufficient to prove that $\mathscr{B}_1 \subseteq \mathscr{B}_2$. Let $f_1 \in \mathscr{B}_1$ be fixed (arbitrarily). Then, $f_1 = \pi^*_{G(\Gamma^{w-\nu},\hat{\sigma}_0)}$ for some $\hat{\sigma}_0 \in \texttt{opt}_\Gamma\Sigma_0^M$. Since $\hat{\sigma}_0 \in \texttt{opt}_\Gamma\Sigma_0^M$ and since $\mathscr{B}_2$ is an Energy-Lattices, there exists $f_2 \in \mathscr{B}_2$ such that $\hat{\sigma}_0 \in \Delta_0^M(f_2,\Gamma^{w-\nu})$, which implies $\pi^*_{G(\Gamma^{w-\nu},\hat{\sigma}_0)} = f_2$. Thus, $f_1 = \pi^*_{G(\Gamma^{w-\nu},\hat{\sigma}_0)} = f_2$. This implies $f_1 \in \mathscr{B}_2$. □

The next theorem summarizes the main point of this section.

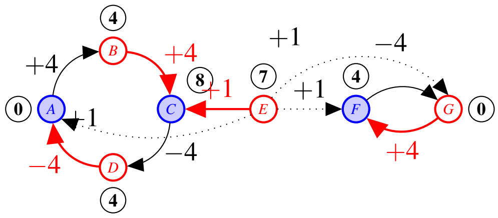
Theorem 7. *Let Γ be a ν-valued MPG, for some $\nu \in \mathbb{Q}$. Then, $\mathscr{B}^*_\Gamma \triangleq \{\pi^*_{G(\Gamma^{w-\nu},\sigma_0)} \mid \sigma_0 \in opt_\Gamma\Sigma_0^M\}$ is the unique Energy-Lattice of $opt_\Gamma\Sigma_0^M$.*

Proof. By Proposition 14 and Proposition 15. □

Example 1. *Consider the MPG Γ_{ex}, as defined in Fig. 1. Then, $\mathscr{B}^*_{\Gamma_{ex}} = \{f^*, f_1, f_2\}$, where f^* is the least-SEPM of the reweighted EG Γ^{w+1}_{ex}, and where the following holds: $f_1(A) = f_2(A) = f^*(A) = 0$; $f_1(B) = f_2(B) = f^*(B) = 4$; $f_1(C) = f_2(C) = f^*(C) = 8$; $f_1(D) = f_2(D) = f^*(D) = 4$; $f_1(F) = f_2(F) = f^*(F) = 4$; $f_1(G) = f_2(G) = f^*(G) = 0$; finally, $f^*(E) = 0$, $f_1(E) = 3$, $f_2(E) = 7$. An illustration of f_1 is offered in Fig. 8a (energy-levels are depicted in circled boldface). whereas f_2 is depicted in Fig. 8b. Notice that $f^*(v) \le f_1(v) \le f_2(v)$ for every $v \in V$, and this ordering relation is illustrated in Fig. 8.*



(a) The extremal-SEPM f_1 of Γ^{w+1}_{ex}



(b) The extremal-SEPM f_2 of Γ^{w+1}_{ex}.

Definition 7 (Extremal-SEPM)**.** *Each element $f \in \mathscr{B}^*_\Gamma$ is said to be an* extremal-*SEPM.*

The next lemma is the converse of Lemma 5.

Lemma 6. *Let the MPG Γ be ν-valued, for some $\nu \in \mathbb{Q}$. Consider any $\sigma_0 \in \Sigma^M_0(\Gamma)$, and assume that $G(\Gamma^{w-\nu}, \sigma_0)$ is conservative. Then, $\sigma_0 \in \mathtt{opt}_\Gamma \Sigma^M_0$.*

Proof. Let $C = (v_1, \ldots, v_\ell v_1)$ any cycle in $G(\Gamma, \sigma_0)$. Then, the following holds (if $v_{\ell+1} = v_1$): $\frac{w(C)}{\ell} = \frac{1}{\ell}\sum_{i=1}^{\ell} w(v_i, v_{i+1}) = \nu + \frac{1}{\ell}\sum_{i=1}^{\ell}\Big(w(v_i, v_{i+1}) - \nu\Big) \ge \nu$, where $\frac{1}{\ell}\sum_{i=1}^{\ell}\Big(w(v_i, v_{i+1}) - \nu\Big) \ge 0$ holds because $G(\Gamma^{w-\nu}, \sigma_0)$ is conservative. By Proposition 1, since $w(C)/\ell \ge \nu$ for every cycle C in $G^\Gamma_{\sigma_0}$, then $\sigma_0 \in \mathtt{opt}_\Gamma \Sigma^M_0$. □

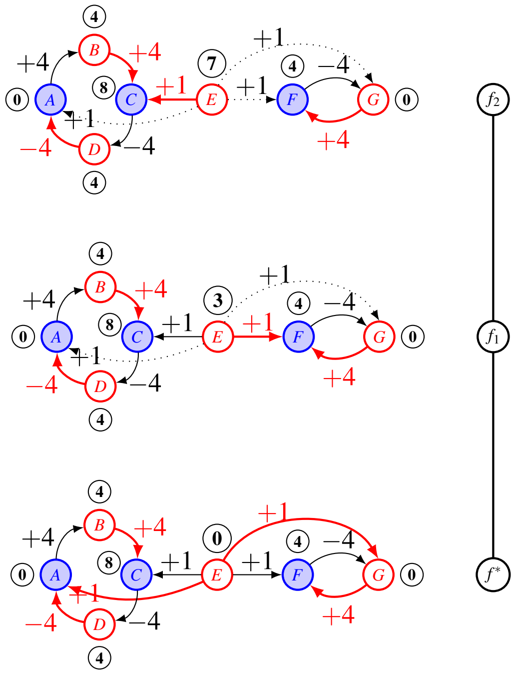
The following proposition asserts some properties of the Extremal-SEPMs.

Proposition 16. *Let the MPG Γ be ν-valued, for some $\nu \in \mathbb{Q}$. Let $\mathscr{B}^*_\Gamma$ be the Energy-Lattice of $\mathtt{opt}_\Gamma \Sigma^M_0$. Moreover, let $f : V \to \mathscr{C}_\Gamma$ be a SEPM for the reweighted EG $\Gamma^{w-\nu}$. Then, the following three properties are equivalent:*

1. *$f \in \mathscr{B}^*_\Gamma$;*
2. *There exists $\sigma_0 \in \mathtt{opt}_\Gamma \Sigma^M_0$ such that $\pi^*_{G(\Gamma^{w-\nu}, \sigma_0)}(v) = f(v)$ for every $v \in V$.*
3. *$V_f = \mathscr{W}_0(\Gamma^{w-\nu}) = V$ and $\Delta^M_0(f, \Gamma^{w-\nu}) \neq \emptyset$;*

(1 $\iff$ 2) Indeed, $\mathscr{B}^*_\Gamma = \{\pi^*_{G(\Gamma^{w-\nu}, \sigma_0)} \mid \sigma_0 \in \mathtt{opt}_\Gamma \Sigma^M_0\}$.

(1 $\Rightarrow$ 3) Assume $f \in \mathscr{B}^*_\Gamma$. Since (1 $\iff$ 2), there exist $\sigma_0 \in \mathtt{opt}_\Gamma \Sigma^M_0$ such that $\pi^*_{G(\Gamma^{w-\nu}, \sigma_0)} = f$. Thus, $\sigma_0 \in \Delta^M_0(f, \Gamma^{w-\nu})$, so that $\Delta^M_0(f, \Gamma^{w-\nu}) \neq \emptyset$. We claim $V_f = \mathscr{W}_0(\Gamma^{w-\nu}) = V$. Since $\forall (v \in V)\, \mathtt{val}^\Gamma(v) = \nu$, then $\mathscr{W}_0(\Gamma^{w-\nu}) = V$ by Proposition 2. Next, $G(\Gamma^{w-\nu}, \sigma_0)$ is conservative by Lemma 5. Since $G(\Gamma^{w-\nu}, \sigma_0)$ is conservative and $f = \pi^*_{G(\Gamma^{w-\nu}, \sigma_0)}$, then $V_f = V$. Therefore, $V_f = \mathscr{W}_0(\Gamma^{w-\nu}) = V$.



$\Delta_0^M(f_1, \Gamma^{w-\nu}) = \{\sigma_0^{(4)}\}$
$\sigma_0^{(4)}(B) = C$
$\sigma_0^{(4)}(D) = A$
$\sigma_0^{(4)}(E) = C$
$\sigma_0^{(4)}(G) = F$

$\Delta_0^M(f_1, \Gamma^{w-\nu}) = \{\sigma_0^{(3)}\}$
$\sigma_0^{(3)}(B) = C$
$\sigma_0^{(3)}(D) = A$
$\sigma_0^{(3)}(E) = F$
$\sigma_0^{(3)}(G) = F$

$\Delta_0^M(f^*, \Gamma^{w-\nu}) = \{\sigma_0^{(1)}, \sigma_0^{(2)}\}$	
$\sigma_0^{(1)}(B) = C$	$\sigma_0^{(2)}(B) = C$
$\sigma_0^{(1)}(D) = A$	$\sigma_0^{(2)}(D) = A$
$\sigma_0^{(1)}(E) = A$	$\sigma_0^{(2)}(E) = G$
$\sigma_0^{(1)}(G) = F$	$\sigma_0^{(2)}(G) = F$

Figure 8: The decomposition of $\texttt{opt}_\Gamma\Sigma_0^M$, for the MPG Γ_{ex}, which corresponds to the Energy-Lattice $\mathscr{B}^*_{\Gamma_{\text{ex}}} = \{f^*, f_1, f_2\}$ (computed in Example 1). Here, $f^* \leq f_1 \leq f_2$. This also brings a lattice $\mathscr{D}^*_{\Gamma_{\text{ex}}}$ of 3 sub-games of Γ_{ex}.

$(1 \Leftarrow 3)$ Since $\Delta_0^M(f, \Gamma^{w-\nu}) \neq \emptyset$, pick some $\sigma_0 \in \Delta_0^M(f, \Gamma^{w-\nu})$; so, $f = \pi^*_{G(\Gamma^{w-\nu}, \sigma_0)}$. Since $V_f = V$ and $f = \pi^*_{G(\Gamma^{w-\nu}, \sigma_0)}$, then $G(\Gamma^{w-\nu}, \sigma_0)$ is conservative. Since $G(\Gamma^{w-\nu}, \sigma_0)$ is conservative, then $\sigma_0 \in \texttt{opt}_\Gamma\Sigma_0^M$ by Lemma 6. Since $f = \pi^*_{G^*}$ and $\sigma_0 \in \texttt{opt}_\Gamma\Sigma_0^M$, then $f \in \mathscr{B}^*_\Gamma$ (as $2 \Rightarrow 1$).

6. A Recursive Enumeration of $\mathscr{B}^*_\Gamma$ and $\texttt{opt}_\Gamma(\Sigma_0^M)$

An enumeration algorithm for a set S provides an exhaustive listing of all the elements of S (without repetitions). As mentioned in Section 5, by Theorem 1, no loss of generality occurs if we assume Γ to be ν-valued for some $\nu \in \mathbb{Q}$. One run of Algorithm 1 allows one to partition an MPG Γ, into several domains $\Gamma_i \triangleq \Gamma_{|C_i}$ each one being ν_i-valued for $\nu_i \in S_\Gamma$; in $O(|V|^2|E|W)$ time and linear space. Still, by Proposition 13, Theorem 4 is not sufficient for enumerating the whole $\texttt{opt}_\Gamma(\Sigma_0^M)$ by means of Algorithm 1; it is enough only for $\Delta_0^M(f^*_\nu, \Gamma^{w-\nu})$ where f^*_ν is the least-SEPM of $\Gamma^{w-\nu}$, which is just the "join" component of $\texttt{opt}_\Gamma(\Sigma_0^M)$. However, we now have a more detailed description of $\texttt{opt}_\Gamma\Sigma_0^M$ in terms $\mathscr{B}^*_\Gamma$, thanks to Theorem 7.

This section offers a recursive enumeration of all the extremal-SEPMs, i.e., $\mathscr{B}^*_\Gamma$, and for computing the corresponding partitioning of $\texttt{opt}_\Gamma(\Sigma_0^M)$. In order to avoid duplicate elements in the enumeration, the algorithm needs to store a lattice $\mathscr{B}^*_\Gamma$ of sub-games of Γ, which is related to $\mathscr{X}^*_\Gamma$. *(T_Γ)*. We shall assume to dispose of a data-structure T_Γ supporting the following operations, given a sub-arena Γ' of Γ: $\texttt{insert}(\Gamma', T_\Gamma)$ stores Γ' into T_Γ; $\texttt{contains}(\Gamma', T_\Gamma)$ returns $\texttt{T}$ *iff* Γ' is in T_Γ and $\texttt{F}$ otherwise. A simple implementation of T_Γ goes by indexing $N^{\text{out}}_{\Gamma'}(v)$ for each $v \in V$.

This runs in $O(|V|^2)$ time, consuming $O(|E|)$ space per stored element. The same approach can be used to store and retrieve SEPMs in $O(|V|^2)$ time and $O(|V|)$ space.

The listing procedure is named enum(), it takes in input a ν-valued MPG Γ; going as follows.

1. Compute the least-SEPM f^* of Γ, and print Γ to output. Theorem 4 can be employed at this stage for enumerating $\Delta_0^M(f^*, \Gamma^{w-\nu})$: indeed, these are all and only those positional strategies lying in the *Cartesian* product of all the arcs $(u,v) \in E$ that are *compatible* with f^* in $\Gamma^{w-\nu}$ (because f^* is the least-SEPM of Γ).
2. Let $S_t \leftarrow \emptyset$ be an empty stack.
3. For each $\hat{u} \in V_0$, do the following:
   - Compute $E_{\hat{u}} \leftarrow \{(\hat{u},v) \in E \mid f^*(\hat{u}) \prec f^*(v) \ominus (w(\hat{u},v) - \nu)\}$; If $E_{\hat{u}} \neq \emptyset$, then:
     - Let $E' \leftarrow E_{\hat{u}} \cup \{(u,v) \in E \mid u \neq \hat{u}\}$ and $\Gamma' \leftarrow (V, E', w, \langle V_0, V_1 \rangle)$.
     - If contains$(\Gamma', T_\Gamma) = \texttt{F}$, do the following:
       * Compute the least-SEPM f'^* of $\Gamma'^{w-\nu}$;
       * If $V_{f'^*} = V$:
         – Push $\hat{u}$ on top of S_t and insert(Γ', T_Γ).
         – If contains$(f'^*, T_\Gamma) = \texttt{F}$, then insert$(f'^*, T_\Gamma)$ and print f'^*.
4. While $S_t \neq \emptyset$:
   - pop $\hat{u}$ from S_t; Let $E_{\hat{u}} \leftarrow \{(\hat{u},v) \in E \mid f^*(\hat{u}) \prec f^*(v) \ominus (w(\hat{u},v) - \nu)\}$, and $E' \leftarrow E_{\hat{u}} \cup \{(u,v) \in E \mid u \neq \hat{u}\}$, and $\Gamma' \leftarrow (V, E', w, \langle V_0, V_1 \rangle)$;
   - Make a *Recursive* call to enum() on input Γ'.

Down the recursion tree, when computing least-SEPMs, children Value-Iterations can amortize by starting from the energy-levels of the parent. The lattice of sub-games $\mathscr{B}^*_\Gamma$ comprises all and only those sub-games $\Gamma' \subseteq \Gamma$ that are eventually inserted into T_Γ at Step (3) of enum(); these are called the *basic* sub-games of Γ. The correctness of enum() follows by Theorem 7 and Theorem 4. In summary, we obtain Theorem 8.

Theorem 8. *There is a recursive algorithm for enumerating (w/o repetitions) all the elements of $\mathscr{B}^*_\Gamma$, on any input MPG Γ, with time* delay $O(|V|^3|E|W)$. *For this, the algorithm employs $O(|E||V|) + \Theta\big(|E||\mathscr{B}^*_\Gamma|\big)$ working space. The algorithm enumerates $\mathscr{X}^*_\Gamma$ (w/o repetitions) in* $O\big(|V|^3|E|W|\mathscr{B}^*_\Gamma|\big)$ total *time, and $O(|V||E|) + \Theta\big(|E||\mathscr{B}^*_\Gamma|\big)$ space.*

(Say $O(f(n))$ time *delay* when the time spent between any two consecutives is $O(f(n))$.)

To conclude we observe that $\mathscr{B}^*_\Gamma$ and $\mathscr{X}^*_\Gamma$ are not isomorphic as lattices, not even as sets (the cardinality of $\mathscr{B}^*_\Gamma$ can be greater that that of $\mathscr{X}^*_\Gamma$). Indeed, there is a surjective antitone mapping φ_Γ from $\mathscr{B}^*_\Gamma$ onto $\mathscr{X}^*_\Gamma$, (i.e., φ_Γ sends $\Gamma' \in \mathscr{B}^*_\Gamma$ to its least-SEPM $f^*_{\Gamma'} \in \mathscr{X}^*_\Gamma$); still, we can construct examples of MPGs such that $|\mathscr{B}^*_\Gamma| > |\mathscr{X}^*_\Gamma|$, i.e., φ_Γ is not into (in case of degeneracy).

7. Conclusion

We offered a faster $O(|E|\log|V|) + \Theta(\sum_{v\in V} \texttt{deg}_\Gamma(v) \cdot \ell(v)) = O(|V|^2|E|W)$ time energy algorithm for the Value Problem and Optimal Strategy Synthesis in MPGs. The result was achieved by introducing a novel scheme based on so called *Energy-Increasing* and *Unitary-Advance* Jumps.

In addition, we observed a unique complete decomposition of $\texttt{opt}_\Gamma\Sigma_0^M$ in terms of extremal-SEPMs in reweighted EGs, offering a pseudo-polynomial total-time recursive algorithm for enumerating (w/o repetitions) all the elements of $\mathscr{X}_\Gamma^*$, i.e., all the extremal-SEPMs, and for computing the components of the corresponding partitioning $\mathscr{B}_\Gamma^*$ of $\texttt{opt}_\Gamma\Sigma_0^M$.

It would be interesting to study further properties enjoyed by $\mathscr{B}_\Gamma^*$ and $\mathscr{X}_\Gamma^*$; also, we ask for more efficient algorithms for enumerating $\mathscr{X}_\Gamma^*$, e.g., we ask for pseudo-polynomial time delay and polynomial space enumerations. We also ask whether the least-SEPM of reweighted EGs of the kind Γ^{w-q}, for $q \in S_\Gamma$, can be computed in $O(|V||E|W)$ time, instead of $O(|V|^2|E|W)$: together with Algorithm 1, this could lead to an improved time complexity upper bound for MPGs (i.e., matching the time spent for solving EGs). To conclude, it would be very interesting to adapt Algorithm 1 to work with the strategy-improvement framework, instead of the value-iteration, as it seems to exhibit a faster converge in practice.

Many questions remain open on this way.

Acknowledgments. This work was supported by *Department of Computer Science, University of Verona, Verona, Italy*, under PhD grant "Computational Mathematics and Biology", on a co-tutelle agreement with *LIGM, Université Paris-Est in Marne-la-Vallée, Paris, France*.

References

A. Ehrenfeucht, J. Mycielski, Positional strategies for mean payoff games, International Journal of Game Theory 8 (2) (1979) 109–113.

U. Zwick, M. Paterson, The Complexity of Mean Payoff Games on Graphs, Theoretical Computer Science 158 (1996) 343–359.

M. Jurdziński, Deciding the winner in parity games is in $\texttt{UP} \cap \texttt{co-UP}$, Information Processing Letters 68 (3) (1998) 119 – 124.

V. Gurvich, A. Karzanov, L. Khachiyan, Cyclic games and an algorithm to find minimax cycle means in directed graphs, USSR Computational Mathematics and Mathematical Physics 28 (5) (1988) 85 – 91.

A. Chakrabarti, L. de Alfaro, T. Henzinger, M. Stoelinga, Resource Interfaces, in: R. Alur, I. Lee (Eds.), Embedded Software, vol. 2855 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 117–133, 2003.

P. Bouyer, U. Fahrenberg, K. Larsen, N. Markey, J. Srba, Infinite Runs in Weighted Timed Automata with Energy Constraints, in: F. Cassez, C. Jard (Eds.), Formal Modeling and Analysis of Timed Systems, vol. 5215 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 33–47, 2008.

L. Brim, J. Chaloupka, L. Doyen, R. Gentilini, J. Raskin, Faster Algorithms for Mean-Payoff Games, Formal Methods in System Design 38 (2) (2011) 97–118.

C. Comin, R. Rizzi, Energy Structure and Improved Complexity Upper Bound for Optimal Positional Strategies in Mean Payoff Games, in: 3rd International Workshop on Strategic Reasoning, vol. 20, 2015.

C. Comin, R. Rizzi, Improved Pseudo-polynomial Bound for the Value Problem and Optimal Strategy Synthesis in Mean Payoff Games, Algorithmica (February, 2016) 1–27.

D. Andersson, S. Vorobyov, Fast algorithms for monotonic discounted linear programs with two variables per inequality, Tech. Rep., Preprint NI06019-LAA, Isaac Netwon Institute for Mathematical Sciences, Cambridge, UK, 2006.

Y. Lifshits, D. Pavlov, Potential theory for mean payoff games, Journal of Mathematical Sciences 145 (3) (2007) 4967–4974.

H. Björklund, S. Sandberg, S. Vorobyov, Memoryless determinacy of parity and mean payoff games: a simple proof, Theoretical Computer Science 310 (1–3) (2004) 365 – 378.

R. Graham, D. Knuth, O. Patashnik, Concrete Mathematics: A Foundation for Computer Science, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edn., 1994.

J. Pawlewicz, M. Pătraşcu, Order Statistics in the Farey Sequences in Sublinear Time and Counting Primitive Lattice Points in Polygons, Algorithmica 55 (2) (2009) 271–282.

pgsolver, The PGSolver Collection of Parity Game Solvers, https://github.com/tcsprojects/pgsolver, 2008-2013.