Anchor-Based Localization Using Distributed Interval Contractors

Olivier Reynet, Olivier Voisin, Luc Jaulin

Anchor-Based Localization Using Distributed Interval Contractors

Olivier Reynet, Olivier Voisin and Luc Jaulin

***Abstract*—This paper presents a new method to solve anchor-based distributed localization problems. This method is based on a generic algorithm using interval contractors. In the theoretical part, we detail a new formalism for distributed contractors. This formalism is used to demonstrate that our distributed algorithm converges to the same fixed point than the centralized algorithm. Then, we use this distributed algorithm to solve an anchor-based distributed localization problem in a Wireless Sensor Network (WSN).**

***Index Terms*—Distributed systems, Constraint Satisfaction Problem, Contractor, Interval Analysis, Multi-agent systems, Wireless Sensor Networks, Localization.**

I. Introduction

WIRELESS Sensor Network (WSN) [1] or Mobile Ad-hoc sensor Networks (MANETs) are widely used in many applications, such as monitoring and control [2], [3]. They may be seen as a multi-agent system whose agents are autonomous and where there is no leader. The management and the efficiency [4]–[7] of these systems always depends on the accuracy of the self-localization of the agents. In this paper, we address the problem of the distributed localization in the presence of anchors. Previous works have successfully used interval analysis to localize nodes in mobile networks [8] or to track acoustical sources [9], [10]. In the continuation of these works, this paper states a new formalism for algorithms using distributed *interval contractors* [11], [12]. Beyond localization or tracking, it may be applied to any kind of collaborative signal and information processing, because it does not depend on the nature of the problem.

The originality of our approach is to distribute computation to interval contractors located in the agents. Interval analysis implies that each variable is supposed to be contained in a interval [11]. Interval contractors are algorithms built from a constraint which reduce the size of intervals. The unremoved part of the intervals is compatible with the constraint. They combine techniques as interval analysis [13]–[16], forward-backward propagation [17], monotonicity [18], [19], pruning [20] and domain bisections to build efficient solvers [12].

Interval contractors power lies in their ability to propagate potentially solution sets and to collaborate with each other. We strongly believe that propagating potentially solution sets is smarter than propagating ponctual estimations, because no solution is dismissed and guarantees may be given on the result [21]–[23]. Ponctual estimations are inherently erroneous and generate erroneous values to be propagated and accumulated through the distributed system [24], [25]. Instead, interval contractors efficiently reduce the size of the solution sets throughout the distributed process. Moreover, each agent can exchange its solution set with the others agents and trigger larger contractions performed by the other agents. That is how agents can collaborate through the coherent use of contractors and how contractors can help at decentralizing a problem.

The following section is a theoretical part. It first deals with interval contractors definition and fundamental properties. Then, we describe the multi-agent framework and a centralized approach. We also give a new formalism using contractor graphs. Finally, this section focuses on the Generic Contractor-based Distributed Algorithm (GCDA). We demonstrate that GCDA converges to the same enclosure of the solution than with the centralized approach. The third section focuses on the practical use of GCDA: a distributed localization problem in a Wireless Sensor Network (WSN) is solved and results are analysed.

II. Theoretical Bases

A. Intervals Arithmetics and Boxes

In this paper, we deal with closed intervals, which are connected and closed subsets of $\mathbb{R}$. The set of all intervals of $\mathbb{R}$ will be denoted by $\mathbb{IR}$. An interval is surrounded by brackets and is defined by its lower bound x^- and its upper bound x^+ :

$$[x] = [x^-, x^+] \tag{1}$$

Interval arithmetic[1] has been deeply studied since the eighties [11], [16], [26]. Therefore, we only recall some fundamentals.

If $\diamond \in \{+, -, *\}$ and if $[x]$ and $[y]$ are two intervals, we can define:

$$[x] \diamond [y] \triangleq [\{x \diamond y \mid x \in [x], y \in [y]\}]\,. \tag{2}$$

For instance,

$$[-1,3] + [2,5] = [1,8] \tag{3}$$

$$[-1,3] * [2,5] = [-5,15] \tag{4}$$

$$[-1,3] \,/\, [2,5] = \left[-\frac{1}{2}, \frac{3}{2}\right] \tag{5}$$

$$\tag{6}$$

Interval elementary functions may also be defined. If $f \in \{\cos, \sin, \mathrm{sqr}, \mathrm{sqrt}, \log, \exp, \dots\}$, is a function from $\mathbb{R}$ to $\mathbb{R}$, we define its interval extension as

$$f([x]) \triangleq [\{f(x) \mid x \in [x]\}]\,. \tag{7}$$

Olivier Reynet and Luc Jaulin are with the LabSTICC, ENSTA Bretagne, 2 rue F. Verny 29806 Brest, FRANCE.

[1] See http://www.ensta-bretagne.fr/jaulin/intervalcourse.pdf, an excellent introduction to interval contractors.

For instance

$$\sin([0,\pi]) = [0,1] \quad (8)$$
$$\text{sqr}([-1,3]) = [-1,3]^2 = [0,9] \quad (9)$$
$$\text{abs}([-7,1]) = [0,7] \quad (10)$$
$$\text{sqrt}([-10,4]) = \sqrt{[-10,4]} = [0,2] \quad (11)$$
$$\log([-2,-1]) = \emptyset. \quad (12)$$

To deal with higher dimensional problems, we have to introduce boxes.

Definition 1. *A cartesian product of intervals is called a box.*

Thus, a n-dimensional box $[\mathbf{x}]$ can be written as $[x_1]\times[x_2]\times\cdots\times[x_n]$.

B. Interval Contractor Definitions and Properties

An intuitive and simple contrator definition could be the following : suppose we have constraint satisfaction problem (CSP) $\mathcal{H}$, which can be formulated as:

$$\mathcal{H} : (\mathbf{f}(\mathbf{x}) = 0, \mathbf{x} \in [\mathbf{x}]) \quad (13)$$

with $[\mathbf{x}] \in \mathbb{IR}^n$. The solution set of $\mathcal{H}$ is defined as:

$$\mathbb{S} = \{\mathbf{x} \in [\mathbf{x}] \mid \mathbf{f}(\mathbf{x}) = 0\} \quad (14)$$

Contracting $\mathcal{H}$ means replacing $[\mathbf{x}]$ by a smaller domain $[\mathbf{x}']$ such that the solution set remains unchanged: $\mathbb{S} \subset [\mathbf{x}'] \subset [\mathbf{x}]$. Then, a contractor for $\mathcal{H}$ is an operator used to contract $[\mathbf{x}]$. Now we can give a more precise definition:

Definition 2. *A contractor is a monotonic mapping $\mathcal{C}$ from $\mathbb{IR}^n$ to $\mathbb{IR}^n$ such that:*

$$\forall [\boldsymbol{x}] \in \mathbb{IR}^n, \mathcal{C}([\boldsymbol{x}]) \subseteq [\boldsymbol{x}] \quad (contractance) \quad (15)$$
$$\forall [\boldsymbol{x}] \subset [\boldsymbol{y}] \Rightarrow \mathcal{C}([\boldsymbol{x}]) \subset \mathcal{C}([\boldsymbol{y}]) \quad (monotony) \quad (16)$$

Monotonic contractors can be built from elementary algebraic operators and interval inclusion functions [11], [14], [27].

Example 1. *Consider the function defined by:*

$$f : \mathbb{R}^2 \to \mathbb{R}$$
$$(x_1, x_2) \to x_2 - (x_1 - 1)^2 \quad (17)$$

and define $\mathcal{C} : \mathbb{IR}^2 \to \mathbb{IR}^2$ as follows:

$$\mathcal{C}([\boldsymbol{x}]) := (\mathcal{C}_{x_1}, \mathcal{C}_{x_2})([\boldsymbol{x}])$$
$$:= ([x_1] \cap (\sqrt{[x_2]} + 1), [x_2] \cap ([x_1]^2 - 1)) \quad (18)$$

$\mathcal{C}$ is a contractor associated to $f(x) = 0$.

Given a initial domain $[\boldsymbol{x}] = [-2,2]\times[-2,2]$, $\mathcal{C}([\boldsymbol{x}])$ returns $[1-\sqrt{2},1]\times[0,2]$, as sketched in Fig. 1. At this step, $\mathcal{C}$ has reached a fixed point, i.e. $\mathcal{C}([\boldsymbol{x}]) = [\boldsymbol{x}]$.

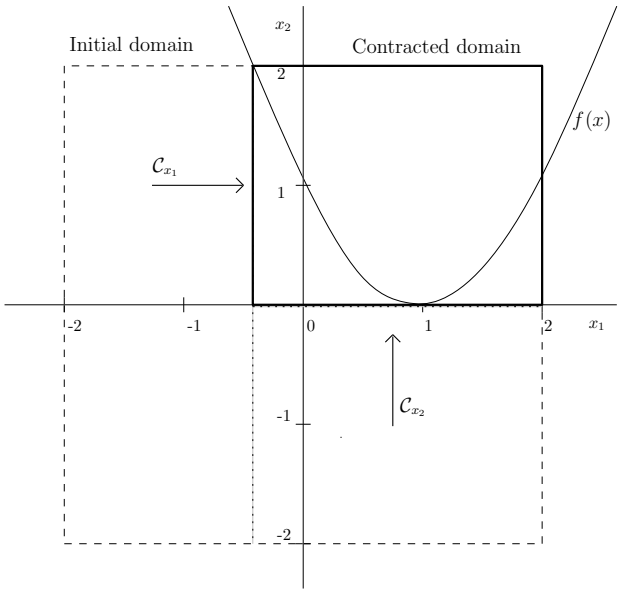


Fig. 1. Illustration of the contractor mecanism on the parabolic constraint f (see eq. (17)). Given a initial domain $[\mathbf{x}] = [-2,2]\times[-2,2]$, applying $\mathcal{C}$ to $[\mathbf{x}]$ results in $[1-\sqrt{2},1]\times[0,2]$

C. Multi-agent Framework Description

Let $\mathcal{A}$ be a set of n agents $\mathcal{A}_i$. An *agent* is a autonomous entity which is able to sense the other agents, to build and compute contractors, and to communicate with the other agents via message passing paradigm. Each agent $\mathcal{A}_i$ has got a vector $\mathbf{p}_i$, which may be estimated by the other agents.

The distributed constraint satisfaction problem can be described by :

- $[\mathbf{p}] = [\mathbf{p}_1]\times[\mathbf{p}_2]\times\cdots\times[\mathbf{p}_n]$, the set of the domains of the vectors. In our interval context, these domains are represented by boxes. The main goal of our algorithm is to contract these boxes.
- $\mathcal{L} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m\}$, the set of all the contractors shared between agents.

D. Generic Contractor-Based Centralized Algorithm

This section proposes a Generic Contractor-based Centralized Algorithm (GCCA) to compute $[\mathbf{p}]$ in the multi-agent framework described in section II-C. We suppose that there is an omniscient supervisor, which has access to the set of contractors $\mathcal{L}$. This supervisor can build a new operator $\mathcal{C}_\infty$, picking and combining [27] contractors from the set $\mathcal{L}$ following a fair strategy as in (19).

$$\mathcal{C}_\infty = \mathcal{C}_1 \circ \mathcal{C}_2 \circ \cdots \circ \mathcal{C}_m \circ \mathcal{C}_1 \circ \mathcal{C}_2 \circ \ldots \quad (19)$$

Lemma 1. *The operator $\mathcal{C}_\infty$ is a contractor.*

Proof: Let first demonstrate the contractance property. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two contractors and $[\mathbf{x}]$ and $[\mathbf{y}]$ two sets such that: $\mathcal{C}_1([\mathbf{x}]) = [\mathbf{y}]$. As $\mathcal{C}_1$ is a contractor, $[\mathbf{y}] \subset [\mathbf{x}]$. As $\mathcal{C}_2$ is a contractor, $\mathcal{C}_2([\mathbf{y}]) = \mathcal{C}_2(\mathcal{C}_1([\mathbf{x}])) \subset [\mathbf{y}]$. Hence, $(\mathcal{C}_2 \circ \mathcal{C}_1)([\mathbf{x}])) \subset [\mathbf{x}]$, which implies that $\mathcal{C}_\infty$ verifies the contractance property.

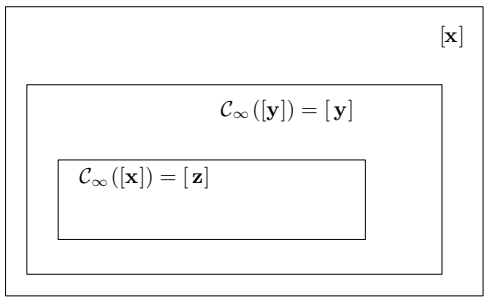


Fig. 2. Elements for the proof by contracdiction of Theorem 2: $[\mathbf{z}] \subset [\mathbf{y}] \subset [\mathbf{x}]$ are such that $\mathcal{C}_\infty([\mathbf{x}]) = [\mathbf{z}]$, $[\mathbf{y}] \subset [\mathbf{x}]$, $\mathcal{C}_\infty([\mathbf{y}]) = [\mathbf{y}]$, and $[\mathbf{z}] \subset [\mathbf{y}]$.

To prove monotonicity, we proceed as follow: let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two contractors and $[\mathbf{x}]$ and $[\mathbf{y}]$ two sets such as $[\mathbf{x}] \subset [\mathbf{y}]$. Then, $\mathcal{C}_2([\mathbf{x}]) \subset \mathcal{C}_2([\mathbf{y}])$, because $\mathcal{C}_2$ is monotonic. Then, $\mathcal{C}_1(\mathcal{C}_2([\mathbf{x}])) \subset \mathcal{C}_1(\mathcal{C}_2([\mathbf{y}]))$, because $\mathcal{C}_1$ is monotonic. It can be rewritten $(\mathcal{C}_1 \circ \mathcal{C}_2)([\mathbf{x}]) \subset (\mathcal{C}_1 \circ \mathcal{C}_2)([\mathbf{y}])$, which means that $\mathcal{C}_1 \circ \mathcal{C}_2$ is monotonic. As all the contractors in $\mathcal{L}$ are monotonic, $\mathcal{C}_\infty$ is a monotonic. ∎

Theorem 2. *$\forall [\boldsymbol{x}] \in \mathbb{IR}^n$, $\mathcal{C}_\infty([\boldsymbol{x}])$ converges to the largest box $[\boldsymbol{z}] \subset [\boldsymbol{x}]$ such that $[\boldsymbol{z}]$ is a fixed point, i.e. $\mathcal{C}_\infty([\boldsymbol{z}]) = [\boldsymbol{z}]$.*

Proof: This theorem has already been proved in [11] p. 92. Nevertheless, we give here another demonstration.

First, we demonstrate that $\mathcal{C}_\infty$ converges using proof by contradiction. Suppose $\mathcal{C}_\infty$ does not converge. Then, we can find boxes $[\mathbf{t}]$ and $[\mathbf{u}]$ and $k \in \{1, 2, \ldots, m\}$ such that $\mathcal{C}_k([\mathbf{t}]) = [\mathbf{u}]$ and $[\mathbf{t}] \subset [\mathbf{u}]$. It means that $\mathcal{C}_k$ does not respect the contractance property $\mathcal{C}_k([\mathbf{t}]) \subset [\mathbf{t}]$, *i.e.* $\mathcal{C}_k$ is not a contractor. Therefore, $\mathcal{C}_\infty$ converges to a fixed point $[\mathbf{z}]$.

Second, we demonstrate that $[\mathbf{z}]$ is the largest fixed point. Using proof by contradiction, we suppose we have three boxes $[\mathbf{x}]$,$[\mathbf{y}]$ and $[\mathbf{z}]$ such that $\mathcal{C}_\infty([\mathbf{x}]) = [\mathbf{z}]$, $[\mathbf{y}] \subset [\mathbf{x}]$, $\mathcal{C}_\infty([\mathbf{y}]) = [\mathbf{y}]$, and $[\mathbf{z}] \subset [\mathbf{y}]$, as sketched in Fig. 2. $[\mathbf{y}] \subset [\mathbf{x}] \Rightarrow \mathcal{C}_\infty([\mathbf{y}]) \subset \mathcal{C}_\infty([\mathbf{x}])$, because $\mathcal{C}_\infty$ is monotonic from lemma 1. This implies that $[\mathbf{y}] \subset [\mathbf{z}]$, which is not true. Therefore, $[\mathbf{z}]$ is the largest fixed point. ∎

Our GCCA centralized approach based on $\mathcal{C}_\infty$ contractor can be stated as detailed in Algorithm 1. From Theorem 2, GCCA reduces $[\mathbf{p}]$ and reaches a fixed point. This approach has already been used to solve localization problem [8]. The content of this section generalizes this previous work, providing a clear formalism for interval contractors and a more generic algorithm.

Algorithm 1 GCCA as computed by the supervisor.

1: Build $\mathcal{C} := \mathcal{C}_1 \circ C_2 \circ \ldots \mathcal{C}_m$ from $\mathcal{L}$
2: **repeat**
3: $\quad [\mathbf{p}] := \mathcal{C}([\mathbf{p}])$
4: **until** a fixed point is reached

E. Generic Contractor-based Distributed Algorithm

In this section, we are looking for a distributed algorithm to compute $[\mathbf{p}]$, because, in many multi-agent frameworks, it may be unpractical or even impossible to gather the whole problem into a single place. In our framework, the agents have the ability to communicate with each other using message passing paradigm. This message passing system is asynchronous: no global time is available, messages can arrive at any times and processing speeds are arbitrary. We also suppose that messages :

- (i) are reliably transferred,
- (ii) can be multicasted.

The messaging primitives are :

- $sendMsg(Destination, MessageContent)$,
- and $getMsg()$ which returns a tuple $(EmitterAgent, MessageContent)$.

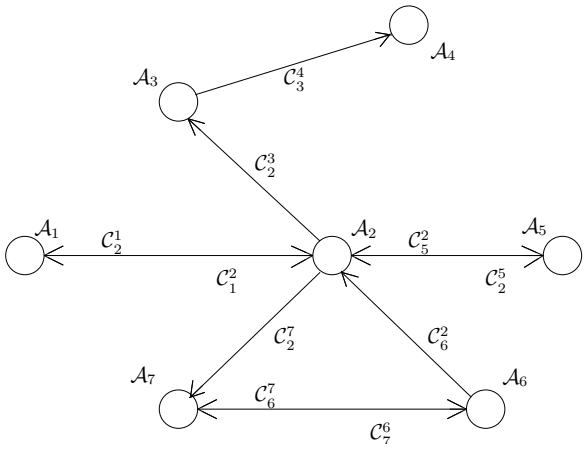


Fig. 3. A contractor graph $\mathcal{G} = (\mathcal{A}, \mathcal{E})$: each edge stands for a contractor $\mathcal{C}_i^j$. The start of the arrow indicates the owner of the contractor. The connectivity is 4 for $\mathcal{A}_2$ and 1 for $\mathcal{A}_3$.

In the following, and for pedagogical reasons, we will only consider binary contractors which can be noted $\mathcal{C}_i^j$ for $i \neq j$ and $i, j \in \{1, 2, \ldots, n\}$. $\mathcal{C}_i^j$ is a contractor of $\mathcal{A}_i$ associated to a constraint between $\mathbf{p}_i$ and $\mathbf{p}_j$. Nevertheless, our approach is not limited to binary contractors and could be extended to any type of contractors.

Our system may be represented by a contractor graph $\mathcal{G} = (\mathcal{A}, \mathcal{E})$ as sketched in Fig. 3. $\mathcal{A}$ is the set of the n vertices of the graph. $\mathcal{E}$ stands for the set of the edges of the graph. An edge between the agents $\mathcal{A}_i$ and $\mathcal{A}_j$ represents a contractor between the $\mathbf{p}_i$ and $\mathbf{p}_j$ variables.

$\mathcal{A}_k$ is a *neighbour* of $\mathcal{A}_i$ if $\mathcal{A}_k$ has got a contractor $\mathcal{C}_k^i$, *i.e.* it exists an arrow starting from $\mathcal{A}_k$ and ending at $\mathcal{A}_i$. We will note $\mathcal{N}_i$ the *set of the neighbours of* $\mathcal{A}_i$. For example, in Fig. 3, $\mathcal{N}_2 = \{\mathcal{A}_1, \mathcal{A}_5, \mathcal{A}_6\}$. It is important to note that our approach does not require $\mathcal{G}$ to be undirected: $\mathcal{A}_k \in \mathcal{N}_i \nRightarrow \mathcal{A}_i \in \mathcal{N}_k$. This kind of situation often appends in a robotics context[2].

Definition 3. *The connectivity c_i of $\mathcal{A}_i$ is:*

$$c_i = card(\mathcal{N}_i) \tag{20}$$

It is the number of edges which end at the vertex $\mathcal{A}_i$.

[2]For example, in a robots swarm, A can see B, but B can not see A, because of obstacles.

We propose the Generic Contractor-based Distributed Algorithm (GCDA) as detailed in Algorithm 2.

Algorithm 2 GCDA as computed by agent $\mathcal{A}_i$

```
1:  procedure GCDA([p_i],N_i)
2:      for all A_k ∈ N_i do
3:          [p_k] := ]-∞,+∞[
4:      end for
5:      SendMsg(N_i,[p_i])
6:      while (A_k,[p_r]) ← getMsg() do
7:          if r == i then
8:              [p_i] := [p_i] ∩ [p_r]        ▷ Update [p_i] from A_k
9:          else if r == k then
10:             [p_k] := [p_k] ∩ [p_r]        ▷ Update [p_k] from A_k
11:         end if
12:         ([p_i],[p_k]) := C_i^k([p_i],[p_k])
13:         if [p_k] has been contracted then
14:             sendMsg(A_k,[p_k])            ▷ A_k must update [p_k]
15:         end if
16:     end while
17:     if [p_i] has been contracted then
18:         sendMsg(N_i,[p_i])                ▷ N_i must update [p_i]
19:     end if
20: end procedure
```

At the beginning of the algorithm, $\mathcal{A}_i$ sets the unknown initial domains of the $[\mathbf{p}_k]$ to $]-\infty,+\infty[$. The distributed process starts when each agent sends to its neighbours its own estimation of $[\mathbf{p}_i]$ (line 4). Then, while $\mathcal{A}_i$ receives messages from the other agents, $\mathcal{A}_i$ updates its knowlegde of $[\mathbf{p}_i]$ and $\mathcal{N}_i$ by intersecting its own estimation with the $[\mathbf{p}_r]$ received from the network. $\mathcal{A}_i$ only sends updates of $[\mathbf{p}_i]$ or $[\mathbf{p}_k]$. Therefore, r stands either for i (the agent himself) or for k (the emitter of the message). These intersection steps are valid, as long as no outlier occur. Indeed, $\mathcal{A}_i$ can reduce $[\mathbf{p}_k]$, if the sender $\mathcal{A}_k$ of the message has a better knowledge of $[\mathbf{p}_k]$. Then, $\mathcal{A}_i$ applies its own contractor $\mathcal{C}_i^k([\mathbf{p}_i],[\mathbf{p}_k])$. If a contraction occurs, $\mathcal{A}_i$ sends its new estimation of $[\mathbf{p}_i]$ to its neighbours and waits for messages. $\mathcal{A}_i$ only sends its new estimation of $[\mathbf{p}_k]$ to $[\mathcal{A}_k]$, because $\mathcal{A}_i$ does not know the neighbours of $[\mathcal{A}_k]$. Only $[\mathcal{A}_k]$ is able to correctly propagate $[\mathbf{p}_k]$ to its neighbours $\mathcal{N}_k$

Theorem 3. *GCDA converges to the same fixed point than GCCA.*

Proof: In GCDA, there is no supervisor and contractors are applied through the network by each agent. The network intersection (lines 8 and 10 of GCDA) combines the contractors' results, replacing contractor composition $\circ$ in GCCA. Therefore, the only difference between GCCA and GCDA is the way the contractors are combined. It has been demonstrated [27] that the chaotic order in which contractors are applied has no impact on the convergence property nor on the fixed point, as long as the order is *fair*. A combining strategy is said to be *fair* if, for any $k \geq 1$ and any contractor $\mathcal{C}$ of $\mathcal{L}$, there exists $j \geq k$ such that $\mathcal{C}$ is computed at rank j. Therefore, the only proof we have to give is that GCDA's strategy is *fair*.

Suppose that GCDA's strategy is not fair. It means that it exists a contractor $\mathcal{C}$ of $\mathcal{L}$ and a step k after which $\mathcal{C}$ is never computed. However, the instruction $sendMsg$ of line 14 and 18 guarantees that all contractors are called if needed, *i.e.* if their domains have changed and have been contracted. Then, if such a contractor $\mathcal{C}$ exists, it means that a message has not been received. But, at the beginning of this section, we have supposed that messages are reliabily transferred. Therefore, GCDA's strategy is fair and GCDA converges to the same fixed point than GCCA. ■

Classical algorithms often propagate a single inaccurate ponctual solution [25], which magnifies the uncertainty of the result. GCDA power lies in the contractors' ability to work together and to propagate only solution sets: only non solution sets are removed. As it will be shown in the next part, most of the time, only few contractions are needed to reach a fixed point. Besides, our interval context naturally generates results which take into account uncertainty: results are boxes which contain solutions. Finally, the contractor graph does not need to be connected (see section III-E) to apply GCDA: lonely agents can still apply GCDA, even if the result might be less accurate.

III. GCDA In Use

A. Simulation Framework

We consider a static Wireless Sensor Network of n agents as sketched in Fig. 4. Each agent is interested in computing its pose vector $\mathbf{p} = (x, y)$, where (x, y) are the 2D-coordinate. Each agent can observe the other agents which are located in some sensing range denoted by r. Sensors' data is the distance d_{ab} between an agent and a neighbour of this agent. This measurement is noisy: in our context, d_{ab} will be represented by an interval $[d_{ab}]$. The bounded noise may be noted: $[-\nu, \nu]$. Therefore we have: $[d_{ab}] = [d_{ab} - \nu, d_{ab} + \nu]$. No other hypothesis is made on the nature of the noise.

Our simulations are such that agents are randomly spread in a two-dimensional space. Some of these agents are anchors, which means that they precisely know their position[3]. If we have m anchor among n agents, we will note the *anchor ratio*:

$$a = \frac{m}{n}. \tag{21}$$

The connectivity of an agent is the number of neighbours which can be sensed by an agent. In our simulation, we set the *mean conectivity* c of the agents at a certain level. The main parameters are then n, c, and a. Then, if r is fixed, we first compute the *density of agents* D of the scene using:

$$D = \frac{c+1}{\pi r^2}. \tag{22}$$

We can deduce the *total surface* of our scene:

$$S = n.D. \tag{23}$$

and the corresponding *edge size*, supposing the scene is a square:

$$e = \sqrt{S}. \tag{24}$$

[3]thanks to GPS for example.

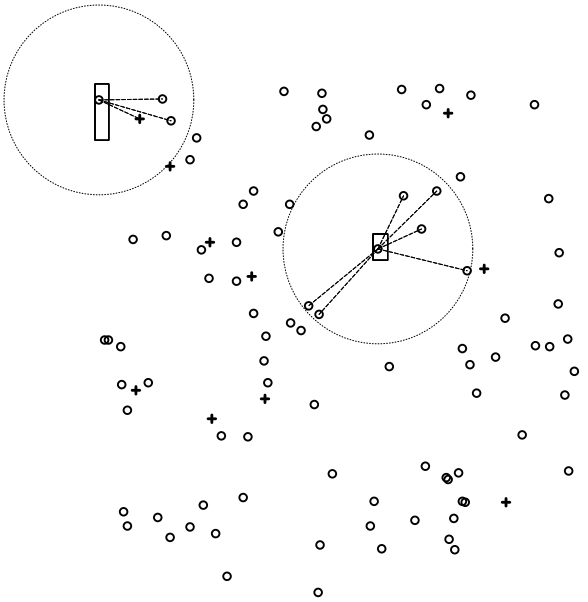


Fig. 4. Simulation with $n = 100$, $a = 10\%$, $c = 10$, $r = 100$, $D = 0.00035$, $S = 285599m^2$ and $e = 534m$. Crosses stand for the anchors. Circles are simple agents. We have sketched the parameters and the result for two agents located at top left and near the center of the scene: the sensing range is represented by a dotted circle and the sensed neighbours are linked with a dashed line. The result of the localization for these agent is represented by a box.

Random agents are generated all over a square scene whose side is set to e using an uniform distribution. Among these agents, a % are randomly chosen to be anchors.

In Fig. 4, a scene is sketched: anchors are represented by crosses and simple agents by circles. We have also drawn the sensing range, the real connectivity and the GCDA result for two agents. The top left agent has a connectivity of 3 and the center agent a connectivity of 6. The result of GCDA is represented by a box around the agents.

B. Contractors Programming

We implement a simple localization contractor based on ranging measurements, a range-only contractor. It is derived from the distance equation:

$$d_{ab} = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2} \tag{25}$$

where d_{ab} is the measured distance between agent a and agent b.

To build contractors, we use the C++ open source IBEX library[4], which is based on Profil/Bias[5] interval library. IBEX is dedicated to the design of interval contractor-based solvers. Given some contraints, IBEX provides tools to automatically build and combine [12] powerful interval contractors.

GCDA is directly implemented in the agents. When a message is received by an agent, the contractor is called and updates are triggered. Then, the propagation process arise following the forward/backward [8], [17] algorithm. The graph $\mathcal{G}$ is browsed forward and backward several times in order to reach the fixed point. GCDA quickly reaches a fixed point: after 5 passes, no improvement is observed. Therefore, in the following simulations, we stop the algortihm when each agent has processed five forward/backward.

[4]see http://www.emn.fr/z-info/ibex/

[5]see http://www.ti3.tu-harburg.de/Software/PROFILEnglisch.html

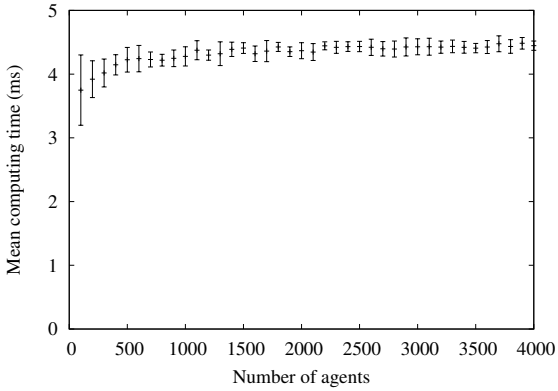


Fig. 5. Mean computing time of an agent versus the number of agents. The connectivity is 10, the anchor ratio 10% and five forward/backward passes are executed. Error bars are extracted from the 100 random simulations which have been computed for each point.

C. Computing Time

As shown in Fig. 5, the mean computing time per agent is near constant versus the total number of agents. This is due to the fact that the computation on each agent does not depend on all the agents, but only on the neighbours ($\mathcal{N}_i$). Hence, for a given connectivity, the computing time per agent is constant.

In this paper, we do not focus on simulating the exchanges through the network. We do not take them into account in the computing time. But, message exchanges are important regarding complexity [28], [29], because, in practice, sending a message takes time. GCDA limits the propagation of messages to the neighbours ($\mathcal{N}_i$). So, the number of messages does not explode with the number of agents.

D. Localization Error Versus Anchor Ratio

We first study the localization error versus the anchor ratio r. The constant parameters of these simulations are $n = 1000$, $c = 10$, $r = 100m$. The ranging noise $\pm 2\%$ of the sensing range r. Anchor ratio has been moved from 1% to 26%. For each anchor ratio, 100 random simulations have been computed: agents are randomly spread over the surface generating a different configuration. For one agent, 4 ms are necessary to compute the localization on an CPU cadenced at 2.00GHz.

In Fig. 6, the localization error is plotted versus the anchor ratio. This error is computed using the position of the center of the box and the true location. The error is normalized by e, the size of the side of the scene[6] and compare to the true

[6]In this case, as the connectivity is fixed to 10, the size of the side of the scene is equal to $1689m$.

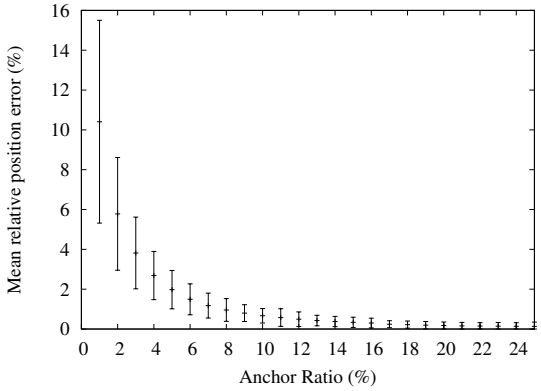


Fig. 6. Relative position error versus anchor ratio for 1000 agents and a connectivity of 10. Error bars are extracted from the 100 random simulations which have been computed for each point. Mean computing time for one agent is 4.2 ms.

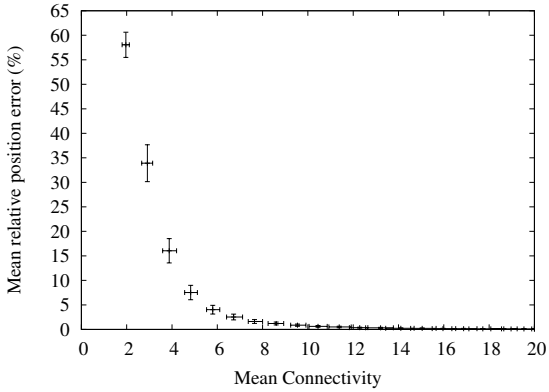


Fig. 7. Relative position error versus mean connectivity for 1000 agents and an anchor ratio of 10%. Error bars are extracted from the 100 random simulations which have been computed for each point.

location. This relative error is lower than 15%, even for a anchor ratio a lower than 5%. A relative error lower than 1% is even reached, when the key threshold of 6% of anchor is exceeded.

E. Localization Error Versus Mean Connectivity

The relative error can also be plotted versus the mean connectivity, as shown in Fig. 7. The mean connectivity has to exceed 5 to decrease the error lower than 10%. In Fig. 8, we plot the connectivity graph of a random simulation with $c = 3$. It clearly shows that the agents may be disconnected or poorly connected. This is the main reason why the error is large when $c \leq 5$. Even if GCDA can be used in these cases, the result on a disconnected agent is obviously bad.

When connectivity is low, results might be improved by using a more efficient contractor [30] on each agent: it will further reduce the size of the box and then we could reuse GCDA to improve the distributed localization. If we were considering mobile networks instead of a static WSN, we could also add some mobility constraint [8] and further reduce the error.

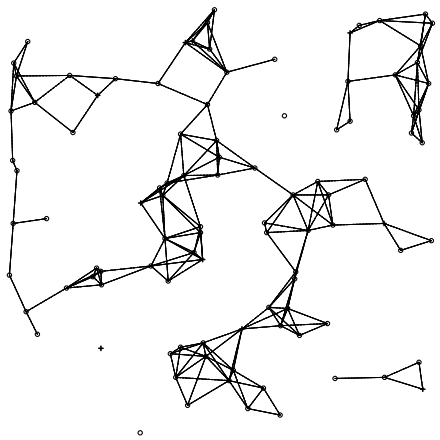


Fig. 8. Connectivity of each agent of a random simulation with $n = 100$, $a = 10\%$, $c = 3$, $r = 100$, $D = 0.00019$, $S = 523599m^2$ and $e = 723m$. Some agents are disconnected, because of the bounded sensing range.

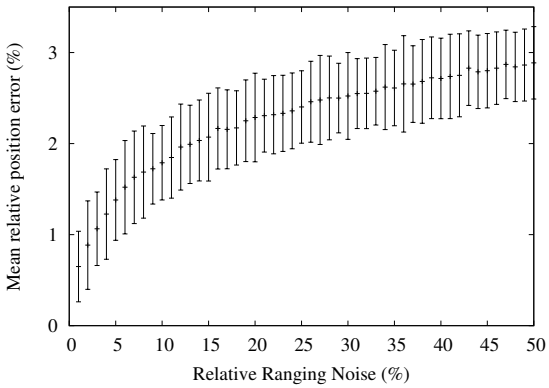


Fig. 9. Relative position error versus relative sensing noise for 1000 agents, an anchor ratio of 10% and a mean connectivity of 10. Error bars are extracted from the 100 random simulations which have been computed for each point.

F. Localization Error Versus Sensing Noise

In Fig. 9, the localization error is plotted versus ν, the intensity of the noise. ν has been moved from 1 to 50% of the sensing range r. In this case, the sensing range is equals to 100 m and the side of the scene is 1689 m. Even in the case of strong noise, GCDA can be applied and a result is obtained. This is due to the bounded error context. Unlike others approaches, the result always contains the solutions, although it is wide. Even in the worst case, *i.e.* when the agent only knows that another agent is detected in any range between 0 and r, GCDA gives sense to this information as follow: the detected agent is contained in a box whose width is $2.r$ and whose center is the agent itsself. Combined to the other sensors information, this is enough to generate new contractions.

IV. Conclusion

GCDA is a powerful algorithm which has been successfully applied to anchor-based distributed localization. It is important to note that our method is fully decentralized. GCDA does not approximate or linearize the mathematical relations. No optimization and no initialization processes are necessary. No assumption about noise gaussianity has been made, thanks to interval analysis context. Applied to localization, our algorithm does not need any anchor placement or gridding. Anchors may be randomly spread. The computation may start from any agent. However, for the moment, GCDA does not take into account outliers. Regarding the future, we aim at robustifying GCDA by taking into account outliers using relaxated contractors [31], [32]. We also want to use GCDA on mobile networks, taking into account the mobility model of the agents.

Acknowledgment

The authors would like to thank the reviewers for their helpful and constructive comment.

References

[1] C. Raghavendra, *Wireless sensor networks*. Springer Verlag, 2006.

[2] S. Kumar, F. Zhao, and D. Shepherd, "Collaborative signal and information processing in microsensor networks," *Signal Processing Magazine, IEEE*, vol. 19, no. 2, pp. 13–14, 2002.

[3] H. Gharavi, S. Kumar, I. of Electrical, and E. Engineers, *Special issue on sensor networks and applications*. IEEE, 2003.

[4] D. Li, K. Wong, Y. Hu, and A. Sayeed, "Detection, classification and tracking of targets in distributed sensor networks," *IEEE signal processing magazine*, vol. 19, no. 2, pp. 17–29, 2002.

[5] L. Eschenauer and V. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*. ACM, 2002, pp. 41–47.

[6] Y. Ko and N. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks," *Wireless Networks*, vol. 6, no. 4, pp. 307–321, 2000.

[7] E. Nerurkar, S. Roumeliotis, and A. Martinelli, "Distributed maximum a posteriori estimation for multi-robot cooperative localization," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 1402–1409.

[8] F. Mourad, H. Snoussi, F. Abdallah, and C. Richard, "Anchor-based localization via interval analysis for mobile ad-hoc sensor networks," *IEEE Transactions on Signal Processing*, vol. 57, no. 8, pp. 3226–3239, 2009.

[9] M. Kieffer, "Distributed Bounded-Error Parameter and State Estimation in Networks of Sensors," in *Numerical Validation in Current Hardware Architectures: International Dagstuhl Seminar, Dagstuhl Castle, Germany, January 6-11, 2008, Revised Papers*. Springer-Verlag New York Inc, 2009, p. 189.

[10] M. Kieffer and E. Walter, "Centralized and distributed source localization by a network of sensors using guaranteed set estimation," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 4. IEEE, pp. IV–IV.

[11] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics*. London: Springer-Verlag, 2001.

[12] G. Chabert and L. Jaulin, "Contractor programming," *Artificial Intelligence*, vol. 173, no. 11, pp. 1079–1100, 2009.

[13] R. Moore, "Interval analysis," *Englewood Cliffs, New Jersey*, 1966.

[14] R. Moore and F. Bierbaum, *Methods and applications of interval analysis*. Society for Industrial Mathematics, 1979.

[15] L. Jaulin and E. Walter, "Set inversion via interval analysis for nonlinear bounded-error estimation," *Automatica*, vol. 29, no. 4, pp. 1053–1064, 1993.

[16] F. Benhamou and W. Older, "Applying interval arithmetic to real, integer, and boolean constraints," *The Journal of Logic Programming*, vol. 32, no. 1, pp. 1–24, 1997.

[17] F. Benhamou, F. Goualard, L. Granvilliers, and J.-F. Puget, "Revising hull and box consistency," in *Int. Conf. On Logic Programming*. MIT press, 1999, pp. 230–244.

[18] I. Araya, G. Trombettoni, and B. Neveu, "Making Adaptive an Interval Constraint Propagation Algorithm Exploiting Monotonicity," *Principles and Practice of Constraint Programming–CP 2010*, pp. 61–68.

[19] G. Chabert and L. Jaulin, "Hull Consistency Under Monotonicity," in *15th International Conference on Principles and Practice of Constraint Programming CP'09 (15th International Conference on Principles and Practice of Constraint Programming)*, ser. LNCS (Lecture Notes in Computer Science), Ian P. Gent, Ed., vol. 5732. Lisbon Portugal: Springer Verlag, 2009, pp. p. 188–195. [Online]. Available: http://hal.archives-ouvertes.fr/hal-00428970/en/

[20] J. Normand, A. Goldsztejn, M. Christie, and F. Benhamou, "A branch and bound algorithm for numerical Max-CSP," *Constraints*, vol. 15, no. 2, pp. 213–237, 2010.

[21] L. Jaulin, M. Kieffer, I. Braems, and E. Walter, "Guaranteed non-linear estimation using constraint propagation on sets," *International Journal of Control*, vol. 74, no. 18, pp. 1772–1782, 2001.

[22] A. Gning and P. Bonnifait, "Constraints propagation techniques on intervals for a guaranteed localization using redundant data," *Automatica*, vol. 42, no. 7, pp. 1167–1175, 2006.

[23] F. Mourad, H. Snoussi, F. Abdallah, and C. Richard, "Guaranteed boxed localization in manets by interval analysis and constraints propagation techniques," in *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*. IEEE, 2008, pp. 1–5.

[24] K. Langendoen and N. Reijers, "Distributed localization in wireless sensor networks: a quantitative comparison," *Computer Networks*, vol. 43, no. 4, pp. 499–518, 2003.

[25] J. Liu, Y. Zhang, and F. Zhao, "Robust distributed node localization with error management," in *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2006, pp. 250–261.

[26] T. Hickey, Q. Ju, and M. Van Emden, "Interval arithmetic: From principles to implementation," *Journal of the ACM (JACM)*, vol. 48, no. 5, pp. 1038–1068, 2001.

[27] K. Apt, "The essence of constraint propagation," *Theoretical computer science*, vol. 221, no. 1-2, pp. 179–210, 1999.

[28] B. Faltings, *Distributed Constraint Programming*, ser. Foundations of Artificial Intelligence. Elsevier, 2006, pp. 699–729.

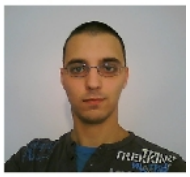
[29] A. Petcu and B. Faltings, "A scalable method for multiagent constraint optimization," in *International Joint Conference on Artificial Intelligence*, vol. 19. Citeseer, 2005, p. 266.

[30] O. Lhomme, "Consistency techniques for numeric CSPs," in *International Joint Conference on Artificial Intelligence*, vol. 13. Citeseer, 1993, pp. 232–232.

[31] L. Jaulin, "Robust set-membership state estimation; application to underwater robotics," *Automatica*, vol. 45, no. 1, pp. 202–206, 2009.

[32] V. Drevelle and P. Bonnifait, "Robust positioning using relaxed constraint-propagation," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 4843–4848.

Olivier Reynet Olivier Reynet was born in Limoges, France in 1976. He received the Ph.D. degree in Electromagnetics from the University of Western Brittany, France in 2003. In 2004, he was awarded the best Ph.D.Prize for his work on active metamaterials by the DGA, the french defense research agency. He is currently Assistant Professor of Embedded Systems at the ENSTA-Bretagne engineering school in Brest, France, since 2007. He does his research on ocean robotics using interval methods and contractor programming.



Olivier Voisin Olivier Voisin was born in 1991. He is an engineer specialized in software embedded systems and automatics.

Luc Jaulin Luc Jaulin was born in Nevers, France in 1967. He received the Ph.D. degree in automatic control from the University of Orsay, France in 1993. He is currently Professor of Robotics at the ENSTA-Bretagne engineering school in Brest, France, since 2004. He does his research on ocean robotics (more precisely with underwater and sailboat autonomous robots) using interval methods and contractor programming.