Randomized Constraints Consensus for Distributed Robust Mixed-Integer Programming

Mohammadreza Chamanbaz, *Member, IEEE*, Giuseppe Notarstefano, *Member, IEEE*, Francesco Sasso, and Roland Bouffanais, *Member, IEEE*

***Abstract*—In this article, we consider a network of processors aiming at cooperatively solving mixed-integer convex programs subject to uncertainty. Each node only knows a common cost function and its local uncertain constraint set. We propose a randomized, distributed algorithm working under asynchronous, unreliable, and directed communication. The algorithm is based on a local computation and communication paradigm. At each communication round, nodes perform two updates: 1) A verification in which they check—in a randomized fashion—the robust feasibility of a candidate optimal point, and 2) an optimization step in which they exchange their candidate basis (the minimal set of constraints defining a solution) with neighbors and locally solve an optimization problem. As a main result, we show that processors can stop the algorithm after a finite number of communication rounds (either because verification has been successful for a sufficient number of rounds or because a given threshold has been reached) so that candidate optimal solutions are consensual. The common solution has proven to be—with high confidence—feasible and, hence, optimal for the entire set of uncertainty except a subset having an arbitrarily small probability measure. We show the effectiveness of the proposed distributed algorithm using two examples: a random, uncertain mixed-integer linear program and a distributed localization in wireless sensor networks. The distributed algorithm is implemented on a multicore platform in which the nodes communicate asynchronously.**

***Index Terms*—Distributed optimization, mixed-integer programming, randomized algorithms, robust optimization.**

Manuscript received May 28, 2020; revised May 31, 2020; accepted August 15, 2020. Date of publication September 17, 2020; date of current version February 26, 2021. This work was supported in part by the European Research Council (ERC) under the European Union's Horizon 2020 Research and Innovation Programme under Grant 638992 - OPT4SMART, and in part by a grant from the Singapore National Research Foundation (NRF) under the ASPIRE Project under Grant NCR-NCR001-040. This article was presented in part at the 20th IFAC World Congress, Toulouse, France, July 2017 [1]. Recommended by Associate Editor M. V. Salapaka. *(Corresponding author: Mohammadreza Chamanbaz.)*

Mohammadreza Chamanbaz and Roland Bouffanais are with the Engineering Product Development (EPD), Singapore University of Technology and Design, Singapore 487372 (e-mail: chamanbaz@u.nus.edu; bouffanais@sutd.edu.sg).

Giuseppe Notarstefano is with the Department of Electrical, Electronic and Information Engineering G. Marconi, University of Bologna, 40126 Bologna, Italy (e-mail: giuseppe.notarstefano@unibo.it).

Francesco Sasso is with the Department of Engineering, Università del Salento, 73100 Lecce, Italy (e-mail: francesco.sasso@unisalento.it).

Digital Object Identifier 10.1109/TCNS.2020.3024483

I. Introduction

ROBUST optimization plays an important role in several areas, such as estimation and control, and has been widely investigated. Its rich literature dates back to the 1950s; see [2] and the references therein. Very recently, there has been a renewed interest in this topic in parallel and/or distributed frameworks. A synchronous distributed random projection algorithm was proposed in [3] and extended to gossip communication in [4], for problems in which each node knows a local cost function and an uncertain constraint. To prove almost sure convergence, the algorithms in [3] and [4] require suitable assumptions on the random set, network topology, and algorithmic weights. A distributed approach based on barrier methods was proposed in [5]. The algorithm needs to start from a strictly feasible point. In [6], a parallel/distributed scheme is considered for solving uncertain problems by means of the scenario approach [7], [8]. The scheme consists of extracting a number of samples from the uncertain set and assigning a portion to each node in a network. A variant of the constraints consensus algorithm introduced in [9] is used to solve the deterministic optimization problem. Primal-dual methods are proposed in [10] for a distributed framework with similar parallel sampling based on the scenario approach. In [11], a cutting plane consensus algorithm is introduced for solving convex optimization problems with common cost, where constraints are distributed through the network processors. For uncertain constraints a worst-case approach based on a pessimizing oracle is used. A distributed proximal minimization algorithm is introduced in [12] for robust convex problems in which each node initially extracts random samples from its local uncertain constraint set.

All aforementioned papers deal with usual (continuous) optimization problems. A centralized method—based on the scenario approach—is presented in [13] for solving robust mixed-integer convex optimization problems. Few recent works address the parallel or distributed solution of deterministic mixed-integer programs. A decentralized (parallel)—but not distributed—approach, based on dual decomposition, is proposed in [14] and [15] to approximately solve mixed-integer linear programs (MILPs) with guaranteed suboptimality bounds. In [16], a distributed version of the algorithm in [15] is proposed. In [17], a distributed algorithm based on primal decomposition is proposed for the same MILP setup. A Lagrange relaxation approach combined with proximal bundle has been used in [18] to solve a demand response problem involving mixed-integer constraints.

In [19], the problem of heterogeneous multivehicle routing is formulated as a MILP and a decentralized algorithm, based on a gossip protocol, is proposed to find a feasible *suboptimal* solution almost surely. A cooperative distributed robust trajectory optimization problem is addressed in [20] in which vehicles sequentially solve a local MILP. Finally, a distributed algorithm based on generation of cutting planes and exchange of active constraints is proposed in [21] to solve deterministic MILPs.

The distributed methods above address deterministic (nonrobust) problems with a finite number of constraints. Extending these methods to uncertain problems would require to combine each proposed algorithm with a procedure to sample uncertain constraints in an online fashion. Thus, the approach we propose in this article can suggest a possible way to address uncertainty in a distributed framework.

The main contribution of this article—which extends the conference paper [1], where this article addresses a more general optimization setup, including a more in-depth analysis and new numerical computations—is the design of a fully distributed algorithm to solve uncertain mixed-integer programs in a network with directed, asynchronous, and possibly unreliable, communication. The problem under investigation is a mixed-integer program in which a cost function depending on a common decision variable has to be minimized subject to a constraint set which is the intersection of local uncertain constraints, each one known only by a single node. Starting from a deterministic constraint exchange idea introduced in [9], the algorithm proposed in this article introduces a randomized, sequential approach in which each node *locally:* 1) performs a probabilistic verification step (based on a "local" sampling of its uncertain constraint set), and 2) solves a deterministic optimization problem with a limited number of constraints. The sequential approach is inspired by the ones proposed in [22]–[25] for *centralized* (not distributed) convex (not mixed-integer) optimization problems. If suitable termination conditions are satisfied, we are able to prove that the nodes reach consensus on a common solution, which is probabilistically feasible and optimal with high confidence. We also propose a stopping criterion ensuring that the distributed algorithm can in fact be stopped after a finite number of communication rounds. Although tailored to mixed-integer programs, the proposed algorithm can solve a more general class of problems known as *S-optimization* problems [26] (which include continuous, integer, and mixed-integer optimization problems). As compared to the literature reviewed above, the proposed algorithm has three main advantages. First, no assumptions are needed on the probabilistic nature of the local constraint sets. Second, each node can sample locally its own uncertain set. Thus, no central unit is needed to extract samples and no common constraint set needs to be known by the nodes. Third and final, nodes do not need to perform the whole sampling at the beginning and subsequently solve the (deterministic) optimization problem. Online extracted samples are used only for verification, which is computationally inexpensive. The optimization is always performed on a number of constraints that remains constant for each node and depends only on the dimension of the decision variable and on the number of neighboring nodes. Since in the addressed setup processors optimize over a common decision variable, integrality constraints cannot be split in the network. As shown later, this increases the number of constraints that nodes need to exchange when the number of integer variables grows. Finally, we remark that the distributed algorithm can be immediately implemented using existing off-the-shelf optimization tools and no *ad hoc* implementation of specific update rules is needed.

The article is organized as follows. In Section II, we provide some preliminaries and formulate the distributed robust mixed-integer convex program (RMICP). Section III presents our distributed, randomized algorithm for finding a solution—with probabilistic robustness—to robust distributed mixed-integer problems. The probabilistic convergence properties of the distributed algorithm are investigated in Section III-B. Finally, extensive numerical simulations are performed in Section IV to show the effectiveness of the proposed methodology.

II. Preliminaries and Problem Setup

In this section, we introduce some preliminary notions of combinatorial optimization that will serve us to set up the RMICP addressed in the article and the methods used for the proposed algorithm.

A. Preliminaries

Given a constraint set $\mathcal{F} = \mathcal{F}^1 \cap \cdots \cap \mathcal{F}^N \subset \mathbb{R}^d$ and a vector $c \in \mathbb{R}^d$, we will denote with $(\mathcal{F}, c)$ the following optimization problem over $S \subset \mathbb{R}^d$:

$$\begin{aligned} &\min c^T \mathbf{x} \\ &\text{subject to } \mathbf{x} \in \mathcal{F}, \\ &\mathbf{x} \in S. \end{aligned}$$

Moreover, let us denote $J(\mathcal{F})$ as the optimal cost of the above problem. For a large class of problems, often known as abstract programs or LP-type problems, a solution to $(\mathcal{F}, c)$ can be identified by considering a subset of the constraints $\mathcal{F}^1, \ldots, \mathcal{F}^N$. This concept is characterized by the notion of *basis*, which is, informally, a "minimal" set of constraints that define a solution. The concept of basis is supported by Helly-type theorems, initially introduced by Eduard Helly in [27]; see also [28]. Before formally defining a basis, we define the *Helly number* and its variation *S-Helly number* [26], [29].

Definition 2.1 (Helly number): Given a nonempty family $\mathcal{K}$ of sets, Helly number $h = h(\mathcal{K}) \in \mathbb{N}$ of $\mathcal{K}$ is defined as the smallest number satisfying the following:

$$\forall i_1, \ldots, i_h \in \{1, \ldots, m\} : K_{i_1} \cap \cdots \cap K_{i_h} \neq \emptyset \Rightarrow$$
$$K_1 \cap \cdots \cap K_m \neq \emptyset$$

for all $m \in \mathbb{N}$ and $K_1, \ldots, K_m \in \mathcal{K}$. If no such h exists, then $h(\mathcal{K}) = \infty$.

For the classical Helly's theorem, $\mathcal{K}$ is a finite family of convex subsets of $\mathbb{R}^d$. One of the important extensions of the Helly number is the *S-Helly number* defined for $S \subset \mathbb{R}^d$. □

Definition 2.2 (S-Helly number): Given a set $S \subset \mathbb{R}^d$, let $S \cap \mathcal{K}$ be the family of sets $S \cap K$ such that $K \subset \mathbb{R}^d$ is convex. The S-Helly number is defined as $h(S) = h(S \cap \mathcal{K})$. □

It is worth pointing out the main difference between the definitions given above. The Helly number states that if the intersection of every h sets is nonempty, then there is a point in common between all the sets. Differently, the S-Helly number states that if the intersection of any h convex sets contains a point in S, then all the sets have a point in common in S.

Definition 2.3 (Basis): Given a constraint set $\mathcal{F} = \mathcal{F}^1 \cap \cdots \cap \mathcal{F}^N$, a basis $\mathcal{B}$ of $(\mathcal{F}, c)$ is a minimal collection of constraints from $\mathcal{F}^1, \ldots, \mathcal{F}^N$ such that the optimal cost of the problem $(\mathcal{F}, c)$ is identical to the one of $(\mathcal{B}, c)$, i.e., $J(\mathcal{F}) = J(\mathcal{B})$. □

We point out that in this article, we are slightly abusing the notation since we are denoting by $\mathcal{B}$ both the collection of constraints (when referring to the basis) and their intersection (when denoting the constraint set of the optimization problem).

The size of a largest basis of problem $(\mathcal{F}, c)$ is called its *combinatorial dimension*. The following result connects the combinatorial dimension with the S-Helly number. This result is presented in [13, Theor. 2], [26, Theor. 1.3], and [29, Theor. 3.11].

Theorem 2.4: Let $\mathcal{F} = \mathcal{F}^1 \cap \cdots \cap \mathcal{F}^N$, then the combinatorial dimension of problem $(\mathcal{F}, c)$ is $h(S) - 1$. □

It is worth noticing that if in problem $(\mathcal{F}, c)$, the sets $\mathcal{F}^1, \ldots, \mathcal{F}^N$ are convex, well-known classes of problems that arise depending on S. If $S = \mathbb{R}^d$, problem $(\mathcal{F}, c)$ is a continuous convex optimization problem. If $S = \mathbb{Z}^d$, then $(\mathcal{F}, c)$ becomes an integer optimization problem. Choosing $S = \mathbb{Z}^{d_Z} \times \mathbb{R}^{d_R}$ with $d = d_Z + d_R$ leads to a mixed-integer convex problem. The S-Helly number for $S = \mathbb{Z}^{d_Z} \times \mathbb{R}^{d_R}$ is given below; see [30].

Theorem 2.5 (Mixed-integer Helly theorem): The Helly number $h(\mathbb{Z}^{d_Z} \times \mathbb{R}^{d_R})$ is equal to $(d_R + 1)2^{d_Z}$. □

This implies that the combinatorial dimension of $(\mathcal{F}, c)$ with $S = \mathbb{Z}^{d_Z} \times \mathbb{R}^{d_R}$ is $(d_R + 1)2^{d_Z} - 1$.

We remark that the combinatorial dimension is an upper bound on the cardinality of the basis, and the basis can have much smaller cardinality than the combinatorial dimension stated in Theorem 2.5.

B. Problem Setup

We consider a network of n processors with limited computation and/or communication capabilities aiming at cooperatively solving the following RMICP:

$$\min_{\mathbf{x} \in S} \; c^T \mathbf{x}$$
$$\text{subject to } \mathbf{x} \in \bigcap_{i=1}^{n} \mathcal{F}^i(q) \;\; \forall q \in \mathbb{Q} \tag{1}$$

where $\mathbf{x} \in S$ is the vector of decision variables, $q \in \mathbb{Q}$ is the vector of uncertain parameters acting on the system, while $\mathcal{F}^i(q) = \{\mathbf{x} \in S : f^i(\mathbf{x}, q) \leq 0\}$ is the constraint set known only by agent i, with $f^i(\mathbf{x}, q) : \mathbb{R}^d \times \mathbb{Q} \to \mathbb{R}$ its related constraint function, which is assumed to be convex for any fixed value of $q \in \mathbb{Q}$. The objective function is considered to be linear. This assumption is without loss of generality. In fact, a nonlinear convex objective function can be transformed into the epigraph form by introducing an extra decision variable. We point out that each processor i only knows part of the problem (the local constraint $\mathcal{F}^i(q)$, the cost vector c, and the domain S) and we stress that there is no central node having access to all constraints. The goal for the network processors is to reach consensus on a solution of the entire problem via a fully distributed algorithm consisting of purely local computation and communication with neighbors.

Problems with this structure arise in a large number of practical applications as, e.g., distributed localization in wireless sensor networks (see Section IV-B).

We make the following assumption on the solution of any deterministic subproblem of (1) in which only a finite number of constraints $\mathcal{F}^i(q)$ (for given sampled q) are considered.

Assumption 2.6 (Nondegeneracy): The minimum point of any subproblem of (1) with at least $h(S) - 1$ constraints is unique and a unique basis exists for the minimum point. □

Assumption 2.6 is not restrictive. In fact, to ensure uniqueness of the optimal point, we could use a strictly convex objective function, a lexicographic ordering, or any universal tie-breaking rule; see [28, Observation 8.1] for further details.

We let the nodes communicate according to a time-dependent, directed communication graph $\mathcal{G}(t) = \{\mathcal{V}, \mathcal{E}(t)\}$, where $t \in \mathbb{N}$ is a universal time which does not need to be known by nodes, $\mathcal{V} = \{1, \ldots, n\}$ is the set of agent identifiers, and $(i, j) \in \mathcal{E}(t)$ indicates that i sends information to j at time t. The time-varying set of incoming (respectively outgoing) neighbors of node i at time t, $\mathcal{N}_{\text{in}}(i, t)$ ($\mathcal{N}_{\text{out}}(i, t)$), is defined as the set of nodes from (respectively to) which agent i receives (respectively transmits) information at time t. A directed static graph is said to be *strongly connected* if there exists a directed path (of consecutive edges) between any pair of nodes in the graph. For time-varying graphs, we use the notion of *uniform joint strong connectivity* formally defined next.

Assumption 2.7 (Uniform joint strong connectivity): There exists an integer $L \geq 1$ such that the graph $\left(\mathcal{V}, \bigcup_{\tau=t}^{t+L-1} \mathcal{E}(\tau)\right)$ is strongly connected for all $t \geq 0$. □

There is no assumption on how uncertainty q enters problem (1), thus making its solution particularly challenging. In fact, if the uncertainty set $\mathbb{Q}$ is an uncountable set, problem (1) is a semi-infinite optimization problem involving an infinite number of constraints. In general, there are two main paradigms to solve an uncertain optimization problem of the form (1). The first approach is a deterministic worst-case paradigm in which the constraints are enforced to hold for *all* possible uncertain parameters in the set $\mathbb{Q}$. This approach is computationally intractable for cases where uncertainty does not appear in a "simple" form, e.g., affine, multiaffine, or convex. Moreover, since some uncertainty scenarios are very unlikely to happen, the deterministic paradigm may be overly conservative. The second approach—the one pursued in this article—is a probabilistic approach where uncertain parameters are considered to be random variables and the constraints are enforced to hold for the entire

set of uncertainty except a subset that has an arbitrarily small probability measure.

III. Randomized Constraints Consensus

In this section, we present a distributed, randomized algorithm for solving problem (1) in a probabilistic sense. Note that since the uncertain constraint sets in (1) are uncountable, it is in general very difficult to verify if a candidate solution is feasible for the entire set of uncertainty or not. We instead use a randomized approach based on Monte Carlo simulations to check probabilistic feasibility.

A. Algorithm Description

The distributed algorithm we propose has a probabilistic nature and consists of two main steps: 1) verification and 2) optimization. The main idea is the following. A node has a candidate basis and candidate solution point. First, it verifies if the candidate solution point belongs to its local uncertain set with high probability. Then, it collects bases from neighbors and solves a convex mixed-integer problem with its basis and its neighbors' bases as a constraint set. If the verification step is not successful, the first violating constraint is also considered in the optimization.

Formally, we assume that q is a random variable, and a probability measure $\mathbb{P}$ over the Borel $\sigma-$algebra of $\mathbb{Q}$ is given. We denote by k_i a local counter keeping track of the number of times the verification step is performed by agent i.

In the verification step, each agent i generates a multisample $\mathbf{q}^i_{k_i}$ with cardinality $M^i_{k_i}$ from the set of uncertainty

$$\mathbf{q}^i_{k_i} \doteq \{q^{(1)}_{k_i,i}, \dots, q^{(M^i_{k_i})}_{k_i,i}\} \in \mathbb{Q}^{M^i_{k_i}}$$

according to the measure $\mathbb{P}$, where $\mathbb{Q}^{M^i_{k_i}} \doteq \mathbb{Q} \times \mathbb{Q} \times \cdots \times \mathbb{Q}$ ($M^i_{k_i}$ times). Node i checks the feasibility of the candidate solution $\mathbf{x}^i(t)$ only at the extracted samples (by simply checking the sign of $f^i(\mathbf{x}^i(t), q^{(\ell)}_{k_i,i})$ for each extracted sample $q^{(\ell)}_{k_i,i}$, $\ell \in \{1, \dots, M^i_{k_i}\}$). If a violation happens, the first violating sample is used as a *violation certificate*.[1]

In the optimization step, agent i transmits its current basis to all outgoing neighbors and receives bases from incoming ones. Then, it solves a convex mixed-integer problem whose constraint set is composed of: 1) a constraint generated at the violation certificate (if it exists); 2) its current basis; and 3) the collection of bases from all incoming neighbors. We define a primitive $[\mathbf{x}^*, \mathcal{B}] = \texttt{Solve}_{\text{MIP}}(\mathcal{F}, c)$ which solves the *deterministic* mixed-integer convex problem defined by the pair $(\mathcal{F}, c)$ and returns the optimal point $\mathbf{x}^*$ and the corresponding basis $\mathcal{B}$. Node i repeats these two steps until a termination condition is satisfied, namely, if the candidate basis has not changed for $2nL + 1$ times, with L defined in Assumption 2.7. The distributed algorithm is formally presented in Algorithm 1.

At this point, it is worth highlighting some key interesting features of the proposed algorithm. First, the verification step, even if it may run possibly a large number of times, consists of simple, inexpensive inequality checks which are not computationally demanding. Moreover, we remark that if at some t the candidate solution has not changed, that is, $\mathbf{x}^i(t) = \mathbf{x}^i(t-1)$, then $\mathbf{x}^i(t)$ has successfully satisfied a verification step and, thus, the algorithm does not perform it again. Second, each node solves a local problem in which the number of constraints depends on the number of neighbors rather than on the total number of agents—which in network applications can be very large. Also, the number of constraints involved in the local problem at each node is fixed. Hence, the complexity of the problem does not change with time. Third, the amount of data a processor needs to transmit does not depend on the number of agents but only on the dimension of the space. Indeed, processor i transmits only $h(S) - 1$ constraints at each iteration and for mixed-integer convex programs, this number only depends on the dimension of the space (see Theorem 2.5). Fourth and final, the proposed distributed randomized algorithm is completely asynchronous and works under unreliable communication. Indeed, t is a universal time that does not need to be known by the processors and the graph can be time-varying. Thus, if nodes run the computation at different speeds, this can be modeled by having no incoming and outgoing edges in that time interval. Similarly, if transmission fails at a given iteration, this is equivalent to assuming that the associated edge is not present in the graph at that iteration.

Remark 3.1 (Implicit constraint S): The constraint S models the fact that optimization problem (1) can be continuous, integer, and/or mixed-integer. For instance, if $S = \mathbb{R}^d$, then all decision variables are continuous and if $S = \mathbb{Z}^{d_Z} \times \mathbb{R}^{d_R}$, the problem

Algorithm 1: Randomized Constraints Consensus.

Input: $\mathcal{F}^i(q)$, c, ε_i, δ_i
Output: $\mathbf{x}^i_{\text{sol}}$
Initialization:
Set $k_i = 1$, $[\mathbf{x}^i(1), \mathcal{B}^i(1)] = \texttt{Solve}_{\text{MIP}}(\mathcal{F}^i(q^i_0), c)$ for some $q^i_0 \in \mathbb{Q}$
Evolution:
Verification:
1) If $\mathbf{x}^i(t) = \mathbf{x}^i(t-1)$, set $q^{\texttt{viol}}_{t,i} = \emptyset$ and goto **Optimization**
2) Extract

$$M^i_{k_i} = \left\lceil \frac{2.3 + 1.1 \ln k_i + \ln \frac{1}{\delta_i}}{\ln \frac{1}{1-\varepsilon_i}} \right\rceil \tag{2}$$

i.i.d samples $\mathbf{q}^i_{k_i} = \{q^{(1)}_{k_i,i}, \dots, q^{(M^i_{k_i})}_{k_i,i}\}$
3) If $\mathbf{x}^i(t) \in \mathcal{F}^i(q^{(\ell)}_{k_i,i})$ for all $\ell = 1, \dots, M^i_{k_i}$, set $q^{\texttt{viol}}_{t,i} = \emptyset$; else, set $q^{\texttt{viol}}_{t,i}$ as the first sample for which $\mathbf{x}^i(t) \notin \mathcal{F}^i(q^{\texttt{viol}}_{t,i})$
4) Set $k_i = k_i + 1$
Optimization:
1) Transmit $\mathcal{B}^i(t)$ to $j \in \mathcal{N}_{\text{out}}(i,t)$, acquire bases from incoming neighbors, and set $\mathcal{Y}^i(t) \doteq \cap_{j \in \mathcal{N}_{\text{in}}(i,t)} \mathcal{B}^j(t)$
2) $[\mathbf{x}^i(t+1), \mathcal{B}^i(t+1)] = \texttt{Solve}_{\text{MIP}}(\mathcal{F}^i(q^{\texttt{viol}}_{t,i}) \cap \mathcal{B}^i(t) \cap \mathcal{Y}^i(t), c)$
3) If $\mathbf{x}^i(t+1)$ unchanged for $2nL + 1$ times, return $\mathbf{x}^i_{\text{sol}} = \mathbf{x}^i(t+1)$

[1] In fact, more than one violated sample—if it exists—can be returned by the verification step. See Remark 3.3 for more details.

is of a mixed-integer nature with d_Z integer and d_R continuous decision variables. Since the decision vector $\mathbf{x}$ is common in all the nodes, the implicit constraint $\mathbf{x} \in S$ needs to be enforced by all nodes but does not need to be communicated among nodes of the network. □

Remark 3.2 (Local optimization and initial constraint): In the deterministic constraints consensus algorithm [9], at each iteration, each node needs to include in the local optimization problem its original constraint set. Here, we can drop this requirement because of the uncertain nature of the problem handled in the verification step. □

Remark 3.3 (Multiple violation certificates): In the verification step of Algorithm 1, we set $q_{t,i}^{\texttt{viol}}$ as the first sample for which $\mathbf{x}^i(t) \notin \mathcal{F}^i(q_{t,i}^{\texttt{viol}})$. However, if more violated samples are found, then in step O_2, we may include $r > 1$ violated constraints in the $\texttt{Solve}_{\text{MIP}}$ primitive, thus solving a larger optimization problem. By doing this, one expects a faster convergence and hence less communications. In fact, as shown in the numerical computations, the number of violated samples r constitutes a tradeoff between the communication burden and the complexity of the local optimization problem. □

Remark 3.4 (Basis computation): In (continuous) convex optimization, a basis, i.e., a minimal set of active constraints, can be efficiently computed. For mixed-integer problems, this can be computationally expensive. A brute-force method is to check all constraints one by one to see whether or not they belong to a basis. Although inefficient, we point out that this routine is applied on a limited number of constraints—the constraint formed at the violation certificate, constraints in the node basis, and constraints in the neighboring bases. Moreover, if a more efficient basis computation procedure were available, it could be immediately used in our distributed algorithm. We note that in view of Theorem 2.5, the combinatorial dimension of the local problems grows exponentially with the number of integer decision variables (d_Z). This clearly affects the cardinality of the basis and, thus, the complexity of its computation and communication. □

B. Algorithm Convergence and Solution Probabilistic Guarantees

Here, we analyze the convergence properties of the distributed algorithm and investigate the probabilistic properties of the solution computed by the algorithm.

We start by defining the set of all possible successive independent random extractions of all the sequences $\{\mathbf{q}_{k_i}^i\}_{k_i=1,\dots,\infty}$, $i = 1, \dots, n$ as follows:

$$\mathbb{S} := \{\mathbf{q} : \ \mathbf{q} = [\mathbf{q}_1, \dots, \mathbf{q}_n] \ , \\ \mathbf{q}_i = \{\mathbf{q}_{k_i}^i\}_{k_i=1,\dots,\infty}, \ \ i = 1, \dots, n\} \ .$$

Moreover, we define the set

$$\mathbb{S}_{\text{sol}} := \{\mathbf{q} \in \mathbb{S} : \ \text{Algorithm 1 terminates}\}$$

which is the domain of the random variables $\mathbf{x}_{\text{sol}}^i$. The probability measure on $\mathbf{x}_{\text{sol}}^i$ is therefore defined on this space, and it is denoted by $\mathbb{P}_\infty$. Now, we are ready to present the first main result of the article.

Theorem 3.5: Let Assumptions 2.6 and 2.7 hold. Given the probabilistic levels $\varepsilon_i > 0$ and $\delta_i > 0$, $i = 1, \dots, n$, let $\varepsilon = \sum_{i=1}^n \varepsilon_i$ and $\delta = \sum_{i=1}^n \delta_i$. Then, the following statements hold.

1) Along the evolution of Algorithm 1, the cost $J(\mathcal{B}^i(t))$ at each node $i \in \{1, \dots, n\}$ is monotonically nondecreasing, i.e., $J(\mathcal{B}^i(t+1)) \geq J(\mathcal{B}^i(t))$, and converges to a common value asymptotically. That is, $\lim_{t\to\infty} J(\mathcal{B}^i(t)) = \bar{J}$ for all $i \in \{1, \dots, n\}$.
2) If the candidate solution of node i, $\mathbf{x}^i(t)$ has not changed for $2Ln + 1$ communication rounds, all nodes have a common candidate solution, i.e., $\mathbf{x}_{\text{sol}}^i = \mathbf{x}_{\text{sol}}$ for all $i = 1, \dots, n$.
3) The following inequality holds for $\mathbf{x}_{\text{sol}}$:

$$\mathbb{P}_\infty\left\{\mathbf{q} \in \mathbb{S}_{\text{sol}} : \mathbb{P}\left\{q \in \mathbb{Q} : \mathbf{x}_{\text{sol}} \notin \bigcap_{i=1}^n \mathcal{F}^i(q)\right\} \leq \epsilon\right\} \\ \geq 1 - \delta.$$

4) Let $\mathcal{B}_{\text{sol}}$ be the basis corresponding to $\mathbf{x}_{\text{sol}}$. The following inequality holds for $\mathcal{B}_{\text{sol}}$:

$$\mathbb{P}_\infty\left\{\mathbf{q} \in \mathbb{S}_{\text{sol}} : \mathbb{P}\left\{q \in \mathbb{Q} : J\left(\mathcal{B}_{\text{sol}} \cap \mathcal{F}(q)\right) > J(\mathcal{B}_{\text{sol}})\right\} \\ \leq \varepsilon\right\} \geq 1 - \delta$$

where $\mathcal{F}(q) \doteq \bigcap_{i=1}^n \mathcal{F}^i(q)$. □

The proof is given in Appendix A.

We briefly discuss the results of this theorem. Statement 1 shows that agents are increasing their local cost and asymptotically achieve the same cost. This result gives insights on the following statement. Statement 2 provides a condition under which processors can stop the distributed algorithm with the guarantee of having a common solution, whose probabilistic properties are shown in the next statements. In the next section, we provide a variation of the algorithm that ensures to stop the algorithm in finite time. We point out, though, that numerical computations show that the algorithm in the original version always satisfies the condition of Statement 2. It is also worth mentioning that for a fixed communication graph, the bound $2nL + 1$ in Statement 2 to stop the algorithm can be replaced by $2D + 1$, where D is the graph diameter. Statement 3 says that the probability of violating the (common) solution $\mathbf{x}_{\text{sol}}$ with a new sample (constraint) is smaller than ε and this statement holds with probability at least $1 - \delta$. Similarly, based on Statement 4, the probability that the cost $J(\mathcal{B}_{\text{sol}})$ associated with $\mathbf{x}_{\text{sol}}$ increases, i.e., the probability of $\mathbf{x}_{\text{sol}}$ not being optimal, if a new sample (constraint) is extracted, is smaller than ε and this statement holds with probability of at least $1 - \delta$.

Finally, note that if problem (1) is infeasible, and, thus, Assumption 2.6 is not satisfied, there are two possibilities. First, at some iteration, an agent samples a collection of constraints that do not intersect and detects infeasibility. This can then be propagated in the network through a flag. Second, the algorithm terminates successfully. If this second scenario occurs, with

confidence of at least $1-\delta$, the set of infeasible constraints has probability measure at most ε.

Remark 3.6 (S-Optimization): The class of problems that Algorithm 1 can solve is not limited to mixed-integer problems. The algorithm immediately applies to any problem having a finite S-Helly number. This class of problems have recently been introduced—under the name of S-optimization problems—in [26]. It is worth mentioning that some classes of nonconvex problems have finite S-Helly number—see [29] and references therein—and hence can be solved using Algorithm 1. □

C. Modified Algorithm With Finite-Time Convergence Guarantee

In this section, we derive a stopping criterion for Algorithm 1 so that it is guaranteed to be halted in *finite time*. Indeed, there is no guarantee for the condition in Statement 2 of Theorem 3.5 to be satisfied (and thus for Algorithm 1 to be halted in finite time), even though we found this to be the case for all simulations we run (see Section IV for more details).

The stopping criterion is derived by borrowing tools from the scenario approach presented in [7] and [8]. In the scenario approach, the idea is to extract a finite number of samples from the constraint set of the semi-infinite optimization problem (1) and solve the obtained (deterministic) optimization problem. The developed theory provides bounds on the number of samples to guarantee a desired probabilistic robustness. The main idea of the stopping criterion is the following. Once each $M^i_{k_i}$ (the cardinality of the multisample used in the verification step of Algorithm 1) exceeds a (common) threshold associated with a proper scenario bound, agents stop generating new samples in the verification step and Algorithm 1 proceeds with the multisample extracted at the last iteration. Thereby, Algorithm 1 solves (in a distributed way) a scenario problem and, in finite time, finds a solution feasible for the extracted samples across all the nodes. Each node has a flag which is communicated locally among all neighbors. The flag is initialized to 0 and is set to 1 once $M^i_{k_i}$ is greater than or equal to the scenario bound.

In [12], scenario bounds are proposed for a distributed robust convex optimization framework in which agents use a different set of uncertainty scenarios in their local optimization programs. In the next lemma, we provide analogous bounds for our problem setup. To this aim, let us consider the following optimization problem:

$$\begin{aligned} &\min\ c^T\mathbf{x} \\ &\text{subject to } \mathbf{x} \in \bigcap_{i=1}^{n}\bigcap_{\ell=1}^{M_i^{\text{scen}}} \mathcal{F}^i(q^{(\ell)}), \\ &\mathbf{x} \in S \end{aligned} \tag{3}$$

where M_i^{scen} is the smallest integer satisfying

$$\delta_i \geq \sum_{\ell=0}^{h(S)-2} \binom{M_i^{\text{scen}}}{\ell} \varepsilon_i^{\ell}(1-\varepsilon_i)^{M_i^{\text{scen}}-\ell}. \tag{4}$$

Lemma 3.7: Given the probabilistic levels $\varepsilon_i, \delta_i > 0$, $i=1,\dots,n$, let $\varepsilon = \sum_{i=1}^n \varepsilon_i$, $\delta = \sum_{i=1}^n \delta_i$, and $M^{\text{scen}} = \sum_{i=1}^n M_i^{\text{scen}}$, where M_i^{scen} is the smallest integer satisfying (4). If agents formulate the sampled optimization problem (3) and all of them agree on an optimal solution $\mathbf{x}^*$

$$\begin{aligned} &\mathbb{P}^{M^{\text{scen}}}\left\{\mathbf{q}\in\mathbb{Q}^{M^{\text{scen}}} : \mathbb{P}\left\{q\in\mathbb{Q} : \mathbf{x}^* \notin \bigcap_{i=1}^{n}\mathcal{F}^i(q)\right\} \leq \varepsilon\right\} \\ &\geq 1-\delta \end{aligned}$$

where $\mathbb{Q}^{M^{\text{scen}}} = \mathbb{Q}\times\mathbb{Q}\times\cdots\times\mathbb{Q}$ (M^{scen} times) and $\mathbb{P}^{M^{\text{scen}}}$ is the product probability measure on $\mathbb{Q}^{M^{\text{scen}}}$. □

The proof follows arguments similar to those in [12, Proposition 1]. Specifically, by using the result in [13, Theor. 3], one can show that the same arguments in [12, Proposition 1] can be followed by properly changing the cardinality of the support constraint set. In particular, in the proof of [12, Proposition 1], the (Helly) number $d+1$ of the (continuous) convex program, with d being the dimension of the decision variable, can be replaced by the S-Helly number $(d_R+1)2^{d_Z}$ (as from Theorem 2.5) of problem (1).

The following proposition summarizes the convergence properties and the solution guarantees of the modified algorithm (Algorithm 1 with the stopping criterion). We report only the feasibility result corresponding to Statement 3 of Theorem 3.5, but also the equivalent of Statement 4 can be proven.

Proposition 3.8: Given the probabilistic levels $\varepsilon_i > 0$ and $\delta_i > 0, i=1,\dots,n$, let $\varepsilon = \sum_{i=1}^n \varepsilon_i$ and $\delta = \sum_{i=1}^n \delta_i$. If nodes stop performing steps V_2 and V_4 of Algorithm 1 when $M^i_{k_i} \geq M_i^{\text{scen}}$ for all $i=1,\dots,n$, then the following statements hold:

1) nodes agree on a solution $\mathbf{x}_{\text{sol}}$ in finite time;
2) the solution $\mathbf{x}_{\text{sol}}$ satisfies

$$\begin{aligned} &\mathbb{P}^{M}\left\{\mathbf{q}\in\mathbb{Q}^{M} : \mathbb{P}\left\{q\in\mathbb{Q} : \mathbf{x}_{\text{sol}} \notin \bigcap_{i=1}^{n}\mathcal{F}^i(q)\right\} \leq \varepsilon\right\} \\ &\geq 1-\delta \end{aligned} \tag{5}$$

where $M = \sum_i M^i_{k_i}$.

Proof: First, note that if the Algorithm does not stop due to Condition 2 of Theorem 3.5, then all nodes stop generating new samples in finite time. When all nodes have stopped generating samples, they start solving the following deterministic optimization problem:

$$\begin{aligned} &\min\ c^T\mathbf{x} \\ &\text{subject to } \mathbf{x} \in \bigcap_{i=1}^{n}\bigcap_{\ell=1}^{M^i_{k_i}} \mathcal{F}^i(q^{(\ell)}), \\ &\mathbf{x} \in S \end{aligned} \tag{6}$$

where $M^i_{k_i} \geq M_i^{\text{scen}}$ is the cardinality of the verification multisample at the stopping point. To prove that nodes agree in finite time on an optimal solution of (6), we resort to the following two arguments: first, as from Statement 1 of Theorem 3.5, the cost is monotonically nondecreasing; second, the number of constraints in the network is fixed, and, thus, the number of possible bases is finite. The proof of the first statement, thus, follows from arguments analogous as those in [9]. Indeed, when nodes stop generating samples, the randomized constraints

TABLE I
AVERAGE—OVER ALL NODES—NUMBER OF TIMES A BASIS IS TRANSMITTED TO THE NEIGHBORS, AVERAGE NUMBER OF TIMES VERIFICATION IS PERFORMED (k_i AT THE CONVERGENCE), AND EMPIRICAL VIOLATION OF THE COMPUTED SOLUTION ($\mathbf{x}_{\text{SOL}}$) OVER 10 000 RANDOM SAMPLES FOR DIFFERENT NUMBER OF NODES AND NEIGHBORS IN EACH NODE. THE SIMULATION IS PERFORMED 100 TIMES FOR EACH ROW AND AVERAGE RESULTS ARE REPORTED

# Nodes n	# Neighbors in each node (degree)	Graph diameter	# Constraints in each node	# Transmissions (averaged)	# Verifications (averaged)	Empirical violation
10	3	4	100	29.86	15.6	8.18×10^{-4}
50	4	4	100	37.58	10.73	2×10^{-4}
100	6	4	100	41.73	9.79	3×10^{-4}
200	8	4	100	43.94	8.92	2.3×10^{-4}

consensus algorithm becomes deterministic and turns out to be a variant of the constraints consensus algorithm proposed in [9]. Due to Assumption 2.6, nodes agree on a $\mathbf{x}_{\text{sol}}$, which is the unique optimal solution of problem (6). Thus, the second statement follows by noting that, from Lemma 3.7, $\mathbf{x}_{\text{sol}}$ satisfies (5) since the number of random constraints at each node $M^i_{k_i}$ is greater than or equal to the scenario bound M^{scen}_i. ■

IV. NUMERICAL SIMULATIONS

We test the effectiveness of the distributed algorithm presented in Section III through extensive numerical simulations. To this end, we consider two different problems: 1) randomly generated MILP with uncertain parameters and 2) distributed convex position estimation in wireless sensor networks. These two numerical simulations are discussed in the next sections.

Numerical computations are run on a Linux-based high performance computing cluster[2] with 256 CPUs. Each node uses only one CPU of the cluster and executes Algorithm 1 in an independent MATLAB environment. The communication is modeled by sharing files between different Matlab environments over a fixed digraph.

A. Uncertain Mixed-Integer Linear Programming

We randomly generate robust MILPs with the following structure:

$$\begin{aligned} &\min \ c^T\mathbf{x} \\ &\text{subject to } (A_i^0 + A_i^q)^T\mathbf{x} \leq b_i, \quad i = 1, \ldots, n \\ &\mathbf{x} \in \mathbb{Z}^2 \times \mathbb{R}^3 \end{aligned}$$

where A_i^0 is a fixed (nominal) matrix and A_i^q is an interval matrix—a matrix whose entries are bounded in given intervals—defining the uncertainty in the optimization problem. We follow the methodology presented in [31] in order to generate the pair A_i^0, b_i and the objective direction c so that the generated linear program remains feasible. To ensure feasibility of the MILP, we increase the volume of the feasible region using the parameter $\gamma > 1$. In the set of simulations reported here, we set $\gamma = 20$. The communication graph $\mathcal{G}$ is a (connected) random k-nearest neighbor graph (with k being the fixed degree or number of neighbors) [32]. Over this fixed graph, agents implement the distributed algorithm according to the asynchronous and unreliable communication protocol (based on MATLAB environments) described above. In particular, only one node at a time is able to read/write from/to each node file containing its basis. Hence, if more than one node is willing to read/write from/to a file, only one would be allowed to do so. This gives rise to an asynchronous and unreliable communication protocol. We use the `intlinprog` function of Mosek [33] to solve optimization problems appearing at each iteration of the distributed algorithm.

For the computations reported in Table I, we varied the number of nodes and neighbors per node, while using only those graphs having a diameter equal to 4. The number of constraints per node is set to 100. We also consider all elements of A_q to be bounded in $[-0.2, 0.2]$. The underlying probability distribution of uncertainty appearing in A_q is selected to be uniform (see, e.g., [34]). The probabilistic accuracy and confidence levels of each agent are $\varepsilon_i = 0.1/n$ and $\delta_i = 10^{-9}/n$, respectively, with n being the number of nodes (first column of Table I). It is assumed that each node keeps the latest information received from neighbors and, hence, if the basis is not updated, there is no need to retransmit it to the neighbors. This also accounts for the asynchronicity of the distributed algorithm. The results presented in Table I shows that with a relatively small number of "design" samples used in step O_2 of Algorithm 1, nodes compute a solution with a high degree of robustness. In order to examine the robustness of the obtained solution, we run an *a posteriori* analysis based on Monte Carlo simulations. To this end, we check the feasibility of the obtained solution for 10 000 random samples extracted from the uncertain sets of the nodes. The empirical violation is measured by dividing the number of samples that violate the solution by 10 000. Since Algorithm 1 has a stochastic nature, for each row of Table I, we generate randomly 100 problems, solve them using Algorithm 1, run an *a posteriori* analysis, and then report the average values.

In Fig. 1, we report the distance of objective value $J(\mathcal{B}^i(t))$ and candidate solutions $\mathbf{x}^i(t) \forall i \in \{1, \ldots, n\}$ from $J(\mathcal{B}_{\text{sol}})$ and $\mathbf{x}_{\text{sol}}$, respectively, along the distributed algorithm execution for a problem instance corresponding to the last row of Table I.

In the following computations, we test the effect of using more violation certificates in the optimization step (see Remark 3.3). Specifically, we use in the optimization step up to 10 violating certificates obtained from the verification step. As expected, this decreases the number of communications

[2]http://idc.sutd.edu.sg/titan/

TABLE II

AVERAGE—OVER ALL NODES—NUMBER OF TIMES A BASIS IS TRANSMITTED TO THE NEIGHBORS, AVERAGE NUMBER OF TIMES VERIFICATION IS PERFORMED (k_i AT THE CONVERGENCE), AND EMPIRICAL VIOLATION OF THE COMPUTED SOLUTION ($\mathbf{x}_{\text{SOL}}$) OVER 10 000 RANDOM SAMPLES FOR DIFFERENT NUMBER OF NODES AND NEIGHBORS IN EACH NODE. WE ALLOW TEN VIOLATION SAMPLES TO BE RETURNED BY THE VERIFICATION STEP. THE SIMULATION IS PERFORMED 100 TIMES FOR EACH ROW AND AVERAGE RESULTS ARE REPORTED

# Nodes n	# Neighbors in each node (degree)	Graph diameter	# Constraints in each node	# Transmissions (averaged)	# Verifications (averaged)	Empirical violation
10	3	4	100	16.1	8.1	10×10^{-3}
50	4	4	100	24.37	7.05	9×10^{-3}
100	6	4	100	26.25	6.3	4×10^{-3}

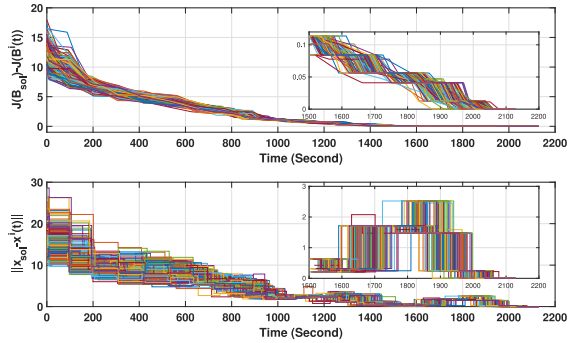


Fig. 1. Top: Distance to $J(\mathcal{B}_{\text{sol}})$ which is the objective value algorithm converges to. Bottom: Distance to $\mathbf{x}_{\text{sol}}$ which is the solution algorithm converges to. The two plots are generated for a problem instance corresponding to the last row of Table I.

required for convergence at the expense of the local optimization complexity. This result is reported in Table II.

In all 700 simulations reported in Tables I and II, Algorithm 1 converges to a solution with desired probabilistic robustness in finite time. We further remark that in none of the simulations we ran, the stopping criterion derived in Section III-C was met. In fact, a closer investigation of the stopping condition presented in Lemma 3.8—namely $M^i_{k_i} \geq M^{\text{scen}}_i$ for all $i \in \{1, \ldots, n\}$—reveals that this condition happens only for extremely large values of the verification counter k_i. In [23, Theor. 4], an analytical suboptimal solution for the sample size M^{scen}_i in the inequality (4) is provided

$$M^{\text{scen}}_i \geq \frac{1.582}{\varepsilon}\left(\ln\frac{1}{\delta_i} + h(s) - 2\right). \tag{7}$$

Solving the inequality $M^i_{k_i} \geq M^{\text{scen}}_i$ for k_i and noting that $\ln(\frac{1}{1-\varepsilon_i}) \simeq \varepsilon_i$ for small ε_i, we obtain

$$k_i \geq \exp\left(\frac{0.58\ln\frac{1}{\delta_i} + 1.58(h(S) - 2) - 2.3}{1.1}\right).$$

The Helly number associated with the MILP envisaged here is 16. Considering, for example, $\varepsilon_i = 0.01$ and $\delta_i = 10^{-10}$, the verification counter ensuring that $M^i_{k_i} \geq M^{\text{scen}}_i$ becomes $k_i \geq 1.2 \times 10^{13}$. By looking at the sixth column of Tables I and II, which indicates the value of verification counter at convergence, one can see that the distributed algorithm converges to a solution with desired probabilistic properties in a much smaller number of verification steps.

B. Distributed Localization in Wireless Sensor Networks

The second numerical example is a problem of distributed position estimation in wireless sensor networks. A centralized version of this problem—with no uncertainty—is formulated in [35]. Consider a two-dimensional[3] space containing a number of heterogeneous wireless sensors which are randomly placed over a given environment. The sensors are of two classes. The first class is wireless sensors with known positions (SwKP) which are capable of positioning themselves up to a given accuracy, i.e., with some uncertainty in their position. They play the role of computational nodes in the distributed optimization framework. These sensors can communicate with each other based on a metric distance. The second class is a wireless sensor with unknown position (SwUP) which has no positioning capabilities. This sensor is only equipped with a short-range transmitter having an isotropic communication range. We further consider a heterogeneous setup in which half of the SwKPs are equipped with a laser transceiver providing an estimate of the relative angle with respect to the SwUP, which is within the range of the laser transceiver. This "angular constraint" can be represented by the intersection of three half-spaces $\mathcal{F}^i_A \doteq \{\mathbf{x} \in \mathbb{R}^2 : a_k\mathbf{x} - b_k \leq 0, k = 1, 2, 3\}$, where $\mathbf{x} \in \mathbb{R}^2$ is the position of SwUP and parameters $a_k \in \mathbb{R}^{1\times 2}$ and $b_k \in \mathbb{R}$ define the three half-spaces as presented in Fig. 2. SwKPs, which are not equipped with laser transceiver, can place a "radial constraint" for the SwUP if it is within the communication range. This constraint can be formulated as $\|\mathbf{x} - p_i\|_2 \leq r$, where $p_j \in \mathbb{R}^2$ is the position of ith SwKP. Using the Schur-complement [36], the feasible set of radial constraint can be represented as

$$\mathcal{F}^i_R \doteq \left\{\mathbf{x} \in \mathbb{R}^2 : \begin{bmatrix} rI_2 & (\mathbf{x} - p_i) \\ (\mathbf{x} - p_i)^T & r \end{bmatrix} \succeq 0\right\}$$

where I_2 is the 2×2 identity matrix and $\succeq$ denotes positive semidefiniteness of the matrix.

The objective of the networked system is to find the smallest box $\{\mathbf{x} \in \mathbb{R}^2 : [\mathbf{x}^l_x, \mathbf{x}^l_y] \leq \mathbf{x} \leq [\mathbf{x}^u_x, \mathbf{x}^u_y]\}$ that is guaranteed to contain the SwUP, i.e., the red dotted box in Fig. 2. The bounding box can be computed by solving four optimization problems with linear objectives. For instance, $\mathbf{x}^l_x$ can be obtained by solving the following optimization problem:

$$\mathbf{x}^l_x = \text{argmin} \begin{bmatrix} 1 \\ 0 \end{bmatrix}^T \mathbf{x}$$

[3]Extension to a three-dimensional space is straightforward.

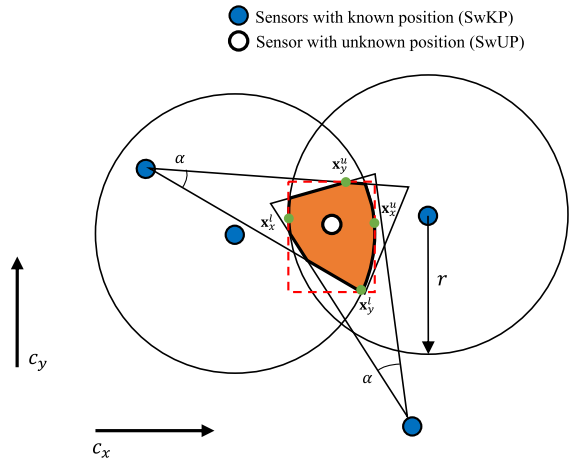


Fig. 2. Sensors with known position (SwKP: blue dots) are computational nodes in the distributed optimization setup. Their objective is to estimate the position of the sensor with unknown position (SwUP: white dot). The orange (shaded) region represents the feasible set for the unknown position.

$$\text{subject to } \mathbf{x} \in \bigcap_{i=1}^{n} \left(\mathcal{F}_A^i \cap \mathcal{F}_R^i \right)$$

where n is the number of SwKPs capable of constraining the position of the SwUP. We point out that for SwKPs that are not equipped with a laser transceiver, $\mathcal{F}_A^i = \mathbb{R}^2$ holds.

There is an uncertainty associated with the position vector reported by SwKPs. This uncertainty can be modeled as a norm-bounded vector

$$\|p_i - \bar{p}_i\|_2 \leq \rho \tag{8}$$

where $\bar{p}_i$ is the nominal position reported by sensor i, p_i is its actual position, and ρ is the radius of the uncertain set.

For simulation purposes, we consider a 10×10 square environment containing n SwKPs (computational nodes) whose purpose is to estimate the position of the SwUP. The communication range of all sensors is considered to be 7 units. Finally, the uncertainty radius ρ in (8) is 0.1 and the distribution of the uncertainty is selected to be uniform. The probabilistic accuracy and confidence parameters are selected to be $\varepsilon_i = 0.1/n$ and $\delta_i = 10^{-9}/n$ for all $i = 1, \ldots, n$, leading to $\varepsilon = 0.1$ and $\delta = 10^{-9}$. The distributed robust localization problem is solved for different values of the number of SwKPs n. The result is reported in Table III .

To conclude, we would like to point out an interesting feature of Algorithm 1. In the results presented in Tables I–III, one can observe that when the number of computational nodes increases while keeping the graph diameter constant, the number of transmissions required for convergence does not change significantly. This suggests that the number of transmissions required for convergence is independent from the number of computational nodes but rather depends on the graph diameter.

TABLE III
AVERAGE—OVER ALL NODES—NUMBER OF TIMES A BASIS IS TRANSMITTED TO THE NEIGHBORS, AVERAGE NUMBER OF TIMES VERIFICATION IS PERFORMED (k_i AT THE CONVERGENCE), AND EMPIRICAL VIOLATION OF THE COMPUTED SOLUTION ($\mathbf{x}_{\text{SOL}}$) OVER 10 000 RANDOM SAMPLES FOR DIFFERENT NUMBER OF KNOWN SENSORS

# Nodes n	# Transmissions (averaged)	# Verifications (averaged)	Empirical violation
10	7.77	8.77	1×10^{-4}
50	12.48	7.36	0
100	9.37	6.52	0
200	7.95	7.42	0

V. CONCLUSION

In this article, we proposed a randomized distributed algorithm for solving RMICPs in which constraints are scattered across a network of processors communicating by means of a directed time-varying graph. The distributed algorithm has a sequential nature consisting of two main steps: 1) verification; 2) optimization. Each processor iteratively verifies a candidate solution through a Monte Carlo simulation and solves a local MICP whose constraint set includes its current basis, the collection of bases from neighbors, and, possibly, a constraint—provided by the Monte Carlo algorithm—violating the candidate solution. The two steps, i.e., verification and optimization, are repeated until a local stopping criterion is met and all nodes converge to a common solution. We analyzed the convergence properties of the proposed algorithm.

APPENDIX
PROOF OF THEOREM 3.5

Proof of Statement 1: The proof of the first statement follows arguments very similar to those in [11] and is thus omitted.

Proof of Statement 2: Since the graph is uniformly jointly strongly connected, for any pair of nodes u and v and for any $t_0 > 0$, there exists a time-dependent path from u to v [37]—a sequence of nodes $\ell_1, \ldots, \ell_k$ and a sequence of time instances $t_1, \ldots, t_{k+1}$ with $t_0 \leq t_1 < \cdots < t_{k+1}$ such that the directed edges $\{(u, \ell_1), (\ell_1, \ell_2), \ldots, (\ell_k, v)\}$ belong to the directed graph at time instances $\{t_1, \ldots, t_{k+1}\}$, respectively. We now prove that the path from u to v is of length at most nL. We recall that n is the number of nodes and L is defined in Assumption 2.7. To this end, following [37], define a time-varying set S_t and initialize it with the node u itself, i.e., $S_{t_0} = u$. Next, given S_t, construct S_{t+1} by adding all nodes ℓ_{t+1} such that $\ell_{t+1} \in \mathcal{N}_{\text{out}}(\ell_t, t+1)$ with ℓ_t being all the nodes in the set S_t. Now, note that if S_t does not contain all nodes, i.e., $S_t \neq \mathcal{V}$, then after L time instants, at least one new node must be added to the set S_t, otherwise taking the union graph over the L time instants, the set S_t and that node would not be connected by any path, thus violating Assumption 2.7. Therefore, in at most nL time instants, S_t will contain all nodes including v. It means that the length of time-varying path connecting u and v is at most nL. Consider nodes i and p. If $\ell_1 \in \mathcal{N}_{\text{out}}(i, t_0)$, then $J(\mathcal{B}^i(t_0)) \leq$

$J(\mathcal{B}^{\ell_1}(t_0+1))$ as the constraint set of node ℓ_1 at time t_0+1 is a superset of the constraint set of node i at time t_0. Iterating this argument, we obtain $J(\mathcal{B}^i(t_0)) \leq J(\mathcal{B}^p(t_0+nL))$. Again since the graph is uniformly jointly strongly connected, there will be a time-varying path of length at most nL from node p to node i. Therefore

$$J(\mathcal{B}^i(t_0)) \leq J(\mathcal{B}^p(t_0+nL)) \leq J(\mathcal{B}^i(t_0+2nL)).$$

If $J(\mathcal{B}^i(t_0)) = J(\mathcal{B}^i(t_0+2nL))$ and considering the point that node p can be any node of the graph, then all nodes have the same cost. That is, $J(\mathcal{B}^1(t)) = \cdots = J(\mathcal{B}^n(t))$. This combined with Assumption 2.6 proves Statement 2.

Proof of Statement 3: We start by defining the following violation sets:

$$\text{Viol}_i := \left\{ q \in \mathbb{Q} :\ \mathbf{x}_{\text{sol}} \notin \mathcal{F}^i(q) \right\}, \qquad i = 1, \ldots, n,$$

$$\text{Viol} := \left\{ q \in \mathbb{Q} :\ \mathbf{x}_{\text{sol}} \notin \bigcap_{i=1}^{n} \mathcal{F}^i(q) \right\}.$$

We observe that

$$\mathbf{x}_{\text{sol}} \notin \bigcap_{i=1}^{n} \mathcal{F}^i(q) \iff \exists i = 1, \ldots, n :\ \mathbf{x}_{\text{sol}} \notin \mathcal{F}^i(q).$$

Hence

$$\text{Viol} = \bigcup_{i=1}^{n} \text{Viol}_i.$$

Therefore

$$\mathbb{P}\{\text{Viol}\} = \mathbb{P}\left\{ \bigcup_{i=1}^{n} \text{Viol}_i \right\} \leq \sum_{i=1}^{n} \mathbb{P}\{\text{Viol}_i\}.$$

At this point, we note that, applying simple properties coming from the probability theory, we have

$$\begin{aligned}
&\mathbb{P}_\infty\left\{ \mathbf{q} \in \mathbb{S}_{\text{sol}} :\ \mathbb{P}\left\{ q \in \mathbb{Q} : \mathbf{x}_{\text{sol}} \notin \bigcap_{i=1}^{n} \mathcal{F}^i(q) \right\} \leq \epsilon \right\} \\
&\quad = \mathbb{P}_\infty\left\{ \mathbf{q} \in \mathbb{S}_{\text{sol}} :\ \mathbb{P}\{\text{Viol}\} \leq \epsilon \right\} \\
&\quad \geq \mathbb{P}_\infty\left\{ \mathbf{q} \in \mathbb{S}_{\text{sol}} :\ \sum_{i=1}^{n} \mathbb{P}\{\text{Viol}_i\} \leq \epsilon \right\} \\
&\quad = \mathbb{P}_\infty\left\{ \mathbf{q} \in \mathbb{S}_{\text{sol}} :\ \sum_{i=1}^{n} \mathbb{P}\{\text{Viol}_i\} \leq \sum_{i=1}^{n} \epsilon_i \right\} \\
&\quad \geq \mathbb{P}_\infty\left\{ \bigcap_{i=1}^{n} \left\{ \mathbf{q} \in \mathbb{S}_{\text{sol}} :\ \mathbb{P}\{\text{Viol}_i\} \leq \epsilon_i \right\} \right\} \\
&\quad \geq 1 - \sum_{i=1}^{n} \mathbb{P}_\infty\left\{ \mathbf{q} \in \mathbb{S}_{\text{sol}} :\ \mathbb{P}\{\text{Viol}_i\} > \epsilon_i \right\}.
\end{aligned}$$

As a consequence, if we prove that

$$\mathbb{P}_\infty\{\mathbf{q} \in \mathbb{S}_{\text{sol}} :\ \mathbb{P}\{\text{Viol}_i\} > \epsilon_i\} \leq \delta_i, \qquad i = 1, \ldots, n \quad (9)$$

then the thesis will follow.

Now, let $i = 1, \ldots, n$ and $k_i \in \mathbb{N}$. We define the events:

$$\text{ExitBad}_i := \{\mathbf{q} \in \mathbb{S}_{\text{sol}} :\ \mathbb{P}\{\text{Viol}_i\} > \epsilon_i\},$$

$$\text{Feas}^i_{k_i} := \left\{ \mathbf{q} \in \mathbb{S}_{\text{sol}} :\ \mathbf{x}_{\text{sol}} \in \mathcal{F}^i(q^{(\ell)}_{k_i,i}) \ \ \forall \ell = 1, \ldots, M^i_{k_i} \right\}.$$

We immediately note that (9) is equivalent to

$$\mathbb{P}_\infty\{\text{ExitBad}_i\} \leq \delta_i, \qquad i = 1, \ldots, n. \quad (10)$$

Thus, we will prove (10) instead of (9).

First, we observe that if $\mathbf{q} \in \text{ExitBad}_i$, then there exists k_i such that $\mathbf{q} \in \text{Feas}^i_{k_i}$. Consequently, the following relation holds:

$$\text{ExitBad}_i \subset \bigcup_{k_i=1}^{\infty} \left(\text{Feas}^i_{k_i} \cap \text{ExitBad}_i \right).$$

Therefore, we have

$$\mathbb{P}_\infty\{\text{ExitBad}_i\} \leq \sum_{k_i=1}^{\infty} \mathbb{P}_\infty\left\{ \text{Feas}^i_{k_i} \cap \text{ExitBad}_i \right\} \quad (11)$$

and we can bound the terms of the last series as follows:

$$\begin{aligned}
&\mathbb{P}_\infty\left\{ \text{Feas}^i_{k_i} \cap \text{ExitBad}_i \right\} \\
&\quad = \mathbb{P}_\infty\left\{ \text{Feas}^i_{k_i} \middle| \text{ExitBad}_i \right\} \mathbb{P}_\infty\{\text{ExitBad}_i\} \quad (12) \\
&\quad \leq \mathbb{P}_\infty\left\{ \text{Feas}^i_{k_i} \middle| \text{ExitBad}_i \right\} \quad (13) \\
&\quad = \mathbb{P}_\infty\Bigg\{ \mathbf{q} \in \mathbb{S}_{\text{sol}} :\ \mathbf{x}_{\text{sol}} \in \mathcal{F}^i(q^{(\ell)}_{k_i,i}) \ \ \forall \ell = 1, \ldots, M^i_{k_i} \\
&\qquad\qquad \Bigg| \ \mathbb{P}\left\{ q \in \mathbb{Q} : \mathbf{x}_{\text{sol}} \notin \mathcal{F}^i(q) \right\} > \epsilon_i \Bigg\} \quad (14) \\
&\quad = \mathbb{P}_\infty\Bigg\{ \mathbf{q} \in \mathbb{S}_{\text{sol}} :\ \mathbf{x}_{\text{sol}} \in \mathcal{F}^i(q^{(\ell)}_{k_i,i}) \ \ \forall \ell = 1, \ldots, M^i_{k_i} \\
&\qquad \Bigg| \ \mathbb{P}\left\{ q \in \mathbb{Q} : \mathbf{x}_{\text{sol}} \in \mathcal{F}^i(q) \right\} \leq 1 - \epsilon_i \Bigg\} \quad (15) \\
&\quad \leq (1-\epsilon_i)^{M^i_k}. \quad (16)
\end{aligned}$$

A few comments are given in order to explain the above set of equalities and inequalities: (12) follows from the chain rule; inequality (13) is due to the fact that $\mathbb{P}_\infty \leq 1$; in (14), we have written explicitly the events $\text{Feas}^i_{k_i}$ and ExitBad_i; (15) exploits the property $\mathbb{P}\{\mathbb{Q} - A\} = 1 - \mathbb{P}\{A\}$; inequality (16) is due to the fact that in order to declare $\mathbf{x}_{\text{sol}}$ feasible, the algorithm needs to perform $M^i_{k_i}$ independent successful trials each having a probability of success smaller than or equal to $(1-\epsilon_i)$.

At this point, combining (11) and (16), we obtain

$$\mathbb{P}_\infty\{\text{ExitBad}_i\} \leq \sum_{k=1}^{\infty} (1-\epsilon_i)^{M^i_{k_i}}$$

and the last series can be made arbitrarily small by an appropriate choice of $M^i_{k_i}$. In particular, by choosing

$$(1-\epsilon_i)^{M^i_{k_i}} = \frac{1}{k_i^\alpha} \frac{1}{\xi(\alpha)} \delta_i \quad (17)$$

where $\xi(\alpha) := \sum_{s=1}^{\infty} \frac{1}{s^\alpha}$ is the Riemann zeta function evaluated at $\alpha > 1$, we have

$$\sum_{k_i=1}^{\infty} (1-\epsilon_i)^{M_{k_i}^i} = \sum_{k_i=1}^{\infty} \frac{1}{k_i^\alpha} \frac{1}{\xi(\alpha)} \delta_i = \frac{1}{\xi(\alpha)} \delta_i \sum_{k_i=1}^{\infty} \frac{1}{k_i^\alpha} = \delta_i.$$

Hence, the choice of sample size is obtained by solving (17) for $M_{k_i}^i$:

$$M_{k_i}^i = \left\lceil \frac{\alpha \log(k_i) + \log(\xi(\alpha)) + \log\left(\frac{1}{\delta_i}\right)}{\log\left(\frac{1}{1-\epsilon_i}\right)} \right\rceil.$$

Note that the optimal value of α minimizing $M_{k_i}^i$ is computed through empirical evaluation to be $\alpha = 1.1$ (see, e.g., [25] for more details). By means of this choice of α, we obtain the number $M_{k_i}^i$ specified in the algorithm, and we can, thus, conclude that (10) is true, and so the thesis follows.

Proof of Statement 4: Define the following two sets:

$$V \doteq \left\{ q \in \mathbb{Q} : \mathbf{x}_{\text{sol}} \notin \bigcap_{i=1}^{n} \mathcal{F}^i(q) \right\}$$

$$W \doteq \left\{ q \in \mathbb{Q} : J\left(\mathcal{B}_{\text{sol}} \cap \mathcal{F}(q)\right) > J(\mathcal{B}_{\text{sol}}) \right\}$$

where $\mathcal{F}(q) \doteq \bigcap_{i=1}^{n} \mathcal{F}^i(q)$. We first prove that $V = W$.

Let $q_v \in V$, that is, $\mathbf{x}_{\text{sol}} \notin \bigcap_{i=1}^{n} \mathcal{F}^i(q_v)$, then $J(\mathcal{B}_{\text{sol}} \cap \mathcal{F}(q_v)) \geq J(\mathcal{B}_{\text{sol}})$ with $\mathcal{F}(q_v) \doteq \bigcap_{i=1}^{n} \mathcal{F}^i(q_v)$. Also, due to Assumption 2.6, any subproblem of (1) has a unique minimum point, and, hence, $J(\mathcal{B}_{\text{sol}} \cap \mathcal{F}(q_v)) \neq J(\mathcal{B}_{\text{sol}})$. Hence, $q_v \in W$ and subsequently $V \subseteq W$.

Now let $q_v \in W$, that is, $J(\mathcal{B}_{\text{sol}} \cap \mathcal{F}(q_v)) > J(\mathcal{B}_{\text{sol}})$, and suppose by contradiction that $\mathbf{x}_{\text{sol}} \in \mathcal{F}(q_v)$, i.e., $q_v \notin V$. Considering the fact that $\mathcal{B}_{\text{sol}}$ is the basis corresponding to $\mathbf{x}_{\text{sol}}$, we conclude that $\mathbf{x}_{\text{sol}} \in \mathcal{B}_{\text{sol}} \cap \mathcal{F}(q_v)$ implying that $J(\mathcal{B}_{\text{sol}} \cap \mathcal{F}(q_v)) \leq J(\mathcal{B}_{\text{sol}})$. This contradicts the fact that $J(\mathcal{B}_{\text{sol}} \cap \mathcal{F}(q_v)) > J(\mathcal{B}_{\text{sol}})$. Hence, $q_v \in V$ and, subsequently, $W \subseteq V$. Since $V \subseteq W$ and $W \subseteq V$, therefore, $V = W$. This argument combined with the result of Statement 4 proves the fifth statement of the theorem. That is, the probability that the solution $\mathbf{x}_{\text{sol}}$ is no longer optimal for a new sample equals the probability that the solution is violated by the new sample.

REFERENCES

[1] M. Chamanbaz, G. Notarstefano, and R. Bouffanais, "Randomized constraints consensus for distributed robust linear programming," in *Proc. 20th IFAC World Congr.*, 2017, pp. 4973–4978.

[2] L. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust Optimization*. Princeton, NJ, USA: Princeton University Press, 2009.

[3] S. Lee and A. Nedić, "Distributed random projection algorithm for convex optimization," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 2, pp. 221–229, Apr. 2013.

[4] S. Lee and A. Nedić, "Asynchronous gossip-based random projection algorithms over networks," *IEEE Trans. Autom. Control*, vol. 61, no. 4, pp. 953–968, Apr. 2016.

[5] Z. J. Towfic and A. H. Sayed, "Adaptive penalty-based distributed stochastic convex optimization," *IEEE Trans. Signal Process.*, vol. 62, no. 15, pp. 3924–3938, Aug. 2014.

[6] L. Carlone, V. Srivastava, F. Bullo, and G. Calafiore, "Distributed random convex programming via constraints consensus," *SIAM J. Control Optim.*, vol. 52, pp. 629–662, 2014.

[7] G. Calafiore and M. Campi, "Uncertain convex programs: Randomized solutions and confidence levels," *Math. Program.*, vol. 102, pp. 25–46, 2004.

[8] G. Calafiore and M. Campi, "The scenario approach to robust control design," *IEEE Trans. Autom. Control*, vol. 51, no. 5, pp. 742–753, May 2006.

[9] G. Notarstefano and F. Bullo, "Distributed abstract optimization via constraints consensus: Theory and applications," *IEEE Trans. Autom. Control*, vol. 56, no. 10, pp. 2247–2261, Oct. 2011.

[10] K. You and R. Tempo, *Networked Parallel Algorithms for Robust Convex Optim. via Scenario Approach*. Cham, Springer, 2018, pp. 341–353.

[11] M. Bürger, G. Notarstefano, and F. Allgöwer, "A polyhedral approximation framework for convex and robust distributed optimization," *IEEE Trans. Autom. Control*, vol. 59, no. 2, pp. 384–395, Feb. 2014.

[12] K. Margellos, A. Falsone, S. Garatti, and M. Prandini, "Distributed constrained optimization and consensus in uncertain networks via proximal minimization," *IEEE Trans. Autom. Control*, vol. 63, no. 5, pp. 1372–1387, May 2018.

[13] G. Calafiore, D. Lyons, and L. Fagiano, "On mixed-integer random convex programs," in *Proc. 51st IEEE Annu. Conf. Decis. Control (CDC)*, 2012, pp. 3508–3513.

[14] R. Vujanic, P. M. Esfahani, P. J. Goulart, S. Mariéthoz, and M. Morari, "A decomposition method for large scale MILPs, with performance guarantees and a power system application," *Automatica*, vol. 67, pp. 144–156, 2016.

[15] A. Falsone, K. Margellos, and M. Prandini, "A decentralized approach to multi-agent MILPs: Finite-time feasibility and performance guarantees," *Automatica*, vol. 103, pp. 141–150, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109819300093

[16] A. Falsone, K. Margellos, and M. Prandini, "A distributed iterative algorithm for multi-agent MILPs: Finite-time feasibility and performance characterization," *IEEE Control Syst. Lett.*, vol. 2, no. 4, pp. 563–568, Oct. 2018.

[17] A. Camisa, I. Notarnicola, and G. Notarstefano, "A primal decomposition method with suboptimality bounds for distributed mixed-integer linear programming," in *Proc. IEEE Conf. Decis. Control (CDC)*, 2018, pp. 3391–3396.

[18] S. J. Kim and G. B. Giannakis, "Scalable and robust demand response with mixed-integer constraints," *IEEE Trans. Smart Grid*, vol. 4, no. 4, pp. 2089–2099, Dec. 2013.

[19] M. Franceschelli, D. Rosa, C. Seatzu, and F. Bullo, "Gossip algorithms for heterogeneous multi-vehicle routing problems," *Nonlinear Anal.: Hybrid Syst.*, vol. 10, pp. 156–174, 2013.

[20] Y. Kuwata and J. P. How, "Cooperative distributed robust trajectory optimization using receding horizon MILP," *IEEE Trans. Control Syst. Technol.*, vol. 19, no. 2, pp. 423–431, Mar. 2011.

[21] A. Testa, A. Rucco, and G. Notarstefano, "Distributed mixed-integer linear programming via cut generation and constraint exchange," *IEEE Trans. Autom. Control*, vol. 65, no. 4, pp. 1–12, Apr. 2019.

[22] M. Chamanbaz, F. Dabbene, R. Tempo, V. Venkataramanan, and Q. G. Wang, "Sequential randomized algorithms for convex optimization in the presence of uncertainty," *IEEE Trans. Autom. Control*, vol. 61, no. 9, pp. 2565–2571, Sep. 2016.

[23] T. Alamo, R. Tempo, A. Luque, and D. Ramirez, "Randomized methods for design of uncertain systems: Sample complexity and sequential algorithms," *Automatica*, vol. 52, pp. 160–172, 2015.

[24] G. Calafiore and F. Dabbene, "A probabilistic analytic center cutting plane method for feasibility of uncertain LMIs," *Automatica*, vol. 43, pp. 2022–2033, 2007.

[25] F. Dabbene, P. Shcherbakov, and B. Polyak, "A randomized cutting plane method with probabilistic geometric convergence," *SIAM J. Optim.*, vol. 20, pp. 3185–3207, 2010.

[26] J. De Loera, R. La Haye, D. Oliveros, and E. Roldán-Pensado, "Chance-constrained convex mixed-integer optimization and beyond: Two sampling algorithms within S-optimization," *J. Convex Analysis*, vol. 25, no. 1, pp. 201–218, 2018.

[27] E. Helly, "Über mengen konvexer körper mit gemeinschaftlichen punkte," *Jahresbericht Deutschen Mathematiker-Vereinigung*, vol. 32, pp. 175–176, 1923.

[28] N. Amenta, "Helly-type theorems and generalized linear programming," *Discrete Comput. Geometry*, vol. 12, no. 3, pp. 241–261, 1994.

[29] N. Amenta, J. De Loera, and P. Soberòn, "Helly's theorem: New variations and applications," *Algebr. geometr. method. discrete mathemat.*, vol. 685, pp. 55–95, 2017.

[30] G. Averkov and R. Weismantel, "Transversal numbers over subsets of linear spaces," *Adv. Geometry*, vol. 12, no. 1, pp. 19–28, 2012.

[31] J. Dunham, D. Kelly, and J. Tolle, "Some experimental results concerning the expected number of pivots for solving randomly generated linear programs," Oper. Res. Syst. Analysis Dep., Univ. North Carolina, Chapel Hill, USA, Tech. Rep. TR 77–16, 1977.
[32] A. Frieze and M. Karoński, *Introduction to Random Graphs*. Cambridge, U.K.: Cambridge University Press, 2015.
[33] E. Andersen and K. Andersen, "The MOSEK interior point optimizer for linear programming: An implementation of the homogeneous algorithm," in *High Performance Optimization*. Berlin, Germany: Springer, 2000, pp. 197–232.
[34] E. Bai, R. Tempo, and M. Fu, "Worst-case properties of the uniform distribution and randomized algorithms for robustness analysis," *Math. Control, Signals, Syst.*, vol. 11, pp. 183–196, 1998.
[35] L. Doherty, K. S. J. Pister, and L. E. Ghaoui, "Convex position estimation in wireless sensor networks," in *Proc. 20th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 3, 2001, pp. 1655–1663.
[36] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge University Press, 2004.
[37] J. Hendrickx, "Graphs and networks for the analysis of autonomous agent systems," Ph.D. dissertation, Mathematical Engineering Dept., Univ. Catholique Louvain, Louvain, Belgium, 2008.

Mohammadreza Chamanbaz (Member, IEEE) received the B.Sc. degree in electrical engineering from the Shiraz University of Technology, Shiraz, Iran, in 2008. He received the Ph.D. degree in control science from the Department of Electrical and Computer Engineering, National University of Singapore, Singapore, in 2014.

He was with Data Storage Institute, Singapore, as a Research Scholar from 2010 to 2014. From 2014 to 2017, he was a Postdoctoral Research Fellow with the Singapore University of Technology and Design, Singapore. He was an Assistant Professor with the Arak University of Technology, Arak, Iran, from 2017 to 2019. He is currently a Senior Research Fellow with the iTrust Center for Research in Cyber Security, Singapore. His research activities are mainly focused on probabilistic and randomized algorithms for analysis and control of uncertain systems, robust and distributed optimization, and secure control of cyber–physical systems.

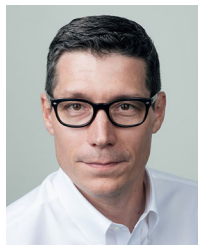
Giuseppe Notarstefano (Member, IEEE) received the Laurea degree (summa cum laude) in electronics engineering from the Università di Pisa, Pisa, Italy, in 2003, and the Ph.D. degree in automation and operation research from the Università di Padova, Padua, Italy, in 2007.

He is a Professor with the Department of Electrical, Electronic, and Information Engineering G. Marconi, Alma Mater Studiorum Universit di Bologna, Bologna, Italy. He was an Associate Professor from 2016 to 2018, and previously an Assistant Professor, Ricercatore, from 2007, with the Universit del Salento, Lecce, Italy. He has been a Visiting Scholar with the University of Stuttgart, Stuttgart, Germany; University of California Santa Barbara, Santa Barbara, CA, USA; and the University of Colorado Boulder, Boulder, CO, USA. His research interests include distributed optimization, cooperative control in complex networks, applied nonlinear optimal control, and trajectory optimization and maneuvering of aerial and car vehicles.

Dr. Notarstefano serves as an Associate Editor for IEEE TRANSACTIONS ON AUTOMATIC CONTROL, IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, and *IEEE Control Systems Letters*. He has been a part of the Conference Editorial Board of the IEEE Control Systems Society and EUCA. He is a recipient of an ERC Starting Grant 2014.

Francesco Sasso received the Laurea degree (summa cum laude) in mathematics from the Università degli Studi di Bari, Bari, Italy, in 2015, and the Ph.D. degree in engineering of complex systems from the Università del Salento, Salento, Italy, in 2020.

He is working on the ERC-granted project OPT4SMART. His research interests include distributed optimization, estimation theory, and differential equations.



Roland Bouffanais (Member, IEEE) received the B.Sc. and M.Sc. degrees in physics from École Normale Supérieure (ENS Lyon), Lyon, France, in 1997 and 1999, respectively. He also received the M.Sc. degree in physics from UPMC Paris Sorbonne University, Paris, France, in 1999. He received the Ph.D. degree in engineering from École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, in 2007.

He was a Postdoctoral Fellow and an Associate with the Department of Mechanical Engineering, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. He joined the faculty of the Singapore University of Technology and Design (SUTD), Singapore, in 2011, and is now an Associate Professor of Engineering and the Principal Investigator of the Applied Complexity Group. His interdisciplinary research spans a vast breadth of areas and is focused on the design and control of decentralized complex systems, multiagent systems, leader–follower consensus dynamics, and nonlinear dynamical systems.