
**UNIVERSITÀ DEGLI STUDI DI UDINE**

Dipartimento di Scienze Matematiche, Informatiche e Fisiche

Tesina di fine anno

# Distributed interval synchronization on directed graphs

**RELATORE**
**Prof. Andrea Fusiello**

**TUTOR**
**Prof. Pietro Corvaja**

**STUDENTI**
**Cristian Curaba**
**Filippo Olivetti**

**Anno Accademico 2021/2022**

# Contents

# Abstract

In this thesis we will introduce the synchronization problem on $(\mathbb{R}, +)$ and $(\mathbb{R}\setminus\{0\}, \cdot)$ (Section 1). In a real world scenario, measurements are obtained with uncertainty and, for computing calculus with error propagation, we will use the interval arithmetic (Section 2). From this point of view, we want to generalize the synchronization problem (which is usually performed on groups) admitting interval arithmetic. This is problematic due to the weak algebraic structure of $(\mathbb{IR}, +)$ and $(\mathbb{IR}_{>0}, \cdot)$: they both are commutative monoids. In Section 3 we generalize the distributed interval synchronization algorithm, described in [1], to directed graphs for both the $(\mathbb{IR}, +)$ and $(\mathbb{IR}_{>0}, \cdot)$ monoids. The pseudo-codes and the effectiveness evaluation are described in the last section.

# Preliminaries

In this section we review some useful concepts of graph theory and set the notation. A graph is a pair $G = (V, E)$ where $V$ is a finite set and $E$ is a family of pairs of elements of $V$ . If the pairs are ordered, then $G$ is called a directed graph (or digraph), otherwise it is called an undirected graph. The elements of $V$ are called vertices or nodes, and the elements of $E$ are called edges. We use $n$ and $m$ to denote the number of vertices and edges respectively, namely $n = |V|$ and $m = |E|$. Note that every directed graph can be turned into an undirected graph by ignoring the orientation of the edges, and every undirected graph can be turned into a directed graph by orienting the edges arbitrarily. A weighted graph is a graph together with a weight function $\omega \colon E \to \mathbb{R}^+$ . If the graph is unweighted, we set $\omega \colon E \to \{1\}$ and call $\omega$ the uniform weight function.

An edge $e = (v, w)$ is said to be incident to both $v$ and $w$. If G is undirected, then $v$ and $w$ are called the endpoints of $e$. If $G$ is directed, then $v$ and $w$ are called the tail and the head of $e$, respectively, and $e$ is said to leave $v$ and enter $w$. An edge of the form $(v, v)$ is called a loop.

We will consider simple graphs, i.e., graphs where there is at most one edge connecting a pair of noes.

A subgraph $G' = (V', E')$ of $G$ is a graph with $V' \subseteq V$ and $E' \subset E$. If $E'$ is a subset of $E$, then $G \setminus E'$ denotes the graph obtained by removing all the edges in $E'$ from $G$. If $V'$ is a subset of $V$, then $G \setminus V'$ denotes the graph obtained by removing all the vertices in $V'$ and their incident edges from $G$.

**Cycle Bases** A *cycle* is an undirected graph is a subgraph in which every vertex has even degree (no need to be connected). A *circuit* is a connected cycle in which every vertex has degree equals two. Let's associate each cycle to a vector in $\mathbb{Z}_2^m$ as follows[1]:

$$C_k(e) = \begin{cases} 1 & \text{the node } i \text{ belongs to the } k-\text{cycle;} \\ 0 & \text{otherwise.} \quad \forall e \in E \end{cases} \tag{1}$$

The set of all cycle of a graph $G$ forms a vector space over $\mathbb{Z}_2$ (the two-module sum of due cycle of $G$ is a cycle of $G$) which is called the *cycle space* of $G$.

It can be shown ([2]) that the dimension of the cycle space is given by the *cyclomatic number*

$$\nu = m - n + c, \tag{2}$$

where $c$ denotes the number of connected components of $G$. If $G$ is connected and T is a spanning tree of $G$, then adding any edge from $E \setminus T$ to $T$ generates a circuit ([3]). The set of such circuits forms a cycle basis, which is referred to as the *fundamental cycle basis*.

If we consider the directed graph $G$ such that for any vertex $v \in V$ it holds A directed cycle is represented by a vector $c \in \mathbb{Q}^m$ such that for any vertex $v \in V$ it holds

$$\sum_{e \in \delta_+(v)} [\boldsymbol{c}]_e = \sum_{e \in \delta_-(v)} [\boldsymbol{c}]_e, \tag{3}$$

where $\delta_+(v)$ and $\delta_-(v)$ denote the edges leaving and entering $v$, respectively, and $[\boldsymbol{c}]_e$ denotes the component of $\boldsymbol{c}$ indexed by edge $e$. Directed cycles may use arcs in forward ($[\boldsymbol{c}]_e > 0$) or backward ($[\boldsymbol{c}]_e > 0$) direction. In particular we will consider only the case

[1]From this moment, we won't distinguish a cycle from its representation as vector in $\mathbb{Z}_2^m$.

where $[\boldsymbol{c}]_e \in \{-1, 0, 1\}$ where $+1$ indicates that the orientation of the edge coincides with the orientation of the cycle, $-1$ if the orientation of the edge is the reverse of the orientation of the cycle, 0 if the edge does not belong to the circuit. It can be shown that an undirected cycle basis can be turned into a directed cycle basis, but the converse is not true ([3]).

**Matrices associated with graphs.** Let $G = (V, E)$ a finite simple (directed or undirected) graph with $n$ nodes and $m$ vertices. The *adjacency matrix* of $G$ is defined as the $n \times n$ matrix $A(G)$ in which:

$$A(G)_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

A node is not considered adjacent to itself, so $A(G)$ has a zero diagonal. We note that the adjacency matrix is symmetric since the graph is undirected.

The *incidence matrix* of a finite simple directed graph $\vec{G} = (V, E)$ with $n$ nodes and $m$ edges is defined as:

$$B(\vec{G})_{ij} = \begin{cases} 1 & \text{if } i \text{ is the head of } e_j, \\ -1 & \text{if } i \text{ is the tail of } e_j, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The rows of the incidence matrix correspond to vertices of G and its columns to edges of G. Each column has exactly two non zero entries, which correspond to the endpoints of the edge associated to that column. It is shown in [2] that, if $G$ is connected, then $\text{rank}(B(G)) = n - 1$. In the following sections, the adjacency or incidence matrix will be denoted simply as $A, B$ when is clear at which graph are associated.

The *degree matrix* $D$ of $G$, i.e., is the $n \times n$ diagonal matrix such that $[D]_{i,i}$ contains the degree of node $i$:

$$D(G)_{ij} = \begin{cases} \Sigma_j A(G)_{i,j}, & \text{if } i = j \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Equivalently, it can be defined as

$$D(G) = \text{diag}(A(G)\mathbf{1}_{n\times 1}), \tag{7}$$

where $\mathbf{1}_{n\times 1}$ denotes a $n \times 1$ matrix filled by ones. We note that, for undirected graphs, $\Sigma_j A(G)_{i,j}$ is the degree of $v_i$ and for directed graphs $\Sigma_j A(G)_{i,j}$ is the outdegree of $v_i$. The *transition matrix* (a sort of normalized adjacency matrix) of $G$ is defined as $P(G) = D(G)^{-1}A(G)$.

We now introduce the Hadamard product and some proprieties that we will use. Let $A$ and $B$ be two real matrices of dimension $m \times r$. The *Hadamard product* (or *entrywise product*) of $A$ and $B$, denoted $A \circ B$, has dimension $m \times r$ as well, and it is simply the product of the corresponding elements

$$A \circ B = \begin{pmatrix} [A]_{1,1}[B]_{1,1} & [A]_{1,2}[B]_{1,2} & \dots & [A]_{1,r}[B]_{1,r} \\ \vdots & \vdots & \vdots & \vdots \\ [A]_{m,1}[B]_{m,1} & [A]_{m,1}[B]_{m,1} & \dots & [A]_{m,r}[B]_{m,r} \end{pmatrix}. \tag{8}$$

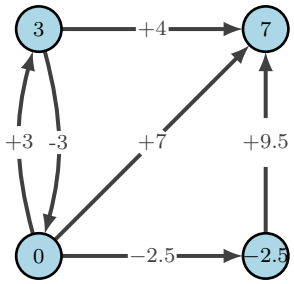



Figure 1: An $(\mathbb{R}, +)$ consistent (edge) labelling.

The Hadamard product is associative, distributive, commutative, and it satisfies the following properties:

$$vec(A \circ B) = \text{diag}(vec(A))vec(B) \tag{9}$$
$$\text{rank}(A \circ B) \leq \text{rank}(A)\text{rank}(B); \tag{10}$$

**Proposition 1.** *Suppose $A, B$ are $m \times n$ matrices, and $D_1$ and $D_2$ are diagonal matrices of size $m$ and $n$, respectively. Then,*

$$D_1(A \circ B)D_2 = (D_1AD_2) \circ B = (D_1A) \circ (BD_2) = (AD_2) \circ (D_1B) = (D_1BD_2) \circ A \tag{11}$$

**Corollary 1.** For any square matrix $A$ and vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ we have

$$\boldsymbol{x}\boldsymbol{y}^T \circ A = \text{diag}(\boldsymbol{x})A\ \text{diag}(\boldsymbol{y}). \tag{12}$$

*Proof.* It follows immediately from Proposition 1 with $B = \mathbb{1}$. □

**Theorem 1** (Perron-Frobenius)**.** *If an $n \times n$ matrix $A$ has non-negative entries (i.e. $A >$ entry-wise) and hasn't block-triangular decomposition then admits a simple, non-negative, real eigenvalue $\lambda$ with maximum absolute value (among all eigenvalues) with corresponding eigenvector positive.*

We note that an adjacency matrix $A(G)$ has a block-triangular decomposition if and only if $G$ is connected. The Perron-Frobenius Theorem implies immediately that, if $G$ is connected, the largest eigenvalue of $A$ has multiplicity 1. Likewise, the largest eigenvalue of the transition matrix $P$ is 1 and has multiplicity 1. It is easy to check that the eigenvector associated to such eigenvalue is $1_{n\times 1}$.

# 1 Basics of Synchronization problem

In this section we will introduce the synchronization problem in the canonical way, later we will discuss the interval arithmetic case. The goal of a synchronization problem is to find values of nodes of a graph, which best "fit" according to the edge labelling.

For example, given $n$ real values unknowns, we want find them based on some given pairwise differences (edge label). This problem is solvable *exactly* only if the pairwise differences are "consistent". In real world scenario, this requirement isn't achievable, then our solution will be the "best" assignment which is the composed by the real numbers that produce the minimum error (based on a reasonably norm). This problem can be generalised considering any group (we gave the example with $(\mathbb{R}, +)$) and is formalized by the following definitions.

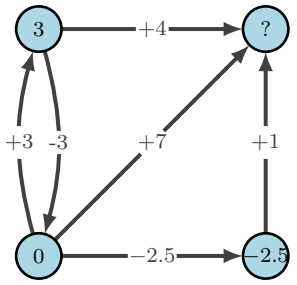



Figure 2: An $(\mathbb{R}, +)$ "inconsistent" (edge) labelling.

**Definition 1** (Labelling)**.** Let $(\Sigma, \circ, 1_\Sigma)$ be a group and $\vec{G} = (V, E)$ be a finite simple digraph, with $n = |V|$ vertices and $m = |E|$ edges. A $\Sigma$-labelled graph is a digraph with labelling of its edge set by elements of $\Sigma$, that is a tuple $\Gamma = (V, E, z)$ where $z \colon E \to \Sigma$ is a labelling of the edges such that if $(u, v) \in E$ then $(v, u) \in E$ and $z(v, u) = z(u, v)^{-1}$.

We note that, since the interval arithmetic isn't a group, in section 3 we are going to make an *ad hoc* definition for a interval labelling. Let's also observe that, for all pair of nodes $(i, j) \in V^2$ the two conditions $x_i \circ z_{i,j} = x_j$ and $x_j \circ z_{j,i} = x_i$ are exactly the same due to the group structure; this won't be the case for the interval labelling.

**Definition 2** (Consistency error)**.** Let $\Gamma = (G, z)$ be a $\Sigma$-labelled graph for $G = (V, E)$ and let $\tilde{x} \colon V \to \Sigma$ be a vertex labelling. Let $f \colon \Sigma \to \mathbb{R}$ be a symmetric, positive definite function with $f(1_\Sigma) = 0$ be the unique minimum. The *consistency error* of $\tilde{x}$ is defined as

$$\varepsilon(\tilde{x}) = \Sigma_{(u,v)\in E} f(\tilde{z}(u, v) \circ z(u, v)^{-1})$$

where $\tilde{z}$ is the edge labelling induced by $\tilde{x} \colon \tilde{z}(u, v) = \tilde{x}^{-1}(u) \circ \tilde{x}(v)$.

We usually say that a labelling is consistent if the consistency error is zero (for every $f \colon \Sigma \to \mathbb{R}$).

**Definition 3** (Synchronization)**.** Given an edge labelling $z$, the *synchronization problem* consists in finding a vertex labelling of $G$ with minimum consistency error.

This formalization can be useful to recover unknown group elements (vertex labels) given a redundant set of noisy measurements of their ratios (edge labels). Synchronization requires the graph to be connected, but errors compensation happens only with cycles.

**Example 4** (Clock synchronization problem)**.** In wireless networks, nodes must often act in coordinated or synchronized fashion. This requires global clock synchronization, wherein all nodes in the network are synchronized to a common clock.

The network is modeled as a directed graph of $n+1$ nodes $\{0, 1, 2, \ldots, n\}$, where each edge represents the ability to transmit and receive packets between the corresponding pair of nodes. Each node $i$ has a clock $t + o_i$[2], where $t$ represents the reference time variable of node 0 and $o_i$ are a unknown real values.

Estimates of clock differences between pair of nodes connected by an edge can be obtained by exchanging of time-stamped packets. So we can obtain the values $o_{i,j}$ (adjusted in the proper way with the delay of time-stamped packets transmission, if needed) which represents an estimate of $o_j - o_i$. The clock synchronization problem can easily be resolved through a synchronization problem in $(\mathbb{R}, +)$ where $x_i = o_i$ and $z_{i,j} = o_{i,j}$.

[2]This is an easy version of the clock synchronization problem

**Synchronization in** $(\mathbb{R}, +)$**.** Let $B$ be a $n \times m$ incidence matrix of $G$ and $\boldsymbol{z}$ the vector containing the edge labels (ordered as in $B$); it is easy to see that for all the edges the equation above becomes $\boldsymbol{x}^T B = \boldsymbol{z}^T$, or

$$B^T \boldsymbol{x} = \boldsymbol{z}. \tag{13}$$

We assume that the graph in connected (each connected component can be solved independently from each other), hence $\text{rank}(B) = n - 1$. Since the solution of the group synchronization problem is uniquely defined up to a global group element, we are allowed w.l.o.g. to arbitrarily set $x_k = 0$ for a chosen $k \in V$. Removing the $x_k$ from the unknowns and the corresponding row in $B$ leaves a full-rank $n - 1 \times m$ matrix $B_k$ (also called "reduced" incidence matrix). Hence we solve

$$B_k^T \boldsymbol{x} = \boldsymbol{z}. \tag{14}$$

The last equation has a unique least squares solution, because we can multiply each side of the equation by $B_k$ . Then, we have that $B_k B_k^T$ is a $(n-1) \times (n-1)$ invertible matrix as $\ker(B_k B_k^T) = \{\mathbf{0}\}$. The mathematical solution is

$$\boldsymbol{x} = (B_k B_k^T)^{-1} B_k \boldsymbol{z}. \tag{15}$$

Considering $f(\cdot) = |\cdot|^2 \colon \Sigma \to \mathbb{R}^+$, the consistency error of the synchronization problem writes

$$\varepsilon(\boldsymbol{x}) = \sum_{(u,v)\in E} |x(v) - x(u) - z(u,v)|^2 = |B^T \boldsymbol{x} - \boldsymbol{z}|^2. \tag{16}$$

Thus, the least squares solution of 14 solves the synchronization problem.

**Proposition 2.** *If $\hat{\boldsymbol{x}}$ is the least-squares solution of Equation 14, then the induced edge labelling $\hat{\boldsymbol{z}} = B_k^T \hat{x}$ solves the following constrained minimization problem*

$$\min_{C\hat{\boldsymbol{z}}=0} |\boldsymbol{z} - \hat{\boldsymbol{z}}|^2, \tag{17}$$

*where $C \in \{-1, 0, 1\}^{(m-n+1)\times m}$ denotes a directed cycle basis matrix.*

This proposition is true for any least-squares problem with the constraint $D\hat{z} = 0$ where $D^T$ is a basis for $\text{null}(B_k)$: $C$ is exactly such a base (see preliminaries).

If $\boldsymbol{c}$ is the indicator vector of a cycle, $\boldsymbol{c}^T \boldsymbol{z}$ is the (algebraic) sum of the edge labels along the cycle, hence the cycle is null iff $\boldsymbol{c}^T \boldsymbol{z} = 0$. If the equations coming from all the circuits in a (directed) cycle basis are stacked, then we get

$$C\boldsymbol{z} = 0. \tag{18}$$

**Synchronization in** $(\mathbb{R}\setminus\{0\}, \cdot)$ In $\Sigma = (\mathbb{R}\setminus\{0\}, \cdot)$ a vertex labelling $x \colon V \to \mathbb{R}$ is consistent with a given edge labelling $z \colon E \to \mathbb{R}$ iff

$$z(u,v) = x(u)^{-1} \cdot x(v) \quad \forall (u,v) \in E. \tag{19}$$

The consistency constraint can be expressed in an equivalent compact matrix form. Let $x$ be the vector containing the vertex labels and let $Z$ be the matrix containing the (inverse of the) edge labels

$$\boldsymbol{x} = \begin{bmatrix} x_1^{-1} \\ x_2^{-1} \\ \vdots \\ x_n^{-1} \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & z_{12} & \dots & z_{1n} \\ z_{21} & 1 & \dots & z_{2n} \\ \vdots & & & \vdots \\ z_{n1} & z_{n2} & \dots & 1 \end{bmatrix}. \tag{20}$$

For a complete graph the consistency constraint rewrites:

$$Z = \boldsymbol{x}\boldsymbol{x}^{-T}, \tag{21}$$

where $\boldsymbol{x}^{-T}$ denotes the row-vector containing the element-wise inverse of $\boldsymbol{x}$, in this case $\boldsymbol{x}^{-T} = (x_1, x_2, \ldots, x_n)$. $Z = \boldsymbol{x}\boldsymbol{x}^{-T}$ contains the edge labels induced by $\boldsymbol{x}$.

Note that Equation 21 implies that $\text{rank}(Z) = 1$, and also that $diag^{-1}(Z) = \mathbf{1}$ and $Z \circ Z^T = \mathbb{1}$.

**Definition 5.** A matrix $Z$ s.t. $Z \circ Z^T = \mathbb{1}$ is said to be *reciprocal*, as this implies that $z_{ij} = 1/z_{ji}$ for all $i, j$. If, in addition, $\forall i, j\ z_{ij} > 0$ than $Z$ is said to be positive.

If the graph is not complete then Z is not fully specified. In this case missing edges are represented as zero entries,[3] i.e. $Z_A \coloneqq Z \circ A$ represents the matrix of the available measures, where $\circ$ is the Hadamard product and $A$ is the adjacency matrix of the graph $G$. Hence the consistency constraint writes

$$Z_A = (\boldsymbol{x}\boldsymbol{x}^{-T}) \circ A. \tag{22}$$

Considering $f(\cdot) = ||1 - \cdot||^2 \colon \Sigma \mapsto \mathbb{R}^+$,, the consistency error of the synchronization problem is

$$\varepsilon(\boldsymbol{x}) = \Sigma_{(i,j)\in E}|z_{ij} - x_i^{-1}x_j|^2 = ||Z_A - (\boldsymbol{x}\boldsymbol{x}^{-T}) \circ A||_F^2, \tag{23}$$

where $||\cdot||_F$ denotes the Frobenius norm. The Hadamard product (or entry-wise product) with $A$ mirrors the summation over the edges of $E$ in the definition of the consistency error. The minimization of $\varepsilon$ is a non-linear least squares problem, for which closed-form solutions do not seem to exist; however, we are going to discuss a solution of related version of this problem.

**Spectral Solution.** Let us consider the noiseless case, and let us start assuming that the graph is complete. Using the consistency constraint and the fact $\boldsymbol{x}^{-T}\boldsymbol{x} = n$ we obtain

$$\frac{1}{n}Z\boldsymbol{x} = \boldsymbol{x} \tag{24}$$

which means that, in the absence of noise, $\boldsymbol{x}$ is the eigenvector of $Z/n$ associated to the eigenvalue 1. Note that, since $Z$ has rank 1, all the other eigenvalues are zero, thus 1 is also the largest eigenvalue of $Z/n$.

We now consider the case of missing edges in which the graph is not complete and the degree matrix $D$ comes into play. Using Proposition 1, the consistency constraint can be expressed as

$$Z_A = (\boldsymbol{x}\boldsymbol{x}^{-T}) \circ A = \text{diag}(\boldsymbol{x})A\text{diag}(\boldsymbol{x}^{-T}) = \text{diag}(\boldsymbol{x})A\text{diag}(\boldsymbol{x})^{-1} \tag{25}$$

which implies that (thanks to $\text{diag}(\boldsymbol{x})\boldsymbol{y} = \text{diag}(\boldsymbol{y})\boldsymbol{x}$)

$$Z_A\boldsymbol{x} = \text{diag}(\boldsymbol{x})A\text{diag}(\boldsymbol{x})^{-1}\boldsymbol{x} = \text{diag}(\boldsymbol{x})A\mathbf{1}_{n\times 1} = \text{diag}(A\mathbf{1}_{n\times 1})\boldsymbol{x} = D\boldsymbol{x}, \tag{26}$$

where $\mathbf{1}_{n\times 1}$ is a vector of ones and $D$ is the degree matrix of the graph (since $A\mathbf{1}$ is the sum of the rows of $A$). This allows to state that

[3] We note that 0 does not belong to the group, hence it is available as "special" value.

**Proposition 3** (Singer, 2011)**.** *The vertex labelling* $\boldsymbol{x}$ *is the eigenvector of* $D^{-1}Z_A$ *associated to the eigenvalue* 1*.*

*Proof.*

$$Z_A\boldsymbol{x} = D\boldsymbol{x} \iff (D^{-1}Z_A)\boldsymbol{x} = \boldsymbol{x}. \tag{27}$$

□

Note that the incomplete data matrix $Z_A$ has full rank in general, thus 1 is not the unique non-zero eigenvalue of $D^{-1}Z_A$, in contrast to the case of a complete graph. However, it can be shown that 1 is the largest eigenvalue of $D^{-1}Z_A$.

**Proposition 4.** *The matrix* $D^{-1}Z_A$ *has real eigenvalues. The largest eigenvalue is* 1 *and it has multiplicity* 1*.*

*Proof.* Observe that: $D^{-1}Z_A = D^{-1}(Z \circ A) = Z \circ (D^{-1}A)$ by virtue of Theorem 1 hence $D^{-1}Z_A$ and $D^{-1}A$ are similar, i.e. they have the same eigenvalues. The matrix $P = D^{-1}A$ is the transition matrix of the graph $G$, which, as a consequence of the Perron-Frobenius Theorem (1), has real eigenvalues and 1 is the largest eigenvalue (with multiplicity 1), if the graph is connected. □

The proof of the proposition above has pointed out that if $Z$ is a reciprocal matrix, the matrix $D^{-1}Z_A$ has a particular structure that yields real eigenvalues, although it is not symmetric. In particular, the eigenvalues do not depend on the measured data, but they depend only on the structure of the graph $G$ (through its transition matrix).

Note that an eigenvector is defined up to scale, and the scale indeterminacy is in agreement with the fact that the solution to synchronization is defined up to a global group element.

**Example 6.** We show an heuristic example where a problem can be effectively formalized and solved through a synchronization problem with $(\mathbb{R}\setminus\{0\}, \cdot)$ group. Suppose a judicial panel has to evaluate the performances of a group of athletes (or various brands) through comparisons: judges have to express pair-wise comparisons like "A is 3 times better than B" or "A is 1/3 better (worse) than B". In this cases, is reasonable formalize the problem with a graph, considering a node for each athlete and an oriented edge for each comparison (just take the mean for different evaluations of a pair). The node labelling (up to a scalar value) obtained through the solution of the synchronization problem, $(\mathbb{R}_{>0}, \cdots)$ case, will be a great absolute (means all compared to a single athlete: the anchor) evaluation of all the athletes.

# 2 Interval arithmetic

In the present section we want to explain what is interval arithmetic and why should we use it. Interval arithmetic is also known as interval computation: this gives us the idea that we want to represent real numbers as intervals (possibly small ones) and execute mathematical computations on it. Interval arithmetic is also a tool to formalize error measurements and find the exact errors propagation through calculations.

An interval $[a, b]$, where $a, b \in \mathbb{R}$, is defined as the set $\{x : a \leq x \leq b\}$. We denote with $\mathbb{IR}$ the interval set, in other words the set of all closed and bounded intervals of $\mathbb{R}$. When we talk about interval arithmetic, we imply that there is an operation such that given

two closed and bounded intervals, it returns a closed and bounded interval. In general, we can define it in a very natural way, just considering operations that are available on $\mathbb{R}$. Given $X, Y \in \mathbb{IR}$ (capital letters will specify elements in $\mathbb{IR}$), we want to define $X \circ Y$ $\circ \in \{+, -, \cdot, /\}$ as follow:

$$X \circ Y = \{x \circ y : x \in X, y \in Y\}. \tag{28}$$

Now, we face the first problem: is the image set of these operations $\mathbb{IR}$? In the case of $+, -, \cdot$, it's not hard to see that all work, but with / we are in trouble: the operation could not be well definite (just consider as divisor ad interval containing zero).

We can discard, for the moment, the operation / and we try to focus our attention only on $\circ \in \{+, -, \cdot\}$. We only highlight that in case we limit our analysis with the set $\mathbb{IR}_{>0} \coloneqq \{[a, b] \in \mathbb{IR} \colon a, b > 0\}$, which is the set of intervals with positive endpoints, division / is well defined thanks to reasoning used in the proof of Proposition 5.

**Proposition 5.** *The operation 28 is well defined in $\mathbb{IR}$ in the case that $\circ \in \{+, -, \cdot\}$. Moreover, $(\mathbb{IR}, \circ)$ is a commutative monoid, but not a group.*

*Proof.* Let $X, Y \in \mathbb{IR}$ and $Z \subseteq \mathbb{R}$ such that $X \circ Y = Z$. We observe that $\min_{z \in Z} z = \min_{x \in X} x + \min_{y \in Y} y$, the analogue is valid for max (and, also, for $-$). In the case of $\cdot$, maximum and minimum of $Z$ exists, but they could be obtained by multiplication of min (or max) depending on the signs of the intervals $X$ and $Y$, so the set $Z$ is then bounded. Moreover, the function $\circ : \mathbb{R}^2 \to \mathbb{R}$ is continuous, so it maps the connected set $X \times Y$ in an other connected set, which must be a (closed) interval in $\mathbb{R}$. Every operation is associative and commutative, because $+, -, \cdot$ are associative and commutative as operation in $\mathbb{R}$.

Note that $+, -$ has $\{0\}$ as identity element, and $\cdot$ has $\{1\}$, which are singletons (the simplest elements of $\mathbb{IR}$).

Finally, only singletons has inverse (except for $\{0\}$ with multiplication): as an example let's consider $[-1, 1]$. Then, by definition, for all $a \in \mathbb{R}$ we have that $\{a\} + [-1, 1]$ is a closed interval of length 2. In general, for all interval $X$ containing $a$ as element, we have that $\{a\} + [-1, 1] \subset X + [-1, 1]$, which is definitely not $\{0\}$. □

We can give, thanks to the proposition above, an operative definition for the interval arithmetic coherently with the Equation 5.

**Definition 7.** Let $\mathbb{IR}$ be the interval set and $[a, b], [c, d] \in \mathbb{IR}$. We define $\circ \in \{+, -, \cdot\}$ as follow:

$$[a, b] \circ [c, d] = [\min\{a \circ c, a \circ d, b \circ c, b \circ d\}, \max\{a \circ c, a \circ d, b \circ c, b \circ d\}].$$

## 2.1 Interval matrix

We follow the analysis conduct in [4]. In this section we consider only square $n \times n$ matrices. Such a matrix $A$ is called nonnegative if all its components are nonnegative. A nonnegative matrix $A$ is reducible if there exists a permutation matrix $P$ such that

$$P^T A P = \begin{pmatrix} B & C \\ 0 & D \end{pmatrix},$$

with $B, C, D$ are all square matrices. We denote with $\rho(A)$ the spectral radius of $A$ and we write $x > 0$ for $x \in \mathbb{R}^n$ if every component of $x$ is positive. Finally, we denote with $\vec{e} = (1, \ldots, 1)^T \in \mathbb{R}^n$.

**Remark 8.** We remind that for every nonnegative irreducible matrix there exists a unique vector $x$ such that $Ax = \rho(A)x$, $\vec{e}^T x = 1$ and $x > 0$. In addition, $\rho(A)$ is the only eigenvalue for which exists a positive eigenvector. Such a vector is called Perron eigenvector.

Now we take in account interval matrices. Let $\underline{A}, \overline{A} \in \mathbb{R}^{n\times n}$ such that $\underline{A} \leq \overline{A}$, then

$$\boldsymbol{A} = [\underline{A}, \overline{A}] = \{A : \underline{A} \leq A \leq \overline{A}\}.$$

The interval matrix $\boldsymbol{A}$ is nonnegative if $\underline{A} \leq 0$. We say that $\boldsymbol{A}$ is irreducible if every $A \in \boldsymbol{A}$ is irreducible.

**Lemma 1.** *A nonnegative interval matrix $\boldsymbol{A} = [\underline{A}, \overline{A}]$ is irreducible if and only if $\underline{A}$ is irreducible.*

*Proof.* If $\boldsymbol{A}$ is irreducible, then by definition also $\underline{A}$ is too. On the other hand, suppose $\underline{A}$ is irreducible, and assume that exists a matrix $A \in \boldsymbol{A}$ that is reducible. Let $P$ be the matrix permutation such that

$$P^T A P = \begin{pmatrix} B & C \\ 0 & D \end{pmatrix}.$$

This implies that

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \leq P^T \underline{A} P = \begin{pmatrix} B_1 & C_1 \\ E_1 & D_1 \end{pmatrix} \leq P^T A P = \begin{pmatrix} B & C \\ 0 & D \end{pmatrix}.$$

Hence, $E_1$ is forced to be a null square matrix and $\underline{A}$ to be a reducible matrix. This is a contradition and proves our thesis. □

Thanks to Lemma 1, if $\underline{A}$ is nonnegative and irreducible, it makes sense to consider the spectral radius and the Perron eigenvector of $\boldsymbol{A}$. We define these concepts as follows.

The spectral radius is

$$\{\rho(A) : A \in \boldsymbol{A}\} = [\rho(\underline{A}), \rho(\overline{A})]$$

where the last equality follows from the continuity of $\rho$ function (see [4] for more details). We now state Perron-Frobenius theorem for interval matrices.

**Theorem 2.** *Let $\boldsymbol{A} = [\underline{A}, \overline{A}]$ be an irreducible nonnegative interval matrix, then the vector $x \in \mathbb{R}^n$ is the Perron eigenvector of a matrix $A \in \boldsymbol{A}$ if and only if*

- $\underline{A} x x^T \leq x x^T \overline{A}$;
- $\vec{e}^T x = 1$;
- $x > 0$.

# 3 Generalization of the synchronization problem with intervals

## 3.1 Formalization

We would like to analyze the synchronization problem on graph, whose edges are labelled with intervals. The main operation we are dealing with is on $\Omega = (\mathbb{IR}, \circ)$, where $\circ$ is $+$ or

$\cdot$. The pair $G = (V, E)$, which include $G$'s structure, is composed by the vertex set $V$ and the edge set $E$. We finally observe that $(\mathbb{IR}, \circ)$ is a commutative monoid with identity elements $\{0\}$ and $\{1\}$ respectively for $\circ$ and $+$. The group structure is absent because only singletons have inverse, so if $I, X, Y \in \mathbb{IR}$ are intervals ($X, Y$ nodes label, $I$ edge label) and we get the equation $X \circ I = Y$, then we aren't able to find the inverse of $I$ and to make $X$ explicit. However this issue could be solved following the procedure explained in Proposition 6.

**Definition 9** (Interval labelling)**.** An interval labelling (called node I-labelling) is a function $\boldsymbol{X} \colon V \to \mathbb{IR}$, where we call $X_k := \boldsymbol{X}(k)$ for all $k \in V$. Given two node I-labelling, $\boldsymbol{x}$ and $\boldsymbol{y}$, we write $\boldsymbol{x} \subseteq \boldsymbol{y}$ if for every $k \in V$ it is true that $\boldsymbol{x}(k) \subseteq \boldsymbol{y}(k)$.

**Definition 10** (Consistent node labelling with an edge I-labelling)**.** A node labelling $x : V \to \mathbb{R}$ is interval-consistent with a given edge labelling $\boldsymbol{z} : E \to \mathbb{IR}$ if

$$\forall (i, j) \in E \qquad x(j) \in x(i) \circ \boldsymbol{z}(i, j).$$

We observe that, given a consistent labelling $x : V \to \mathbb{R}$, for all $a \in \mathbb{R}$ the function $y \colon V \to \mathbb{R}$ defined as $y(i) = x(i) \circ a$ is a consistent labelling. We are in front of an equivalence relation, so we can just only look for a solution up to a translation factor. This fact can be implemented just taking one random node (or a chosen one) and setting its value to a constant, such as $1_\Omega$: we call this node *anchor*.

In the next definition we want to formalize the following idea: our best solution to the interval synchronization problem consists on finding the interval labelling which contains, in every node $i$, all (and only) the possible values of the node $i$ of a consistence node labelling (with a fixed anchor).

**Definition 11** (Interval synchronization problem)**.** Given an edge I-labelling $\boldsymbol{z} : E \to \mathbb{IR}$, the solution to the interval synchronization problem is the interval node labelling $\boldsymbol{s} : V \to \mathbb{IR}$ defined as the image of all the interval-consistent node labelling, i.e. for a fixed $k \in V$, let $D$ be

$$D = \{x : V \to \mathbb{R} : x \text{ is interval consistent } \wedge\, x(k) = 1_\Omega\}$$

then $\boldsymbol{s}(i) = \bigcup_{x \in D} x(i)$. We claim that $\boldsymbol{s}(i)$ is an interval.

Finding the solution $\boldsymbol{s}$ to the interval synchronization problem can be really challenging. Due to this fact, our algorithm will find a less refinement labelling, which contains the solution.

**Definition 12** (I-labelling limit)**.** We say that an I-labelling is a *limit* I-labelling if $X_k \neq \emptyset$ for all $k \in V$ and

$$X_j \subseteq (\boldsymbol{z}(i, j) \circ X_i) \qquad \forall (i, j) \in E.$$

Given two nodes $i, j \in V$ such that $(i, j) \in E$ and a I-labelling, we can spread $X_i$, i.e. computing $\boldsymbol{z}(i, j) \circ X_j$, only following the direction of the edge $(i, j)$, because of the lack of inverses in $(\mathbb{IR}, \circ)$. The presence of a limit I-labelling is, intuitively, a good goal because the concept of a consistent I-labelling is very weak. In fact, an I-labelling where $X_k = (-\infty, +\infty)$ for all $k \in V$ is consistent, but it tells nothing about any possible consistent node labelling.

We adopt an iterative approach, where each node evaluates its label only on the basis of its neighborhoods. Each node periodically communicates its label to the adjacent nodes; there is no constrain about the timing of this communication, and it is not required that

the label is sent to all the adjacent nodes at once. By iterating these steps, we claim that the node labels asymptotically "converge" to the interval solution $\boldsymbol{s}$. This mean that with an iterative approach, which is described later, we end up with a limit I-labelling such that $\boldsymbol{s}(k) \subseteq X_k$ for every $k \in V$. It is easy to see that there are many possible limit I-labelling, but there are some which are the best, i.e. the ones where mean amplitude is the smallest possible. This goal strictly depends on the conditions we have imposed at the beginning of the algorithm, but also on the algorithm itself.

## 3.2 How interval synchronization works

**Interval synchronization for** $(\mathbb{IR}, +)$ In section 2, we have already analyzed the basics of the synchronization problem with a general group, but now we want to focus on synchronization defined on the commutative monoid $\mathbb{IR}$. The genesis of this problem starts from the impossibility of taking precise measurements between nodes. In order to take in account edge labels errors, which usually are symmetric intervals with some (hopefully) small radius, we use interval arithmetic of section 1. In this section we focus on building a procedure to solve synchronization with the interval arithmetic case.

The following works only for $\circ = +$. Let $B$ be a $n \times m$ incidence matrix of a graph $G$ and $\boldsymbol{z} \in \mathbb{IR}$ the vector containing the edge labels (ordered as in $B$). Moreover, let $\boldsymbol{x} \in \mathbb{R}^{|V|}$ be the image set of a node labelling. Similarly to what we have done in section 2, it's easy to see that if the node labelling is interval consistent then we have that

$$\boldsymbol{x} \in (B_k B_k^T)^{-1} B_k \boldsymbol{z}.$$

So, we can consider the following I-labelling

$$\boldsymbol{X} = (B_k B_k^T)^{-1} B_k \boldsymbol{z}, \tag{29}$$

where the $k$-th element of $\boldsymbol{X}$ is $X_k$, that is the I-label of node $k$.

We remind that our aim is to find $\boldsymbol{s}$ (definition (11)), or at least an I-labelling which is near to $\boldsymbol{s}$. So, is (29) a good solutions? As we have said, we are sure that $\boldsymbol{s}(k) \subseteq X_k$, so our I-labelling contains our "true" solution. The tightness of $\boldsymbol{X}$ is strictly dependent upon the quality of data we are dealing with.

**Interval synchronization for** $(\mathbb{IR}_{>0}, \cdot)$ The analysis is pretty similar to the one we have done about the spectral solution, in Section 1. Let $Z$ be the matrix defined as in equation 20, where $\boldsymbol{z}$ is the edge I-labels (if there are no edges connecting two nodes we put $\{0\}$). Consider a node label $x : V \to \mathbb{R}$ and $x^{-T}$ as in (20), then, by definition, we have that

$$x x^{-T} \in Z.$$

Let $A$ be the adjancecy matrix of the digraph $G$ and $Z_A$ as in (22). Hence, we proceed as we have done in (25). We have that

$$\mathrm{diag}(x) A \mathrm{diag}(x)^{-1} \in Z,$$

it follows that $Dx \in Z_A x$, where $D$ is the matrix defined in (6). So our goal is to find a node label $x$ such that

$$x \in D^{-1} Z_A x.$$

Let $S = \{x : V \to \mathbb{R} \,|\, x \in D^{-1} Z_A x\}$, then we have to find $\boldsymbol{x} = \bigcup_{x \in S} x$ (with a little abuse of notation), and we claim that $\boldsymbol{x}$ has as image set $\mathbb{IR}$. Actually, it is sufficient to find at

least a node I-labels $\boldsymbol{y}$ for which $\boldsymbol{y} \supseteq \boldsymbol{x}$. Unfortunately, it's not so easy to find such $\boldsymbol{x}$ or an estimate $\boldsymbol{y}$, and there are no predefined INTLAB function for such a task. We have tried manually to implement a function that, given $G$ and $Z$, computes $\boldsymbol{x}$. However, we were able to find an interval estimate $\boldsymbol{y}$ with some high probability contains $\boldsymbol{x}$. In other words, we were not able to prove that all solutions of $x \in D^{-1}Z_A x$ are contained in $\boldsymbol{y}$, but we are sure that at least *part* of that solutions is there (see Figure 3 for some results of an experiment).

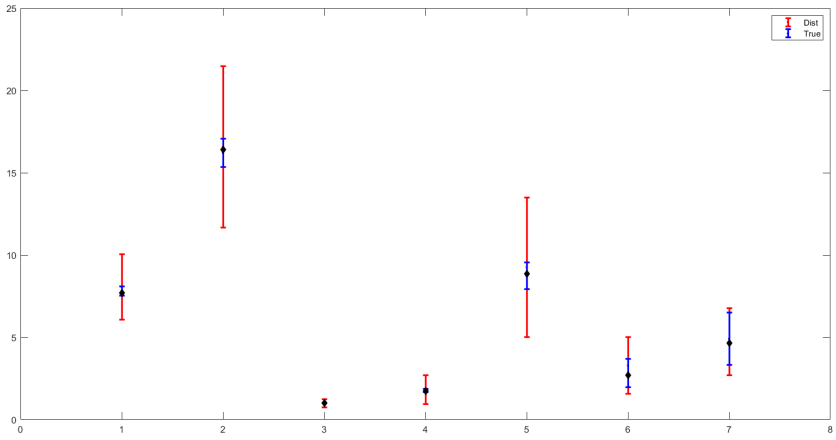



Figure 3: This errorbar is made over a graph $G$ of seven nodes. It represents in red the estimate of the synchronization problem over $(\mathbb{IR}, \cdot)$ given by the 'standard' way we described. In blue IL algorithm results are represented (which are generally better).

In the following section, we would like to generalize the algorithm presented in [1] also to directed graph in $(\mathbb{IR}, +)$ and in $(\mathbb{IR}_{>0}, \cdot)$.

# 4 IL algorithms

**IL algorithm on** $(\mathbb{IR}, +)$ The algorithm we describe is the same as the one discussed in [1], but considering directed graph. We give a little remands on the main steps of the following algorithm (we denote it with IL algorithm, which stands for 'Interval Labelling '). The method proceeds considering one random edge at time, and this is due to the fact that we want to deal with a distributed system, in which every node has only a 'local' knowledge. Every node computes a measurement (or several measurements) of his neighbors and set a interval label on the edge that connects the two. All details of IL algorithm's working are discussed in this section. Now we introduce the first prototype, which will be upgraded with the help of Lemma 2.

**First prototype of IL algorithm on** $(\mathbb{IR}, +)$

1. We start selecting an *anchor*, for instance $k \in V$, and adding the constraint $\boldsymbol{x}(k) = \{0\}$.

2. We set all labels as $(-\infty, +\infty)$, except the anchor which has $\{0\}$.

3. Every unit of time we propagate a randomly chosen node label along a directed edge, i.e. following the corresponding arrows. Formally, given $(i, j) \in E$, let $X_i$ and

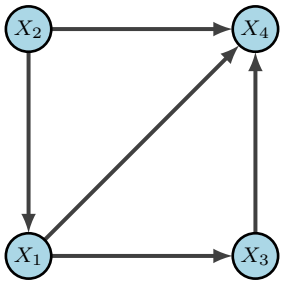


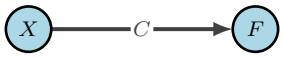

Figure 4

$X_j$ the labels in $\mathbb{IR}$ at time $t$. Let $X'_j = X_i + \boldsymbol{z}(i,j)$, then the update value of node $j$ at time $t+1$ is $X_j \cap X'_j$.

4. We keep on repeating step 3, until $\boldsymbol{X}$ is a limit I-labelling, i.e.

$$X_j \subseteq (\boldsymbol{z}(i,j) + X_i) \qquad \forall (i,j) \in E.$$

**Issues concerning the first prototype** There might be some graphs in which the algorithm we have just described doesn't work as we expect. For example, consider the graph in Figure 4: if the node $X_4$ is the anchor, then our algorithm is useless because we are not able to update any node. On the other hand, if node $X_2$ is not the anchor, then there is no way to update the value $X_2$, so it remains $(-\infty, +\infty)$. All problems comes from the lack of inverses in $\mathbb{IR}$. However, we are not totally hopeless because we are still able to say something about nodes which are not reachable from the anchor. Here we explain how we are able to upgrade performance of IL algorithm.




**Proposition 6.** *Consider two nodes with I-labels $X, F$ linked by a directed edge from $X$ to $F$ with I-label $C$. This situation mathematically implies that we can add an edge from $F$ to $X$ with I-label $-C$.*

*Proof.* Consider a situation like the one in the figure, where $C = [c,d]$, $F = [e,f]$. We know that $X + C = F$ but we want also to upgrade $X = [x,y]$. We have that $[e,f] = X + [c,d]$ and, in principle, this equation says that for every $x \in X$ and for every $t \in [c,d]$ then $x + t \in [e,f]$. We can observe also that the interval $[e,f] - [c,d]$ must contains $X$, because doing $-$ operation in $\mathbb{IR}$ we have just substitute the exists quantifier with a 'for all' one. In other words, we have that

$$[e,f] = X + [c,d] \implies [e,f] - [c,d] = [e-d, f-c] \supset X.$$

In addition, it's easy to see that all elements in $[e-d, f-c]$ could be actually in $X$, but not all at the same time otherwise it would be valid that

$$[e-d, f-c] + [c,d] = [e,f],$$

which is clearly wrong. So the estimate $[e-d, f-c]$ for $X$ it is the better possible knowing that $[e,f] = X + [c,d]$. The inclusion $[e,f] - [c,d] \supset X$, having in mind how IL algorithm works, implies that we can add a directed edge from $F$ to $X$ with I-label $-C$. This proves the thesis. □

### IL algorithm on $(\mathbb{IR}, +)$ over directed and undirected graphs

**Lemma 2.** *Developing the previous consideration, we state that computing the IL algorithm over a digraph is totally equivalent to run it over a specific non-directed graph. The sense of considering a non-directed edge with label $E \in \mathbb{IR}$ must be clarified: this edge is treated by IL algorithm as if there are two directed edges one with $E$ label and the other with $-E$ label.*

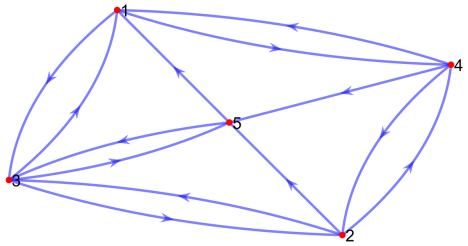



Figure 5

*Proof.* Consider the digraph in the adjacent figure. Sometimes it happens that it has a double edge between two nodes, i.e. $(1,3),(3,1) \in E$, sometimes only one, i.e. $(5,1) \in E$ but $(1,5) \notin E$, and in the remaining cases there are not edges. If we select an edge $(i,j) \in E$, the IL algorithm will update, at random time, vertex $X_j$ using $X_i$ and $\boldsymbol{z}_e$, where $\boldsymbol{z}$ is the edge I-label. However, because of what we have said before we know that $X_i$ must be contained in $X_j - \boldsymbol{z}_e$, and this is completely equivalent of having an edge $(j,i)$ with I-label $-\boldsymbol{z}_e$. If such edge already exists, then we have two measurements from a node to a successor, which means that we can consider just one edge with the intersection of I-label.

We now have to deal with graph for which if $(i,j) \in E$ then also $(j,i) \in E$. We now focus only on two nodes $i,j$ linked by edges $(i,j),(j,i)$. Let $\boldsymbol{z}(i,j) = [a,b]$ and $\boldsymbol{z}(j,i) = [c,d]$. We have to manipulate our edges following what we have described in the previous, i.e. computing intersections $\boldsymbol{z}(i,j) \cap (-\boldsymbol{z}(j,i))$ and $\boldsymbol{z}(j,i) \cap (-\boldsymbol{z}(i,j))$. If, for instance, we have that $a \leq -d \leq b \leq -c$ then we can deduce that the first intersection is $[-d,b]$. A straightforward calculation from the previous chain of inequalities implies that

$$c \leq -b \leq d \leq -a.$$

This inequality tells us that the latter intersection is $[-b,d]$, which is the negative interval of the previous one. Finally, we end up with a couple of edges with opposite I-labels. The reasoning and the result holds in general for any possible combination of $a, b, -c, -d$ along real line. This fact concludes the proof. □

We are now ready to illustrate the complete algorithm. The first prototype IL algorithm is still valid but should be 'initialized' on an appropriate graph, like the one we described in Lemma 2. All issues we have suggested at the beginning of the section are now fixed.

**Final IL algorithm on** $(\mathbb{IR}, +)$

0. Given a digraph G, we build a digraph G' following the step described in the proof of Lemma 2. In detail, for every edge $(i,j) \in E$ we add, if it's not present, a new edge $(j,i) \in E$ giving the I-label $-\boldsymbol{z}(i,j)$. If $(j,i)$ is already an edge, we update its I-label with $\boldsymbol{z}(j,i) \cap (-\boldsymbol{z}(i,j))$. Such an action has to be performed for every edge in the starting graph G.

1. We start selecting a node *anchor*, for instance $k \in V$, and adding the constraint $\boldsymbol{x}(k) = \{0\}$.

2. We set all labels as $(-\infty, +\infty)$, except the anchor which has $\{0\}$.

3. Every unit of time we propagate a randomly chosen node label along a directed edge, i.e. following the corresponding arrows. Formally, given $(i,j) \in E$, let $X_i$ and $X_j$ the labels in $\mathbb{IR}$ at time $t$. Let $X'_j = X_i + \boldsymbol{z}(i,j)$, then the update value of node $j$ at time $t+1$ is $X_j \cap X'_j$.

4. We keep on repeating step 3, until $\boldsymbol{X}$ is a limit I-labelling,[4] i.e.

$$X_j \subseteq (\boldsymbol{z}(i,j) + X_i) \qquad \forall (i,j) \in E.$$

It remains only to proof that IL algorithm reaches an end in a finite number of steps. This is not so easy to proof, but we claim that it's correct. However in application, we are not really interested in actually reach an end because such a number depends on the graph structure and the I-labelling and could be very high. We state this problem in a more precise way in the Conclusion section.

**IL algorithm on** $(\mathbb{IR}_{>0}, \cdot)$ We can adapt the algorithm above to the $(\mathbb{IR}_{>0}, \cdot)$ case. The differences are listed below item by item:

0. instead of $-\boldsymbol{z}(i,j)$ we will consider $\frac{1}{\boldsymbol{z}(i,j)}$;

1. the anchor $\boldsymbol{x}(k)$ is equal to $\{1\}$ instead of $\{0\}$;

2. instead of $X'_j = X_i + \boldsymbol{z}(i,j)$ we will consider $X'_j = X_i * \boldsymbol{z}(i,j)$;

3. we will obtain the $I$-labelling limit when

$$X_j \subseteq (\boldsymbol{z}(i,j) * X_i) \qquad \forall (i,j) \in E.$$

We note that the " $> 0$" in $\mathbb{IR}_{>0}$ is necessary due to the fact that $\frac{1}{\boldsymbol{z}(i,j)}$ (in item 0) could be undefined. However, if we relax the item 0 of the algorithm we can implement it on all $\mathbb{IR}$, because we will only need the $\cdot$ operation.

## 4.1 Pseudo-codes

We now discuss about the code that is behind IL algorithm. The pseudo-code puts in a more formal way all steps of IL algorithm. What we present here is the main core of

[4] In applications, this is not really the case because we cannot predict the number of steps we have to face. However, we use some stop criteria which we describe in the next section (pseudo-code).

**Algorithm 1** IL algorithm on $(\mathbb{IR}, +)$

**Require:** $G$ digraph with $n$ nodes and $m$ edges, $k \in V$ is the anchor, $z$ is $m$-vector of intervals (I-labelling). Stop criteria: max_updates, max_iterations, max_steadiness integer values.

```
G ← step_0(G)                                   ▷ do step 0 of IL algorithm
B ← incidence(G)                                ▷ Incidence matrix of G
integer iter = 0, steady[n], update[n]          ▷ steady, update are n × 1 vectors
interval x[n]                                   ▷ n × 1 vector with interval values
x[k] ← {0}
update[k] ← 1
steady(k) ← max_steadiness + 1                  ▷ anchor is steady
while iter < max_iterations do
    s = random_integer(n)
    N = successors(G, s)                        ▷ vector of nodes successors of s
    if length(N) = 0 then
        continue
    end if                                      ▷ If there are no successors we restart the cycle
    t = random_integer(length(N))
    k = findedge(G, s, t)                       ▷ k is the label of (s, t) edge
    if ¬update(s) and ¬update(t) then
        continue
    end if                                      ▷ we restart the cycle if both nodes are still (-inf,+inf)
    old = x(t)
    if update(s) then
        iter ← iter + 1
        x(t) = intersection(x(t), x(s) + z(k))  ▷ step 3 of IL algorithm
        update(t) ← update(t) + 1
        if old = x(t) then                      ▷ no updates
            steady(t) ← steady(t) + 1
        end if
    end if
    if steady > max_steadiness or update > max_updates then   ▷ stop criteria
        break
    end if
end while
```

the algorithm. There are unexplained functions like step_0, which enlarge the number of edge of graph G and also add (or update) edges I-label. It is sufficient that we feed our algorithm just with measurements of nodes, no matter if there are holes (otherwise we always deal with complete graphs).

In the pseudo-code we introduced some stop criteria, the one which represent the reaching of a limit is steadiness. In fact, we increasing it by +1 every time there is no update (note that we explicit avoid to increase steadiness when $x(s) = (-\infty, +\infty)$ thanks to 'if' statement in line 16).

**How to choose a proper anchor** For the choice of the anchor we recall paper [1], in which the topic is discussed in depth. For our purpose, we just highlight that the first (natural) approach could be to see what is the graph topology and choose, for example, the node with the highest out-closeness. However, this choice does not respect the distributed approach we want to apply. Hence, anchor selection is now turned into anchor 'election', which consist of a distributed algorithm (Algorithm 3.2 of [1]) where all nodes start from a real label. These labels are updated randomly (in a certain way) and, depending on the quality of the information sent by nodes, all labels stabilized on the one that gives the best updates (this is anchor). Such an algorithm provides a unique anchor.

**Performance of IL algorithm on** $(\mathbb{IR}, +)$ We have left implicit the way in which, given a graph G, we produce edge I-labels. We generate random real numbers and we compute the difference between the ones that are linked with an edge (head - tail). We finally create an interval with random ends that contains the previous difference. In details, we can give the following pseudo-code.

**Algorithm 2** How to choose proper edge I-labels

$B \leftarrow$ incidence$(G)$
integer max_values
integer radius
$n \leftarrow$ number_nodes(G); $m \leftarrow$ number_edges(G)
$x \leftarrow$ max_values $*$ rand_numbers(n,1) ▷ $n \times 1$ vector of random real number
$z \leftarrow B^T * x$
$z \leftarrow$ interval$(z -$ radius $*$ rand_numbers$(m, 1), z +$ radius $*$ rand_numbers$(m, 1))$

What could be of much interest is to see is how we measure the performance of this algorithm. Recall we have describe, in Section 3.2, what is interval synchronization and how to compute the limit I-labelling. We explore the results in the next section. Here, for the pseudo-code we consider some typical tools we usually use in MATLAB. In particular, we used INTLAB to perform the interval arithmetic.

**Algorithm 3** Solving synchronization through linear system

$B\_k \leftarrow B$
$B\_k \leftarrow B\_k(k, :) = [\,]$ ▷ k-th row is empty
$x \leftarrow$ solve_linear_system(B_k $*$ (B_k)$^{\mathrm{T}}$, (B_k) $*$ z)
$x \leftarrow [x(1 : (k-1)); 0; x(k : \text{end})]$ ▷ add 0 in anchor index

## 4.2 Experiments

**IL algorithm on** $(\mathbb{IR}, +)$ We consider the graph in Figure 4 with Algorithm 1. We note

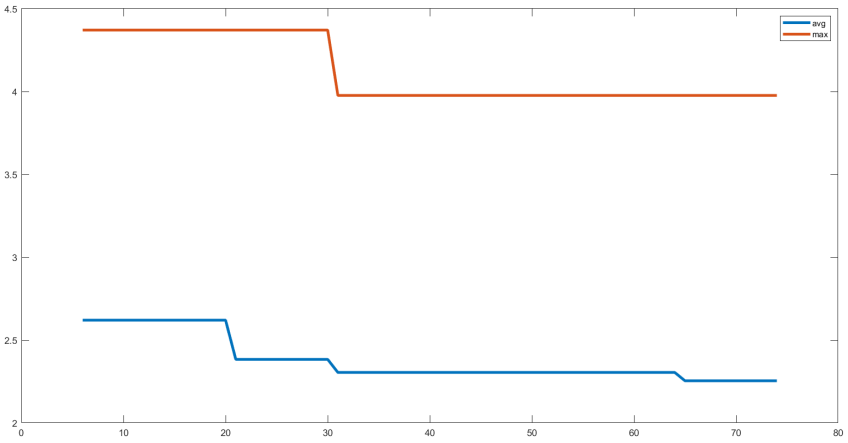



Figure 6: Plot of the average radius (orange) and the maximum radius (blue) iteration by iteration. The initial radius of I-labels has mean 5.

that IL algorithm needs 75 iterations to complete his tasks. As we can see in Figure 6, there is always a gap in the first few iterations because there are still nodes with $(-\infty, +\infty)$ label. Once all nodes have updated $(-\infty, +\infty)$ label, IL algorithm keeps improving labels, as expected, until one of the stop criteria is reached (in this case steadiness).

As [1] correctly notes, the greater is the cyclomatic number the better the final average radius. This could be reported easily by experiments, but it's almost a trivial result. Indeed, more edges available implies more possibilities to update nodes I-label with respect to a situation with fewer edges.

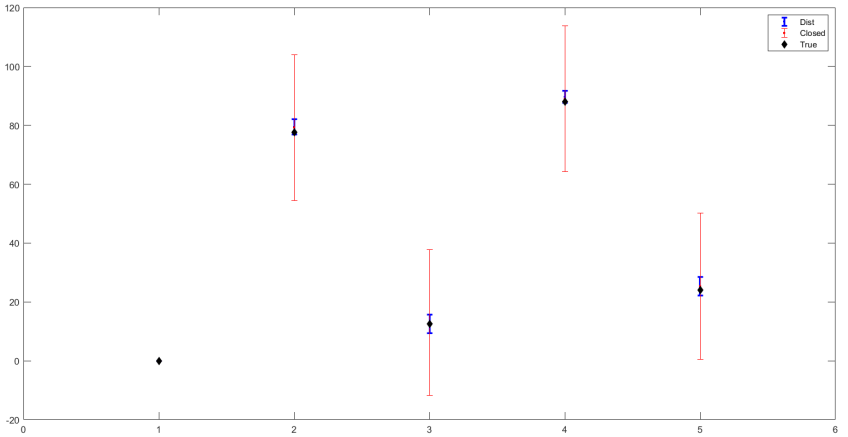



Figure 7: Plot of the final radius of each node with IL algorithm (blue) and with the closed form of linear system (red). The black diamond represent the point from which edge I-labels are generated randomly (see Algorithm 2). The initial radius of I-labels has mean 5.

Through Figure 7 we can underline the difference between IL algorithm and solving the linear system 29. There is one nodes I-label that is a point, and is of course the anchor in both cases. On the other hand, even in a very tiny graph of just five nodes there is an appreciable difference in the quality of the results. The sense of the issue is that solving

the closed form of the problem gives a node I-labeling which is not limit. Hence, it can be updated using IL algorithm to obtain an I-labeling 'closer' to the true solution $\boldsymbol{s}$ of Definition 15.

**IL algorithm on** $(\mathbb{IR}_{>0}, \cdot)$ For the $(\mathbb{IR}_{>0}, \cdot)$ case, we will test the performance of our algorithm based on a random sample. We create a sample of $N = 40$ graphs generated by an algorithm which briefly works as following: $n$ nodes are randomly generated in a unit square; an edge links due nodes if and only if the due nodes have a distance minor equal a given parameter $r$. From this, we consider a We note that, for $r \geq \sqrt{2}$ the algorithm will surely produce a complete graph. For the I-labelling, we randomly choose a node labelling (which will be in the I-labelling that we get from the IL algorithm) and produce a consistent I-labelling with average radius equal to 5. The sample is generated with parameter $r = 0.23 + \frac{u}{3N}$ with $u \in 1, 2, \ldots, N$ all with $n = 40$ nodes.

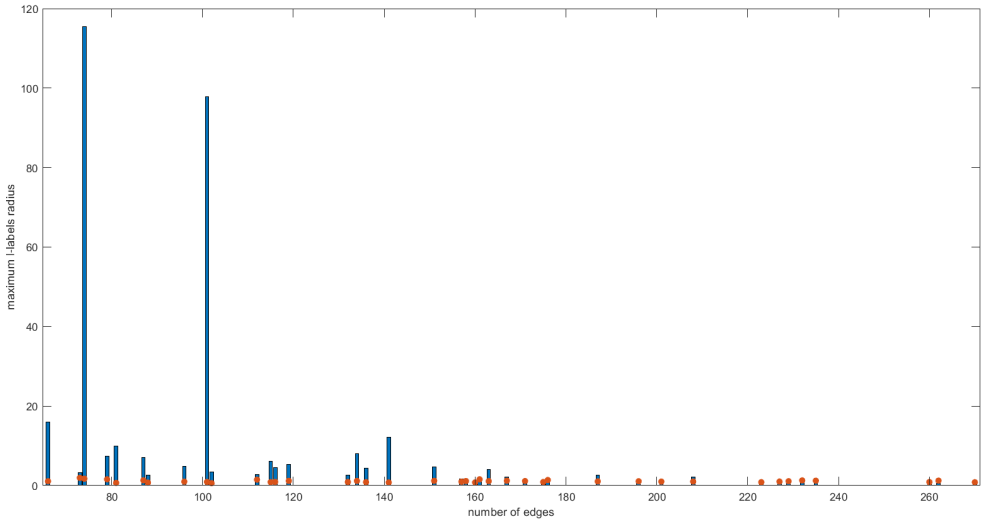



Figure 8: Numbers of edges vs. maximum node I-labels radius (blue) and average edge I-labels radius (red).

We can note, from Figure 8, that despite more of the times our algorithm will drastically decrease the maximum radius of each node, can happen that, for some specific graph structure with few edges, an "unfortunate" node doesn't get a great update. We also observe that, with strongly connected graphs (high cyclomatic number), the maximum radius is way higher than the average edge I-labels radius, which ensure a good effectiveness of our algorithm. This is also shown from the Figure 9.

We can also note that the performance algorithm quickly improve until the cyclomatic number (Equation 2) becomes approximately equal to 130, then the effectiveness remains more or less the same.

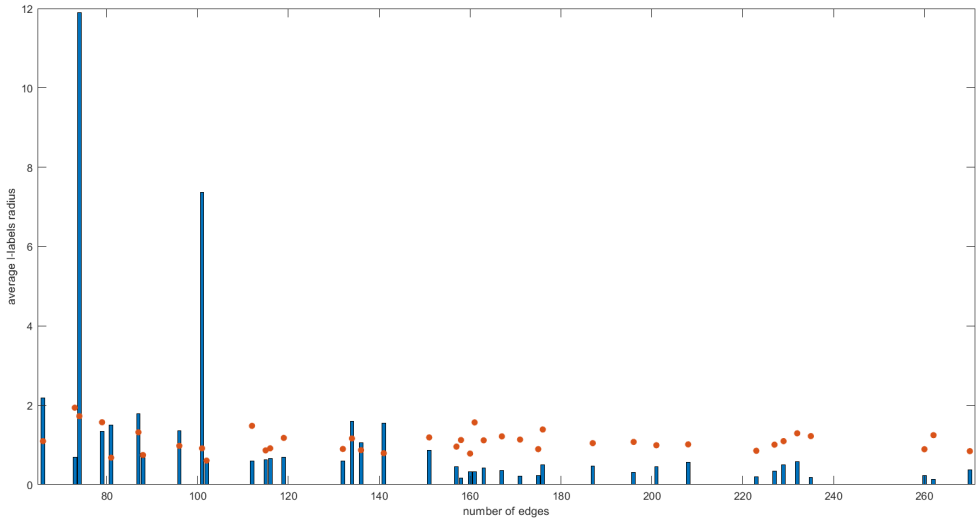



Figure 9: Numbers of edges vs. average node I-labels radius (blue) and average edge I-labels radius (red).

# References

[1] Andrea Fusiello, Pier Luca Montessoro, *Distributed interval synchronization*, University of Udine, 26 september 2022.

[2] Bollobas B., 1998. *Modern Graph Theory*, Springer.

[3] Kavitha, T., Liebchen, C., Mehlhorn, K., Michail, D., Rizzi, R., Ueckerdt, T., Zweig, K., 2009. *Cycle bases in graphs: Characterization, algorithms, complexity, and applications*, 199–243.

[4] Jiri Rohn, *Perron Vectors of an Irreducible Nonnegative Interval Matrix*, august 4 2005, Institute of Computer Science, Czech Academy of Sciences, 182 07 Prague, Czech Republic.