SIAM J. CONTROL OPTIM.
Vol. 52, No. 1, pp. 629–662

DISTRIBUTED RANDOM CONVEX PROGRAMMING VIA CONSTRAINTS CONSENSUS*

L. CARLONE†, V. SRIVASTAVA‡, F. BULLO§, AND G. C. CALAFIORE¶

Abstract. This paper discusses distributed approaches for the solution of random convex programs (RCPs). RCPs are convex optimization problems with a (usually large) number N of randomly extracted constraints; they arise in several application areas, especially in the context of decision-making under uncertainty; see [G. C. Calafiore, *SIAM J. Optim.*, 20 (2010), pp. 3427–3464; G. C. Calafiore and M. C. Campi, *IEEE Trans. Automat. Control*, 51 (2006), pp. 742–753]. We here consider a setup in which instances of the random constraints (the *scenario*) are not held by a single centralized processing unit, but are instead distributed among different nodes of a network. Each node "sees" only a small subset of the constraints, and may communicate with neighbors. The objective is to make all nodes converge to the same solution as the centralized RCP problem. To this end, we develop two distributed algorithms that are variants of the constraints consensus algorithm [G. Notarstefano and F. Bullo, *Proceedings of the* 46*th IEEE Conference on Decision and Control*, New Orleans, LA, 2007, pp. 927–932; G. Notarstefano and F. Bullo, *IEEE Trans. Automat. Control*, 56 (2011), pp. 2247–2261]: the active constraints consensus algorithm, and the vertex constraints consensus (VCC) algorithm. We show that the active constraints consensus algorithm computes the overall optimal solution in finite time, and with almost surely bounded communication at each iteration of the algorithm. The VCC algorithm is instead tailored for the special case in which the constraint functions are convex also with respect to the uncertain parameters, and it computes the solution in a number of iterations bounded by the diameter of the communication graph. We further devise a variant of the VCC algorithm, namely *quantized vertex constraints consensus* (qVCC), to cope with the case in which communication bandwidth among processors is bounded. We discuss several applications of the proposed distributed techniques, including estimation, classification, and random model predictive control, and we present a numerical analysis of the performance of the proposed methods. As a complementary numerical result, we show that the parallel computation of the scenario solution using the active constraints consensus algorithm significantly outperforms its centralized equivalent.

Key words. distributed optimization, distributed random convex programming, constraints consensus, distributed constraints removal, parallel model predictive control

AMS subject classifications. 68W15, 90C25, 90C05, 68W10, 68W20

DOI. 10.1137/120885796

1. Introduction.

Uncertain optimization problems arise in several engineering applications, for instance, system design and production management, identification and control, manufacturing, and finance; see, e.g., [6]. Uncertainty arises due to the presence of imprecisely known parameters in the problem description. For instance,

*Received by the editors July 24, 2012; accepted for publication (in revised form) November 8, 2013; published electronically February 20, 2014. This work was funded by PRIN grant 20087W5P2K from the Italian Ministry of University and Research, by NSF Award CPS 1035917, and by ARO Award W911NF-11-1-0092. An early presentation of the active constraints consensus algorithm appeared in [1]. Differences between [1] and the current paper include the vertex constraints consensus algorithms, the distributed outliers removal algorithm, proofs and discussions related to the proposed approaches, and applications to distributed estimation and parallel model predictive control.
http://www.siam.org/journals/sicon/52-1/88579.html

†College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 (luca.carlone@gatech.edu).
‡Mechanical and Aerospace Engineering Department, Princeton University, Princeton, NJ 08544 (vaibhavs@princeton.edu).
§Center for Control, Dynamical Systems, and Computation, University of California Santa Barbara, Santa Barbara, CA 93106 (bullo@engineering.ucsb.edu).
¶Dipartimento di Automatica e Informatica, Politecnico di Torino, 10129 Torino, Italy (giuseppe.calafiore@polito.it).

a system design problem may be affected by the uncertainty in the values of some system components, and control problems can be affected by the inexact knowledge of system model and of the disturbances acting on the system. In the case of uncertain convex optimization problems where the uncertainty in the problem description has a stochastic model (e.g., one assumes random uncertain parameters, with some given probability distribution), the random convex program (RCP) paradigm recently emerged as an effective methodology to compute "probabilistically robust" solutions; see, e.g., [7, 8, 9].

An instance of an RCP problem typically results in a standard convex programming problem with a large number N of constraints. There are two main reasons for which it is interesting to explore distributed methods for solving RCP instances: first, the number N of constraints may be too large to store or solve on a single processing unit; second, there exist application endeavors in which the problem description (objective function and constraints) is naturally distributed among different nodes of an interconnected system. This may happen, for instance, when system constraints depend on measurements acquired by different interacting sensors.

In the past few decades, the perspective for solving such large-scale or multinode problems has switched from *centralized* approaches to *distributed* ones. In the former approach, problem data are either resident on a single node or transmitted by each node to a central computation unit that solves the (global) optimization problem. In distributed approaches, the computation is distributed among nodes that must reach a consensus on the overall problem solution through local computation and internodal communication. The advantages of the distributed setup are essentially threefold: (i) distributing the computation burden and the memory allocation among several processors; (ii) reducing communication by avoiding gathering all available data to a central node; (iii) increasing the robustness of the systems with respect to failures of the central computational unit.

Following this distributed optimization philosophy, we here consider a network of agents or processors that has to solve an RCP in a distributed fashion. Each node in the network knows a subset of the constraints of the overall RCP, and the nodes communicate with each other with the purpose of determining the solution of the overall problem. Our solution methodology relies on each node iteratively exchanging a small set of relevant constraints and determining the solution to the RCP in finite time. This methodology is in fact a variation of the constraints consensus algorithm proposed in [4] and further developed in [5].

Related work. Distributed and parallel optimization have received significant attention in the literature. In earlier works [10, 11], Lagrangian-based decomposition techniques are used to develop decentralized algorithms for large-scale optimization problems with separable cost functions. In the seminal work [12], Tsitsiklis investigates the parallel computation of the minimum of a smooth convex function under a setup in which each processor has partial knowledge of the global cost function, and they exchange information about the gradients of their local cost functions to compute the global solution. Recently, Nedíc and Ozdaglar [13] generalize the setup of [12] to distributed computation and provide results on the convergence rate and error bounds for unconstrained problems in synchronous networks. In a similar spirit, Zhu and Martínez [14] study the primal-dual subgradient algorithm for the distributed computation of the optimal solution of a constrained convex optimization problem with inequality and equality constraints. Wei, Ozdaglar, and Jadbabaie [15] study a distributed Newton method under a setup in which each node has partial knowledge of the cost function, and the optimization problem has linear global constraints. Boyd et al. [16] propose a technique based on dual-decomposition that alternates the updates on different components of the optimization variable. In all these approaches, the proposed algorithms converge to the global solution asymptotically.

An alternative approach to distributed optimization [4, 5, 17, 18] is based on the following idea: nodes exchange a small set of constraints at each iteration and converge in finite time to a consensus set of constraints that determines the global solution of the optimization problem. In particular, Notarstefano and Bullo [4, 5] propose the constraints consensus algorithm for abstract optimization, while Bürger et al. [17, 18] present a distributed simplex method for solving linear programs. The algorithms studied in this paper belong to the latter class of algorithms that converge in finite time. Particularly, our first algorithm, the active constraints consensus, is an adaptation to the RCP context of the constraints consensus algorithm in [5]. Both these algorithms work under similar setups, have a similar approach, and have very similar properties. The main difference between the two algorithms is in the computation of the set of constraints to be transmitted at each iteration. This computation for the algorithm in [5] may need to solve a number of convex programs that grows linearly in the number of constraints and subexponentially in the dimension of the problem, while the algorithm considered here always requires the solution of only one convex program. This lower local computation comes at the expense of potentially larger communication at each iteration. In particular, the number of constraints exchanged at each iteration may be higher for the active constraints consensus algorithm than the constraints consensus algorithm.

Paper structure and contributions. In section 2 we recall some preliminary concepts on the constraints of convex programs (support constraints, active constraints, etc.). In section 3 we introduce the main distributed random convex programming model, and we describe the setup in which the problem has to be solved. The active constraints consensus algorithm is presented and analyzed in section 4. In the active constraints consensus algorithm, each node at each iteration solves a local optimization problem and transmits to its neighbors the constraints that are tight at the solution (i.e., that are satisfied with equality). We show that the active constraints consensus algorithm converges to the global solution in finite time and that it requires almost surely bounded communication at each iteration. We give some numerical evidence of the fact that the active constraints consensus algorithm converges in a number of iterations that is linear in the communication graph diameter. We also provide numerical evidence that parallel implementation of the active constraints consensus algorithm significantly reduces the computation time over the centralized computation time. As a side result, we show that the active constraints consensus algorithm may distributively compute the solution of any convex program and that it is particularly effective when the dimension of the decision variable is small compared with the number of constraints.

For the special case when the constraints of the RCP are convex in the uncertain parameters, we develop the *vertex constraints consensus* (VCC) algorithm in section 5. In the VCC algorithm, each node at each iteration constructs the convex hull of the uncertain parameters, which define local constraints, and transmits its extreme points to the neighbors. We prove that the VCC algorithm converges to the global solution in a number of iterations equal to the diameter of the communication graph. Moreover, we devise a *quantized vertex constraints consensus* (qVCC) algorithm in which each node has a bounded communication bandwidth and does not necessarily transmit all the extreme points of the convex hull at each iteration. We provide theoretical bounds on the number of the iterations required for the qVCC algorithm to converge.

Further, we show in section 6 that each of the proposed algorithms can be easily modified to enable a *distributed constraints removal strategy* that discards outlying constraints, in the spirit of the RCPV (RCP with violated constraints) framework described in [2]. In section 7 we present several numerical examples and applications of the proposed algorithms to distributed estimation, distributed classification, and parallel model predictive control. Conclusions are drawn in section 8.

2. Preliminaries on convex programs. Consider a generic d-dimensional convex program

$$P[C]: \quad \min_{x \in X} \quad a^\top x \qquad \text{subject to} \qquad f_j(x) \le 0 \qquad \forall j \in \{1, \ldots, N\}, \tag{2.1}$$

where $x \in X$ is the *optimization variable*, $X \subset \mathbb{R}^d$ is a compact and convex domain, $a \in \mathbb{R}^d$ is the *objective direction*, and $f_j : \mathbb{R}^d \to \mathbb{R}$, $j \in \{1, \ldots, N\}$, are convex functions defining the problem *constraints*. We use the notation $C \doteq \{c_1, \ldots, c_N\}$ to denote the set of constraints for problem $P[C]$, where each element $c_j \in C$ denotes the corresponding constraint $f_j(x) \le 0$. The presence of the compact domain X is a technical requirement that guarantees boundedness of the objective and attainment of the solution in the optimization problem. However, this requirement is not restrictive in practice, since realistic design problems typically involve a priori bounds on the admissible range of the variables, and these ranges naturally define a domain X. Moreover, the set X may also encode deterministic constraints (e.g., the nonnegativity constraints of (7.2) in section 7). We denote the solution of problem $P[C]$ by $x^*(C)$ and the corresponding optimal value by $J^*(C)$; we assume by convention that $x^*(C) =$ `NaN` and $J^*(C) = \infty$ whenever the problem is infeasible. We now introduce some definitions in accordance with [2].

DEFINITION 2.1 (support constraint set). *The support constraint set,* $\texttt{Sc}(C) \subseteq C$, *of problem* $P[C]$ *is the set of* $c \in C$ *such that* $J^*(C \backslash \{c\}) < J^*(C)$.

The cardinality of the set of support constraints is upper bounded by $d + 1$, and this upper bound reduces to d if the problem is feasible; see Lemmas 2.2 and 2.3 in [2].

DEFINITION 2.2 (invariant and irreducible constraint set). *A constraint set* $S \subseteq C$ *is said to be* invariant *for problem* $P[C]$ *if* $J^*(S) = J^*(C)$. *A constraint set* $S \subseteq C$ *is said to be* irreducible *if* $S \equiv \texttt{Sc}(S)$.

DEFINITION 2.3 (nondegenerate problems). *Problem* $P[C]$ *is said to be* nondegenerate *when* $\texttt{Sc}(C)$ *is invariant.*

DEFINITION 2.4 (essential constraint sets). *An invariant constraint set* $S \subseteq C$ *of minimal cardinality is said to be an* essential *set for problem* $P[C]$. *The collection of all essential sets of problem* $P[C]$ *is denoted as* $\texttt{Es}(C)$.

DEFINITION 2.5 (constraints in general position). *Constraints* $f_j(x) \le 0$, $j \in \{1, \ldots, N\}$, *are said to be in* general position *if the index set* $\{i \in \{1, \ldots, N\} : f_i(x) = 0\}$ *has cardinality no larger than d for all* $x \in X$. *In other words, the constraints are in general position if no more than d of the N surfaces* $f_j(x) = 0$, $j \in \{1, \ldots, N\}$, *intersect at any point of the domain* X.

Feasible convex programs may have more than one solution; i.e., several values of the optimization variable may attain the same optimal objective value. The convex program $P[C]$ satisfies the *unique minimum condition* if problem $P[C_i]$ admits a unique solution for any $C_i \subseteq C$. A convex program that does not satisfy the unique minimum condition can be modified into an equivalent problem that satisfies the unique minimum condition by applying a suitable *tie-breaking rule*[1] (e.g., choosing the lexicographic smallest solution within the set of optimal solutions); see [2]. Accordingly and without loss of generality, in the following we consider convex programs satisfying the unique minimum condition. We conclude this section with the following definition.

DEFINITION 2.6 (active constraint set). *The active constraint set* $\texttt{Ac}(C) \subseteq C$ *of a feasible problem* $P[C]$ *is the set of constraints that are tight at the optimal solution* $x^*(C)$, *that is,* $\texttt{Ac}(C) = \{c_j, j \in \{1, \dots, N\} : f_j(x^*(C)) = 0\}$. *By convention, the active constraint set of an infeasible problem is the empty set.*

2.1. Properties of the constraint sets. We now study some properties of the constraint sets in a convex program. We first state the properties of monotonicity and locality in convex programs.

PROPOSITION 2.7 (monotonicity and locality [19, 2]). *For the convex optimization problem* $P[C]$, *constraint sets* $C_1, C_2 \subseteq C$, *and a generic constraint* $c \in C$, *the following properties hold:*

(i) Monotonicity*:* $J^*(C_1) \leq J^*(C_1 \cup C_2)$.

(ii) Locality*: if* $J^*(C_1) = J^*(C_1 \cup C_2)$, *then*

$$J^*(C_1 \cup \{c\}) > J^*(C_1) \iff J^*(C_1 \cup C_2 \cup \{c\}) > J^*(C_1 \cup C_2). \tag{2.2}$$

Let the number of different essential sets in C be n_e, and let $\texttt{Es}_i(C)$ be the ith essential set. The following proposition discusses the relationships between support, essential, and active constraint sets.

PROPOSITION 2.8 (properties of the constraint sets). *The following statements hold for the constraint sets of a feasible problem* $P[C]$*:*

(i) *The set of active constraints contains the set of support constraints, that is,* $\texttt{Ac}(C) \supseteq \texttt{Sc}(C)$.

(ii) *The set of active constraints contains the union of all the essential sets, that is,* $\texttt{Ac}(C) \supseteq \cup_{i=1}^{n_e} \texttt{Es}_i(C)$.

Proof. See Appendix A.1.

We now state an immediate consequence of Proposition 2.8.

COROLLARY 2.9 (invariance of active constraint set). *The active constraint set of problem* $P[C]$ *is an invariant constraint set for* $P[C]$.

3. Distributed random convex programming. In this section, we first recall some basic concepts of (standard) random convex programming [2], and then we define our setup for distributed random convex programming in section 3.2.

3.1. Definition and properties of random convex programs. An RCP is a convex optimization problem of the form

$$\begin{aligned} P[C]: \quad \min_{x \in X} \quad & a^\top x \qquad \text{subject to} \\ & f(x, \delta^{(j)}) \leq 0, \quad j \in \{1, \dots, N\}, \end{aligned} \tag{3.1}$$

[1]Practical examples of tie-breaking rules are reported in Appendix I of [1] and Appendix A of [3]. An example that is closer to the discussion of this paper is provided in [5, Example II.1]. In particular, the authors show how to recast a generic linear problem into one satisfying the uniqueness condition and discuss how this ensures monotonicity and locality, which are required for constraints consensus.

where $\delta^{(j)}$ are N independent and identically distributed (iid) samples of a random parameter $\delta \in \Delta \subseteq \mathbb{R}^\ell$ having probability distribution $\mathbb{P}$, and $f(x,\delta) : \mathbb{R}^d \times \Delta \to \mathbb{R}$ is convex in x, for any $\delta \in \Delta$ (the dependence of f on δ can instead be generic). The multisample $\omega \doteq \{\delta^{(1)}, \delta^{(2)}, \ldots, \delta^{(N)}\}$ is called a *scenario*, and the solution of problem (3.1) is called a *scenario solution*. Notice that, a priori (i.e., before any sample is actually extracted), ω is a random variable with probability distribution $\mathbb{P}^N$ (the N-fold product of the marginal distribution $\mathbb{P}$). For a given *realization* $\tilde{\omega} \doteq \{\tilde{\delta}^{(1)}, \tilde{\delta}^{(2)}, \ldots, \tilde{\delta}^{(N)}\}$ of ω, an instance of the RCP (3.1) has precisely the format of the convex program in (2.1), for $f_j(x) \doteq f(x, \tilde{\delta}^{(j)})$, and for this reason, with slight abuse of notation, we kept the name $P[C]$ for (3.1). Clearly, the set of constraints C depends on the multisample ω, i.e., $C = C(\omega)$.

A key feature of an RCP is that we can bound a priori the probability that the scenario solution remains optimal for further realization of the uncertainty [2]. We introduce the following definition.

Definition 3.1 (violation probability). *The* violation probability $\texttt{Viol}^*(\omega)$ *of the RCP* (3.1) *is defined as*

$$\texttt{Viol}^*(\omega) \doteq \mathbb{P}\{\delta \in \Delta : \, J^*\left(C\left(\omega \cup \{\delta\}\right)\right) > J^*\left(C\left(\omega\right)\right)\},$$

where $J^*\left(C\left(\omega\right)\right)$ denotes the optimal value of (3.1)*, and $J^*\left(C\left(\omega \cup \{\delta\}\right)\right)$ denotes the optimal value of a modification of problem* (3.1)*, where a further random constraint $f(x,\delta) \le 0$ is added to the problem.*

If problem (3.1) is nondegenerate with probability one, then the violation probability of the solution satisfies

$$\mathbb{P}\{\omega \in \Delta^N : \texttt{Viol}^*(\omega) \le \epsilon\} \ge 1 - \Phi(\epsilon; \zeta - 1, N), \tag{3.2}$$

where $\Phi(\epsilon; q, N) \doteq \sum_{j=0}^{q} \binom{N}{j} \epsilon^j (1-\epsilon)^{N-j}$ is the cumulative distribution function of a binomial random variable, and ζ is equal to d, if the problem is feasible with probability one, and is equal to $d+1$, otherwise; see Theorem 3.3 of [2]. Furthermore, if one knows a priori that problem (3.1) is feasible with probability one, then the violation probability $\texttt{Viol}^*(\omega)$ also represents the probability with which the optimal solution $x^*(\omega)$ of (3.1) violates a further random constraint, that is,

$$\texttt{Viol}^*(\omega) = \mathbb{P}\{\delta \in \Delta : \, f(x^*(\omega), \delta) > 0\};$$

see section 3.3 in [2].

For a given $\beta \in (0,1)$, the bound in (3.2) is implied by

$$\mathbb{P}\{\omega \in \Delta^N \; : \texttt{Viol}^*(\omega) \le 2(\log \beta^{-1} + \zeta - 1)/N\} \ge 1 - \beta. \tag{3.3}$$

In practice, one chooses a *confidence level* $1-\beta$ close to 1 and picks N large enough to achieve a desired bound on the probability of violation. These bounds on the violation probability depend neither on the uncertainty set Δ nor on the probability distribution of δ over Δ. Hence, the RCP framework relaxes the basic assumptions underlying robust and chance-constrained optimization [2].

Remark 1 (distributed scenario optimization). Equations (3.2)–(3.3) provide a priori guarantees on the robustness of the scenario solution; i.e., they bound the violation probability before the realization $\tilde{\omega}$ of the random multisample ω is actually extracted. Once the realization $\tilde{\omega}$ is extracted, the RCP (3.2) becomes a standard convex program. In this paper we deal with the distributed solution of such a problem instance, i.e., a given realization of an RCP. For simplicity of notation, in the following sections we use $\delta^{(j)}$ (instead of $\tilde{\delta}^{(j)}$) to denote a realization of the jth sample, and we remark that, unless specified otherwise, these are no longer intended as random variables but are given vectors, since the distributed algorithms are applied a posteriori, i.e., after the extraction of the multisample.

3.2. A distributed setup for random convex programs. We next describe a distributed formulation of an RCP problem instance. The proposed formulation is similar to the distributed abstract optimization setup in [4, 5]. Consider a system composed of n interacting nodes (e.g., *processors*, *sensors*, or, more generically, *agents*). We model internodal communication by a directed graph $\mathcal{G}$ with vertex set $\{1, \dots, n\}$: a directed edge (i, j) exists in the graph if node i can transmit information to node j. We assume that the directed graph $\mathcal{G}$ is strongly connected, that is, it contains a directed path from each vertex to any other vertex. Let $\mathcal{N}_{\text{in}}(i)$ and $\mathcal{N}_{\text{out}}(i)$ be the set of incoming and outgoing neighbors of agent i, respectively. Let the *diameter* of the graph $\mathcal{G}$ be $\mathtt{diam}(\mathcal{G})$. We state the distributed random programming problem as follows.

PROBLEM 1 (distributed random convex programming). *A networked system with a strongly connected communication graph has to compute the scenario solution for an instance of the random convex program* (3.1)*, under the following setup:*

(i) *Each node knows the objective direction a and the domain X;*

(ii) *each node initially knows only a subset $C_i \subseteq C$ of the constraints of problem* (3.1) (*the local constraint set*)*, $\cup_{i=1}^{n} C_i = C$;*

(iii) *a generic node i can receive information from the incoming neighbors $\mathcal{N}_{\text{in}}(i)$ and can transmit information to the outgoing neighbors $\mathcal{N}_{\text{out}}(i)$.*

Let $N_i \doteq |C_i|$, for each $i \in \{1, \dots, n\}$, and let $N = |C|$. Since each node has only partial knowledge of problem constraints, it needs to cooperate with the other nodes to compute the solution of $P[C]$. Note that we assume the domain X to be known by all nodes. This is a mild assumption in practice, since, as mentioned in section 2, X describes problem-specific constraints (e.g., nonnegativity of some variables). An iteration at a node is initiated when the node receives local information from each of its neighbors. In the following, we assume that, at any iteration $t \in \mathbb{Z}_{\geq 0}$, node i in the network is able to solve local convex optimization problems of the form

$$
\text{(3.4)} \qquad \begin{aligned} P[L_i(t)] : \min_{x \in X} \quad & a^\top x \qquad \text{subject to} \\ & f_j(x) \leq 0 \qquad \forall\, c_j \in L_i(t), \end{aligned}
$$

where $L_i : \mathbb{Z}_{\geq 0} \to \mathtt{pow}(C)$ is the subset of constraints that is locally known at node i at time t (possibly with $|L_i| \ll |C|$), and $\mathtt{pow}(C)$ represents the set of all subsets of $C \doteq \{c_1, \dots, c_N\}$. We refer to the solution of problem (3.4) as the *local solution* $x_i^*(t) \doteq x^*(L_i(t))$ and to the associated value of the objective function as the *local optimal value* $J_i^*(t) \doteq J^*(L_i(t))$ (under the convention that $x_i^*(t) = \mathtt{NaN}$ and $J_i^*(t) = \infty$ for infeasible problems).

4. Active constraints consensus algorithm. In this section we describe the active constraints consensus distributed algorithm to solve the RCP (3.1). We assume that a generic node i at time t can store a small *candidate* constraint set $A_i(t)$, the local optimal solution $x_i^*(t)$, and the local optimal objective $J_i^*(t)$. In the active constraints consensus algorithm, each node initially solves the local convex program $P[C_i]$, finds the active constraints $\mathtt{Ac}(C_i)$, and initializes $A_i(0) = \mathtt{Ac}(C_i)$, $x_i^*(0) = x^*(C_i)$, and $J_i^*(0) = J^*(C_i)$. At each iteration t of the algorithm, node i receives the objective values $J_j^*(t)$ and the candidate sets $A_j(t)$ from the incoming neighbors, $j \in \mathcal{N}_{\text{in}}(i)$, and builds the constraint set in the following way:

$$
L_i(t+1) = A_i(t) \cup \big(\cup_{j \in \mathcal{N}_{\text{in}}(i)} A_j(t) \big) \cup C_i.
$$

Algorithm 1: Active Constraints Consensus

Input : a, X, C_i, and $\texttt{dm} = \texttt{diam}(\mathcal{G})$;
Output : $x^*(C)$, $J^*(C)$, and $\texttt{Ac}(C)$;

% Initialization:

$A_i(0) = \texttt{Ac}(C_i)$, $J_i^*(0) = J^*(C_i)$, $x_i^*(0) = x^*(C_i)$, and $\texttt{ncc} = 1$;
$t = 0$;

% ACC iterations:

while $\texttt{ncc} < 2\texttt{dm} + 1$ **and** $J_i^*(t) < \infty$ **do**

  % Poll neighbors and build:
  $L_i(t+1) = A_i(t) \cup \big(\cup_{j \in \mathcal{N}_{\text{in}}(i)} A_j(t)\big) \cup C_i$;
  $\tilde{J}_i^*(t+1) = \max_{j \in \mathcal{N}_{\text{in}}(i)} J_j^*(t)$;
  % Check infeasibility:
  if $\tilde{J}_i^*(t+1) = \infty$ **then**
    $A_i(t+1) = \emptyset$, $J_i^*(t+1) = \infty$, $x_i^*(t+1) = \texttt{NaN}$;
    exit;
  % Update candidate set:
  $A_i(t+1) = \texttt{Ac}\,(L_i(t+1))$, $J_i^*(t+1) = J^*\,(L_i(t+1))$, $x_i^*(t+1) = x^*\,(L_i(t+1))$;
  % Update ncc for stopping condition:
  if $J_i^*(t+1) = J_i^*(t)$ **then**
    $\texttt{ncc} = \texttt{ncc} + 1$;
  else
    $\texttt{ncc} = 1$;
  $t = t + 1$;

return $x_i^*(t)$, $J_i^*(t)$, $A_i(t)$;

Each node then solves problem $P[L_i(t+1)]$ and updates the local quantities, setting $A_i(t+1) = \texttt{Ac}(L_i(t+1))$, $x_i^*(t+1) = x^*(L_i(t+1))$, and $J_i^*(t+1) = J^*(L_i(t+1))$. We note that the solution to $P[L_i(t+1)]$ requires the objective direction a and domain X to be known at node i. The algorithm is iterated until a stopping condition is met (see Remark 2). The details of the algorithm to be executed by each node i, $i \in \{1, \dots, n\}$, are reported as a pseudocode in Algorithm 1. The key properties of the active constraints consensus algorithm are summarized in the following proposition.

PROPOSITION 4.1 (properties of the active constraints consensus algorithm). *For a distributed RCP (Problem* 1*) and the active constraints consensus algorithm (Algorithm* 1*), the following statements hold:*

(i) *The local optimal objective $J_i^*(t)$ is monotonically nondecreasing with the iterations t;*
(ii) *the local optimal objective and the local solution converge in a finite number T of iterations to the optimal value $J^*(C)$ and the scenario solution $x^*(C)$ of the RCP;*
(iii) *for each node i, the local candidate set $A_i(T)$ coincides with the active set $\texttt{Ac}(C)$ of the RCP;*
(iv) *if constraints in C are in general position, at each iteration of Algorithm* 1*, each node transmits to each of the outgoing neighbors its current objective value $J_i^*(t)$ and at most d constraints.*

Proof. The proof of the proposition is an adaptation of the proof of Theorem IV.4 in [5]. We report the proof in Appendix A.2. The main difference in the proofs is that we tailor the demonstration to the exchange of active constraints (instead of the constraints in the *basis*) and we consider explicitly the case of infeasible programs.

Remark 2 (stopping rule for the active constraints consensus algorithm). An important fact for the demonstration of claim (i) of Proposition 4.1 is that if the local optimal objective $J_i^*(t)$ at one node does not change for $2\mathtt{diam}(\mathcal{G})+1$ iterations, then convergence has been reached. This fact can be used for implementing a local stopping condition: node i stores an integer (`ncc` in Algorithm 1) that counts the number of iterations in which the local optimal objective has not changed (`ncc` is an acronym for "no change counter"). Then the node can stop the algorithm as soon as this counter reaches the value $2\mathtt{diam}(\mathcal{G})+1$. The node can also stop iterating the algorithm when an infeasible instance is discovered in its local problem or within the local problems of its neighbors. In particular, as soon as node i discovers infeasibility, it sets its objective to $J_i^* = \infty$ and propagates it to the neighbors; as a consequence, all nodes detect the infeasibility in at most $\mathtt{diam}(\mathcal{G})$ iterations. For simplicity we let the nodes exchange the local objectives at every iteration; however, it is worth noticing that the objectives from the neighbors serve only as infeasibility flags, i.e., they influence the algorithm only when $J_j^* = \infty$ for some $j \in \mathcal{N}_{\mathrm{in}}(i)$.

Remark 3 (comparison with constraints consensus algorithm [5]). The constraints consensus algorithm [5] also distributively computes the solution of a convex program and is, in fact, identical to the active constraints consensus algorithm whenever the active constraint set and the essential constraint set (basis) are identical. However, in general, the constraints consensus algorithm requires the nodes to compute a basis of the local set of constraints at each iteration, and such a computation may be expensive. In particular, for the computation of a basis of a degenerate d-dimensional problem with N_i constraints, the algorithm proposed in [5] requires the solution of a number of convex optimization problems that depends linearly on N_i and subexponentially on d. On the other hand, the active set computation in the active constraints consensus algorithm requires the solution of at most one convex program. Particularly, if the local solution $x_i^*(t)$ satisfies all incoming neighbors constraints, then no optimization problem is solved, and the update rule of the active constraints consensus algorithm requires only to check whether some of the incoming constraints are active. This lower computational expense is achieved at the price of a potentially higher communication cost. In particular, the active constraints consensus algorithm transmits the set of active constraints at each iteration, and the active constraint set is a superset of each basis.

Remark 4 (distributed convex programming and constraints exchange). The active constraints consensus algorithm can be used for the distributed computation of the solution of any convex program. The distributed strategy is particularly advantageous when the dimension of the decision variable is small and the number of constraints is large (as in the RCP setup), since in this case the nodes exchange only a small subset of constraints of the local constraint sets. Moreover, each constraint $f_j(x) \doteq f(x, \delta^{(j)})$ of an RCP is parameterized in the realization $\delta^{(j)}$; therefore "exchanging" the constraint $f_j(x)$ reduces to transmitting the vector $\delta^{(j)} \in \mathbb{R}^\ell$.

5. Vertex constraints consensus algorithms.

In this section, we propose distributed algorithms for RCPs, specialized to the case of constraints that are convex with respect to the parameter δ.

Assumption 1 (convex uncertainty). For any given $x \in X$, the function $f(x,\delta)$ in (3.1) is convex in $\delta \in \Delta$.

Consider an instance of the RCP in (3.1). Let the *feasible set* of problem $P[C]$ be $\mathtt{Sat}(C) \doteq \{x \in X \ : \ f(x,\delta^{(j)}) \le 0 \text{ for all } j \in \{1,\dots,N\}\}$ (set of variables

x that *satisfy* all constraints in C [2]). Let $\texttt{co}(C)$ denote the convex hull of vectors $\delta^{(j)} \in \Delta$, $j \in \{1, \dots, N\}$, and let $\texttt{vert}(C) \subseteq C$ denote the constraints corresponding to the vectors that form the vertices of $\texttt{co}(C)$. We now state the following lemma.

LEMMA 5.1 (invariance of the vertex set). *If problem* (3.1) *satisfies Assumption* 1, *then* $\texttt{vert}(C) \subseteq C$ *is an invariant constraint set.*

Proof. In order to demonstrate that $\texttt{vert}(C)$ is invariant we show that every $\bar{x}$ that satisfies the constraints in $\texttt{vert}(C)$ also satisfies the constraints in $C \backslash \texttt{vert}(C)$. Assume that $\bar{x}$ satisfies the constraints $c_j \in \texttt{vert}(C)$, where a generic constraint c_j is in the form $f(\bar{x}, \delta^{(j)}) \leq 0$. Now consider a constraint $\bar{c}$ in the set $C \backslash \texttt{vert}(C)$, corresponding to the vector $\bar{\delta}$. Since $\bar{\delta}$ is in the convex hull having vertices $\{\delta^{(j)} : c_j \in \texttt{vert}(C)\}$, it can be written as a convex combination of the vertices as follows: $\bar{\delta} = \sum_{c_j \in \texttt{vert}(C)} \tau_j \delta^{(j)}$, with $\sum_{c_j \in \texttt{vert}(C)} \tau_j = 1$ and $\tau_j \geq 0$. Therefore, for Jensen's inequality, it holds that $f(\bar{x}, \bar{\delta}) = f(\bar{x}, \sum_{c_j \in \texttt{vert}(C)} \tau_j \delta^{(j)}) \leq \sum_{c_j \in \texttt{vert}(C)} \tau_j f(\bar{x}, \delta^{(j)}) \leq 0$, which proves that also the constraint $\bar{c} \in C \backslash \texttt{vert}(C)$ is satisfied at $\bar{x}$. ∎

As a consequence of the above lemma, the problem $P[\texttt{vert}(C)]$ is equivalent to the problem $P[C]$ in the sense that they admit the same solution. We now present the VCC algorithm.

5.1. The vertex constraints consensus algorithm. The VCC algorithm assumes that at time t a generic node i in the network can store a *candidate set* $V_i(t)$ that is initialized to $V_i(0) = \texttt{vert}(C_i)$ (i.e., it computes the convex hull of the vectors $\delta^{(j)}$, $c_j \in C_i$, and stores the constraints associated with the vertices of the convex hull). At each iteration t of the VCC algorithm, node i receives the candidate sets $V_j(t)$ from the incoming neighbors, $j \in \mathcal{N}_{\text{in}}(i)$, and builds the constraint set $L_i(t+1) = V_i(t) \cup \big(\cup_{j \in \mathcal{N}_{\text{in}}(i)} V_j(t)\big)$. Then, the node updates its candidate set with the following rule: $V_i(t+1) = \texttt{vert}\big(L_i(t+1)\big)$. The algorithm is iterated for $\texttt{diam}(\mathcal{G})$ iterations, as summarized in Algorithm 2. After the main loop, each node can compute the optimal solution $x_i^*(t)$ and the optimal objective $J_i^*(t)$ by using the set $V_i(t)$ and exploiting the knowledge of the objective direction a and the domain X.

Algorithm 2: Vertex Constraints Consensus (VCC)

Input : a, X, C_i, and $\texttt{dm} = \texttt{diam}(\mathcal{G})$;
Output : $x^*(C)$, $J^*(C)$, and $\texttt{vert}(C)$;

% Initialization:
$V_i(0) = \texttt{vert}(C_i)$;
$t = 0$;

% VCC iterations:
while $t < \texttt{dm}$ **do**
  % Poll neighbors and build:
  $L_i(t+1) = V_i(t) \cup \big(\cup_{j \in \mathcal{N}_{\text{in}}(i)} V_j(t)\big)$;
  % Update candidate set:
  $V_i(t+1) = \texttt{vert}\,(L_i(t+1))$;
  $t = t + 1$;
% Compute optimal solution and optimal objective:
$x_i^*(t) = x^*(V_i(t))$, $J_i^*(t) = J^*(V_i(t))$;
return $x_i^*(t)$, $J_i^*(t)$, $V_i(t)$;

PROPOSITION 5.2 (properties of the VCC algorithm). *For a distributed random convex program (Problem 1) that satisfies Assumption 1, and the VCC algorithm (Algorithm 2), the following statements hold:*

(i) *The local optimal objective $J_i^*(t) \doteq J^*(V_i(t))$ is monotonically nondecreasing with the iterations t;*

(ii) *in $T \leq \mathtt{diam}(\mathcal{G})$ iterations the local solution at a generic node i coincides with the scenario solution of the RCP;*

(iii) *for each node i the local candidate set $V_i(T)$ satisfies $V_i(T) = \mathtt{vert}(C) \supseteq \mathtt{Sc}(C)$.*

Proof. See Appendix A.3.

Remark 5 (computational complexity of convex hull). At each iteration of the VCC algorithm each node computes and transmits the convex hull of a set of vectors in $\mathbb{R}^\ell$. There is an extensive literature on the complexity of convex hull computation and on the expected number of vertices in the convex hull; see, e.g., [20, 21, 22]. In particular, it is known that the convex hull of N points in $\mathbb{R}^\ell$ can be computed in $\mathcal{O}(N \log N + N^{\lceil \ell/2 \rceil})$ iterations. Moreover, for N points uniformly sampled from the interior of an ℓ-dimensional polytope, there exist algorithms (see, e.g., [22]) that construct the convex hull in $\mathcal{O}(N)$ average time, and the resulting hull has an $\mathcal{O}((\log N)^{\ell-1})$ expected number of vertices.

Remark 6 (distributed uncertain linear programs). A remarkable context in which the VCC algorithm can be applied is that of uncertain linear programs.

Consider an RCP instance of a standard-form uncertain linear program

$$\begin{aligned} &\min_{x \in X} \quad a^\top x \qquad \text{subject to} \\ &\qquad u_i^\top\big(z^{(j)}\big)\, x + v_i\big(z^{(j)}\big) \leq 0 \text{ for each } i \in \{1, \dots, r\}, \text{ and } j \in \{1, \dots, N\}, \end{aligned} \tag{5.1}$$

where X is a polytope and $z^{(j)}$ are iid realizations of some random uncertain parameter $z \in \mathcal{Z}$, where $\mathcal{Z}$ is some arbitrary space entering the data $u_i(z) \in \mathbb{R}^d$, $v_i(z) \in \mathbb{R}$ in an *arbitrary* way. For given realizations of z, problem (5.1) is linear in x; however, since $u_i(z)$, $v_i(z)$ may be generic nonconvex functions, the constraints are nonconvex in z and the problem does not satisfy Assumption 1 in general. However, we can easily transform the problem into one satisfying Assumption 1: we define the random parameters $\delta_i = \delta_i(z) \doteq [u_i^\top(z) \;\; v_i(z)] \in \mathbb{R}^{1 \times (d+1)}$ and we reparameterize (5.1) as

$$\begin{aligned} &\min_{x \in X} \quad a^\top x \qquad \text{subject to} \\ &\qquad \delta_i^{(j)} [x^\top \; 1]^\top \leq 0 \text{ for each } i \in \{1, \dots, r\}, \text{ and } j \in \{1, \dots, N\}, \end{aligned} \tag{5.2}$$

where $\delta_i^{(1)}, \dots, \delta_i^{(N)}$ are realizations of δ_i. Clearly, each constraint $\delta_i^{(j)}[x^\top \; 1]^\top \leq 0$ is now a linear function of δ_i, hence Assumption 1 is satisfied, and the VCC algorithm can be applied to problem (5.2), operating on the vertices of the convex hull of the $\delta_i^{(j)}$ parameters. Also, problem (5.2) can be formally cast in the standard RCP format (3.1) by setting $f(x, \delta) = \max_{i \in \{1, \dots, r\}} \delta_i [x^\top \; 1]$, where δ contains the collection of the δ_i, $i \in \{1, \dots, r\}$.

Remark 7 (constraints reexamination). The active constraints consensus algorithm requires each node i to reexamine its local constraint set C_i at each iteration. This reexamination is attributed to the fact that a constraint that is not active at a given iteration may become active at a later iteration (see [5] for a similar argument for the constraints consensus algorithm). The VCC algorithm, instead, requires knowledge of C_i only for the initialization, and utilizes only the current candidate set and newly received constraints to determine the new candidate set. At a generic iteration t of the VCC algorithm at node i, any constraint that lies in the interior of the computed convex hull $\mathtt{co}(L_i(t))$ will never belong to any candidate set at future iterations (a point in the interior of the convex hull will lie in the interior of any future convex hull; therefore, it cannot be one of its vertices), and therefore, it can be discarded.

We conclude this section by noticing that the update rule of the VCC algorithm is independent of the objective direction a. Therefore, each node does not need to know the objective direction to reach consensus on the set of constraints defining the feasible set of problem $P[C]$. However, agents use the objective direction a in the final computation of $x_i^*(t)$ and $J_i^*(t)$.

5.2. Quantized VCC algorithm. The size of the constraint set to be transmitted at each iteration of the VCC algorithm may grow exponentially with the dimension of the parameter vector. Such communication at each iteration of the algorithm may not be sustainable for nodes with a limited communication bandwidth. In this section, we address this issue and modify the VCC algorithm to develop the *quantized VCC* (qVCC) algorithm. The qVCC algorithm differs from the VCC algorithm on the following fronts: (i) each node can transmit at most a fixed number m of constraints in a single communication round (*bounded communication bandwidth*), and (ii) a generic node i at time t stores an ordered set, called the transmission set, $T_i(t)$, along with the *candidate set*, $V_i(t)$. The algorithm works as follows. Each node initializes $V_i(0) = T_i(0) = \mathtt{vert}(C_i)$; i.e., both sets contain the constraints corresponding to the vertices of the convex hull $\mathtt{co}(C_i)$. At each iteration t of the qVCC algorithm, each node selects the first m constraints in $T_i(t)$, defining the *current message* $M_i(t)$, and transmits $M_i(t)$ to the outgoing neighbors. When a node receives the messages $M_j(t)$ from the incoming neighbors, $j \in \mathcal{N}_{\text{in}}(i)$, it builds the constraint set $L_i(t+1) = V_i(t) \cup \big(\cup_{j\in\mathcal{N}_{\text{in}}(i)} M_j(t)\big)$. Then, node i updates its candidate set with the following rule: $V_i(t+1) = \mathtt{vert}\big(L_i(t+1)\big)$. Moreover, it updates the transmission set with the rule $T_i(t+1) = T_i(t)\backslash\{M_i(t) \cup \big(V_i(t)\backslash V_i(t+1)\big)\} \oplus \{V_i(t+1)\backslash V_i(t)\}$, where $\oplus$ denotes the concatenation of two ordered sets. Roughly speaking, the updated transmission set, $T_i(t+1)$, is obtained from the previous one, $T_i(t)$, by removing (i) the constraints transmitted at time t, i.e., $M_i(t)$; (ii) the constraints that disappeared from the candidate set after the update, i.e., $V_i(t)\backslash V_i(t+1)$; and (iii) by adding the constraints that became part of the candidate set after the update, $V_i(t+1)\backslash V_i(t)$. Note that the set $T_i(t)$ has to be ordered to implement a first-in-first-out (FIFO) strategy for transmitting constraints to the neighbors. The algorithm is iterated until a stopping condition is met. The qVCC algorithm for node i is summarized in Algorithm 3. For simplicity, a centralized stopping condition is presented in Algorithm 3; however, we describe a distributed stopping condition later in Corollary 5.4.

Properties of the qVCC algorithm are summarized in Proposition 5.3. Here, we let $N_{\max}$ be the maximum number of local constraints assigned to a node, i.e., $N_{\max} = \max_{i\in\{1,\dots,n\}} N_i$, and let $d_{\max}$ be the maximum in-degree of a node in the network, i.e., $d_{\max} = \max_{i\in\{1,\dots,n\}} |\mathcal{N}_{\text{in}}(i)|$.

PROPOSITION 5.3 (properties of the qVCC algorithm). *For a distributed random convex program (Problem 1) that satisfies Assumption 1, and the qVCC algorithm (Algorithm 3), the following statements hold:*

(i) *The local optimal objective function $J_i^*(t) \doteq J^*(V_i(t))$ is monotonically nondecreasing in the iterations t;*

(ii) *in $T \le \lceil \frac{N_{\max}}{m} \rceil \frac{(d_{\max}+1)^{\mathrm{diam}(\mathcal{G})}-1}{d_{\max}}$ iterations, the local solution at a generic node i converges to the scenario solution of the RCP;*

Algorithm 3: Quantized Vertex Constraints Consensus (qVCC)

Input : a, X, C_i, $\texttt{dm} = \texttt{diam}(\mathcal{G})$, m;
Output : $x^*(C)$, $J^*(C)$, and $\texttt{vert}(C)$;

% Initialization:
$V_i(0) = \texttt{vert}(C_i)$, $T_i(0) = \texttt{vert}(C_i)$, and $\texttt{stop = 0}$;
$t = 0$;

% qVCC iterations:
while $\texttt{stop} = 0$ **do**
  % Build local message $M_i(t)$ by selecting the first m constraints in $T_i(t)$
  % Poll neighbors and build:
  $L_i(t+1) = V_i(t) \cup \big(\cup_{j\in\mathcal{N}_{\text{in}}(i)} M_j(t)\big)$;
  % Update candidate set and transmission set:
  $V_i(t+1) = \texttt{vert}\,(L_i(t+1))$;
  $T_i(t+1) = T_i(t)\backslash\{M_i(t) \cup \big(V_i(t)\backslash V_i(t+1)\big)\} \oplus \{V_i(t+1)\backslash V_i(t)\}$;
  % Check stopping condition:
  if (all nodes have empty transmission set) **then**
    $\texttt{stop} = 1$;
  $t = t + 1$;
% Compute optimal solution and optimal objective:
$x_i^*(t) = x^*(V_i(t))$, $J_i^*(t) = J^*(V_i(t))$;
return $x_i^*(t)$, $J_i^*(t)$, $V_i(t)$;

(iii) *for each node i the local candidate set $V_i(T)$ satisfies $V_i(T) = \texttt{vert}(C) \supseteq \texttt{Sc}(C)$.*

Proof. See Appendix A.4.

We notice that the upper bound on T obtained in Proposition 5.3 corresponds to the worst case in which all constraints in the local sets need to be transmitted among the nodes. In principle, this bound can be used as a stopping condition for the qVCC algorithm; however, the bound may be pessimistic in practice, and therefore we now present an alternative distributed stopping rule for the qVCC algorithm.

Corollary 5.4 (stopping rule for the qVCC algorithm). *For the qVCC algorithm, if at time t all the transmission sets $T_i(t)$, $i \in \{1, \dots, n\}$, are empty, then the qVCC algorithm has converged to the scenario solution of the random convex program $P[C]$. Moreover, such a situation can be autonomously detected by each node in $\texttt{diam}(\mathcal{G})$ iterations.*

Proof. If at time t the transmission sets are empty, a generic node i satisfies $V_i(t+1) = V_i(t)$ (no message is received from the incoming neighbors). Moreover, the update rule of the transmission set becomes $T_i(t+1) = T_i(t)\backslash\{M_i(t) \cup \big(V_i(t)\backslash V_i(t+1)\big)\} \oplus \{V_i(t+1)\backslash V_i(t)\} = \emptyset$. Therefore, the local candidate set and the transmission set remain unchanged for all future iterations, i.e., the qVCC algorithm has converged.

Regarding the second statement, we notice that each node having nonempty transmission set can communicate this situation to all other nodes in $\texttt{diam}(\mathcal{G})$ iterations. Therefore, if for $\texttt{diam}(\mathcal{G})$ iterations no node notifies that the local transmission set is nonempty, all transmission sets need be empty, and convergence is reached. $\square$

According to Corollary 5.4, one may implement the stopping condition of

Algorithm 3 through the exchange of binary flags among nodes. In particular, each node with a nonempty transmission set reports this information to the neighbors by broadcasting a binary flag, and each node may stop iterating the algorithm if no flag is received for $\mathtt{diam}(\mathcal{G})$ iterations.

6. Distributed RCP with violated constraints. The RCP framework allows the generalization of the probabilistic guarantees of the scenario solution to the case in which r constraints are purposely violated with the aim of improving the objective value $J^*(C)$. Given a problem $P[C]$ and a set $R_r \subset C$, with $|R_r| = r$, RCP theory provides a bound for the probability that the solution $x^*(C \backslash R_r)$ satisfies a future realization of the random constraints; see [2] (recall that in our notation $x^*(C \backslash R_r)$ is the optimal solution of an RCP with constraint set $C \backslash R_r$).

In this section we study distributed strategies for removing constraints from an RCP. The RCP theory allows generic constraints removal procedures, with the only requirement being that the procedure is permutation invariant (i.e., changing the order of the constraints in C must not change the constraints removed by the procedure). We now present a distributed procedure for removing r constraints. The procedure works as follows: at each outer iteration, the nodes perform one of the distributed algorithms presented before (i.e., ACC, VCC, or qVCC). After attaining convergence, each node selects the constraint c with largest Lagrange multiplier (since nodes share the same set of candidate constraints after convergence, they will choose the same constraint), and each node removes the constraint c from the local constraint set. The distributed procedure is then repeated for r outer iterations (i.e., it terminates after removing the desired number of constraints, r). The distributed constraints removal procedure is summarized in Algorithm 4. The acronym CC in Algorithm 4 refers to one of the distributed algorithms presented in the previous sections (i.e., ACC, VCC, or qVCC).

We now state some properties of the distributed constraints removal procedure.

PROPOSITION 6.1 (distributed constraints removal). *The distributed constraints removal procedure in Algorithm* 4 *is permutation invariant. Moreover, if the active constraints consensus algorithm is used for distributed computation of the solution to the RCP in Algorithm* 4*, and assuming that strong duality holds, then the set of*

Algorithm 4: Distributed Constraints Removal

```
Input   : a, X, dm = diam(G), and r;
Output  : x*(C\R_r), J*(C\R_r), and R_r;

% Initialization:
η = 0, R_η = ∅;

% Outer iterations:
while η < r do
    compute [x*_η, J*_η, L_η] = CC(a, X, C_i, dm, [m]);
    select c ∈ L_η with largest Lagrange multiplier;
    C_i = C_i\{c}, and R_{η+1} = R_η ∪ {c};
    η = η + 1;

% Compute optimal solution and optimal objective:
[x*_r, J*_r, L_r] = CC(a, C_i, dm, [m]);

return x*_r, J*_r, R_r;
```

removed constraints corresponds to the set computed with the centralized constraints removal based on marginal costs [2].

Proof. We start by establishing the first statement. We consider the case in which the active constraints consensus algorithm is used for implementing the distributed removal procedure. It follows from Proposition 4.1 that the local candidate set at each node after convergence coincides with the set of active constraints. Neither the set of active constraints nor the Lagrange multipliers depend on the order of the constraints in C, and therefore the removal procedure is permutation invariant. The permutation invariance of the distributed constraints removal based on the VCC algorithm can be demonstrated using similar arguments. The second statement is a straightforward consequence of the fact that, when strong duality holds, the active constraints are the only ones that have associated Lagrange multipliers greater than zero (*complementary slackness*); therefore, after performing the active constraints consensus algorithm, each node is guaranteed to know all the constraints with nonzero Lagrange multipliers, from which it can select the one with the largest multiplier. ∎

The constraints removal based on marginal costs [2] iteratively removes constraints with largest Lagrange multiplier from the initial set of constraints C. The practical importance of Proposition 6.1 is twofold. First, we can devise a distributed strategy (Algorithm 4) that removes outlying constraints from the original RCP and improves the optimal objective. This is relevant when the constraints are connected to noisy measurements of an unknown parameter and we are interested in discarding *outliers* in a distributed fashion. An application of Algorithm 4 to distributed estimation with outliers is reported in section 7.4. Second, the proposition guarantees that the distributed removal strategy allows us to discard the same constraints as a centralized strategy; this is a desirable condition since the effectiveness of the removal strategy influences the optimal objective of the RCP with violated constraints. We conclude this section with some comments on the trade-off between the use of the ACC and the VCC algorithms in the distributed removal procedure (Algorithm 4). First, we notice that the active constraints consensus algorithm is able to return a constraint set only in feasible problems (otherwise the active constraint set is empty by convention); therefore, the ACC-based removal procedure applies only to feasible problem instances. On the other hand, under Assumption 1, the VCC-based removal procedure applies in the infeasible case as well. However, when using the VCC (or the qVCC), it is not possible to establish the parallel with the centralized case, since it is possible to have constraints with nonzero Lagrange multipliers that are not in the set computed by the VCC algorithm.

7. Applications and numerical examples.

7.1. Distributed ellipsoidal estimation.

In this section we discuss the problem of determining a confidence ellipsoid for an unknown random parameter. We study this problem considering three settings: (i) nodes in a network can directly measure the parameter (section 7.1.1), (ii) nodes can measure a linear function of the parameter (section 7.1.2), and (iii) nodes may take linear measurements of the parameter using possibly different measurement models (section 7.1.3).

7.1.1. Computing a confidence ellipsoid.

In this section we discuss the problem of determining a confidence ellipsoid for an unknown random parameter $y \in \mathbb{R}^q$ for which N iid realizations $y^{(j)}$, $j \in \{1, \dots, N\}$, are available. We consider first the case in which all the N realizations are collected at a single unit that solves the problem in a centralized way, and then outline a distributed setup of this problem in Remark 8.

A generic (bounded) ellipsoid, parameterized by its center $\hat{y} \in \mathbb{R}^q$ and shape matrix $W_y \in \mathbb{R}^{q \times q}$, $W_y \succ 0$, is represented as

$$\mathcal{E}_y = \{y \in \mathbb{R}^q : (y - \hat{y})^\top W_y (y - \hat{y}) \leq 1\}. \tag{7.1}$$

As a measure of size of $\mathcal{E}_y$ we consider the volume, which is proportional to the square root of the determinant of W_y^{-1}. Then, the problem of finding the smallest ellipsoid enclosing the given realizations can be stated in the form of the convex optimization problem,

$$\begin{aligned} &\min_{\hat{y}, W_y \succ 0} \ \texttt{log det}(W_y^{-1}) \qquad \text{subject to} \\ &\qquad (y^{(j)} - \hat{y})^\top W_y (y^{(j)} - \hat{y}) \leq 1 \quad \text{for each } j \in \{1, \dots, N\}, \end{aligned} \tag{7.2}$$

where $\texttt{log det}(\cdot)$ returns the logarithm of the determinant of a positive definite matrix. The number of variables in this problem is $q(q+3)/2$, corresponding to q variables describing the center $\hat{y}$, plus $q(q+1)/2$ variables describing the free entries in the symmetric matrix W_y. We can convert the optimization problem (7.2) into an equivalent one having linear cost function by introducing a slack variable (see Remark 3.1 in [2]); the dimension of the problem with linear objective is then $d = q(q+3)/2 + 1$. Since the vectors $y^{(j)}$ are iid realizations of the random variable y, problem (7.2) clearly belongs to the class of RCPs. Moreover, this problem is always feasible, and its solution is unique (see, for instance, section 3.3 in [23]). Therefore, we can apply (3.3) to conclude a priori, i.e., before knowing the actual realizations, that with high probability $1 - \beta$ (here, β is typically set to a very low value, say $\beta = 10^{-9}$) the ellipsoid computed via (7.2) is a $(1-\epsilon)$-confidence ellipsoid for y, with $\epsilon = 2(\log \beta^{-1} + d - 1)/N$. In other words, we know with practical certainty that $\mathcal{E}_y$ contains y with probability larger than $1 - \epsilon$, i.e., it encloses a probability mass of at least $1 - \epsilon$ of y. Furthermore, we observe that the constraints in (7.2) are convex functions also with respect to the "uncertainty" terms $y^{(j)}$; hence this problem satisfies Assumption 1, enabling the application of the VCC or qVCC algorithm.

Remark 8 (distributed computation of measurement ellipsoid). The solution to the optimization problem (7.2) can be computed in a distributed fashion using any of the algorithms proposed in this paper; for instance, one may consider a setup in which n nodes are available, and each node initially knows N_i local realizations of y, with $\sum_{i=1}^n N_i = N$. The application of the ACC, the VCC, or the qVCC algorithm entails that each node iteratively exchanges a subset of realizations $y^{(j)}$ with its neighbors in order to reach a consensus on the set of realizations defining the optimal solution to (7.2).

7.1.2. Ellipsoidal parameter estimation in a linear model. We now extend the previous setup and consider the case in which linear measurements y of an unknown parameter θ are used to infer an ellipsoid of confidence for the parameter itself. Consider the classical situation in which y is related to θ via a linear model

$$y = F\theta, \tag{7.3}$$

with $F \in \mathbb{R}^{q \times p}$, where θ is the input parameter and y is a measured output. Suppose that $\theta^{(1)}, \dots, \theta^{(N)}$ are N iid realizations of the unobservable parameter θ, and that $y^{(1)}, \dots, y^{(N)}$ are the corresponding observed measurements $y^{(i)} = F\theta^{(i)}$. We first consider the centralized case in which a single node uses the measurements to infer an ellipsoid of confidence for θ. Given the observations $y^{(1)}, \dots, y^{(N)}$, we can compute a unique minimum-size ellipsoid $\mathcal{E}_y$ containing the observations by solving problem (7.2). From the reasoning in section 7.1.1 we know with practical certainty that $\mathcal{E}_y$ is a $(1-\epsilon)$-confidence ellipsoid for y. Now, the condition $y \in \mathcal{E}_y$, together with the linear relation in (7.3), implies that the set of parameters θ that is compatible with output $y \in \mathcal{E}_y$ is a (possibly unbounded) ellipsoid $\mathcal{E}$ described by the quadratic inequality condition $(F\theta - \hat{y})^\top W_y(F\theta - \hat{y}) \leq 1$, that is,

$$\begin{bmatrix} \theta \\ 1 \end{bmatrix}^\top \begin{bmatrix} F^\top W_y F & F^\top W_y \hat{y} \\ * & \hat{y}^\top W_y \hat{y} - 1 \end{bmatrix} \begin{bmatrix} \theta \\ 1 \end{bmatrix} \leq 0. \tag{7.4}$$

Since $y \in \mathcal{E}_y$ if and only if $\theta \in \mathcal{E}$, and since with practical certainty $\mathbb{P}\{y \in \mathcal{E}_y\} \geq 1-\epsilon$, we also have that $\mathbb{P}\{\theta \in \mathcal{E}\} \geq 1-\epsilon$, and hence we found a region $\mathcal{E}$ within which θ must be contained with probability no smaller than $1-\epsilon$.

In the next section, we provide an extension of this linear estimation framework to a distributed setup in which n nodes collect linear measurements of θ, using possibly heterogeneous models.

7.1.3. Ellipsoidal parameter estimation in heterogeneous network. Suppose that there are n_s subsets of nodes, say $\mathcal{V}_1, \dots, \mathcal{V}_{n_s}$, such that each node in $\mathcal{V}_j$ uses the same linear measurement model,

$$y_i = F_j \theta \quad \text{for each } i \in \mathcal{V}_j, \tag{7.5}$$

and it collects N_i measurements,

$$y_i^{(k)} = F_j \theta^{(k)} \quad \text{for each } k \in \{1, \dots, N_i\},$$

where $\theta^{(k)}$, $k \in \{1, \dots, N_i\}$, are iid. Moreover, it is assumed that realizations of θ available at a node i are independent from realizations available at node j for each i, j. We here detail the procedure for computing a confidence ellipsoid for θ by first assuming a centralized case in which all measurements from nodes in $\mathcal{V}_j$ are available at a central node, and then we refer to Remark 9 for an outline of the corresponding distributed implementation.

If all measurements from nodes in $\mathcal{V}_j$ are available to a central computational unit, then this unit can first construct (by solving problem (7.2)) an ellipsoid of confidence $\mathcal{E}_y^j$ for the collective measurements $y_i^{(k)}$, $i \in \mathcal{V}_j$, $k \in \{1, \dots, N_i\}$,

$$\mathcal{E}_y^j = \{y : \ (y - \hat{y}_j)^\top W_j (y - \hat{y}_j) \leq 1\},$$

and then infer an ellipsoid of confidence $\mathcal{E}_j$ for θ according to (7.4),

$$\mathcal{E}_j = \left\{ \theta \in \mathbb{R}^p : \begin{bmatrix} \theta \\ 1 \end{bmatrix}^\top \begin{bmatrix} F_j^\top W_j F_j & F_j^\top W_j \hat{y}_j \\ * & \hat{y}_j^\top W_j \hat{y}_j - 1 \end{bmatrix} \begin{bmatrix} \theta \\ 1 \end{bmatrix} \leq 0 \right\}.$$

This procedure can be repeated for each $\mathcal{V}_j$, $j \in \{1, \dots, n_s\}$, thus obtaining n_s ellipsoidal sets $\mathcal{E}_j$ that (with practical certainty) contain θ with probability no smaller than $1-\epsilon_j$. "Fusing" the information from all the confidence ellipsoids $\mathcal{E}_j$, a standard probabilistic argument leads to stating that (again with practical certainty) the unknown parameter is contained in the intersection $\mathcal{I} = \cap_{j=1}^{n_s} \mathcal{E}_j$ with probability no smaller than $\mu \doteq \prod_{j=1}^{n_s}(1-\epsilon_j)$. Clearly, any set that contains the intersection $\mathcal{I}$ has a probability no smaller than μ of containing θ. We may then find an ellipsoid $\mathcal{E}$ covering the intersection $\mathcal{I}$ as follows. We describe the to-be-computed ellipsoid $\mathcal{E}$ as

$$\begin{bmatrix} \theta \\ 1 \end{bmatrix}^\top \begin{bmatrix} W & W\hat{\theta} \\ * & \hat{\theta}^\top W \hat{\theta} - 1 \end{bmatrix} \begin{bmatrix} \theta \\ 1 \end{bmatrix} \leq 0,$$

where $\hat{\theta}$ is the center of the ellipsoid and $W \succ 0$ is its shape matrix. Then a sufficient condition for $\mathcal{E}$ to contain $\mathcal{I}$ can be obtained through the so-called $\mathcal{S}$-procedure [24]: if there exist n_s scalars $\tau_j \geq 0$, $j \in \{1, \ldots, n_s\}$, such that

$$\begin{bmatrix} W & W\hat{\theta} \\ * & \hat{\theta}^\top W \hat{\theta} - 1 \end{bmatrix} - \sum_{j=1}^{n_s} \tau_j \begin{bmatrix} F_j^\top W_j F_j & F_j^\top W_j \hat{y}_j \\ * & \hat{y}_j^\top W_j \hat{y}_j - 1 \end{bmatrix} \preceq 0,$$

then $\mathcal{E} \supseteq \cap_{j=1}^{n_s} \mathcal{E}_j$. Defining a vector $\tilde{\theta} = W\hat{\theta}$, we can write the previous condition as

$$\begin{bmatrix} W - \sum_{j=1}^{n_s} \tau_j (F_j^\top W_j F_j) & \tilde{\theta} - \sum_{j=1}^{n_s} \tau_j (F_j^\top W_j \hat{y}_j) \\ * & -1 - \sum_{j=1}^{n} \tau_j (\hat{y}_j^\top W_j \hat{y}_j - 1) \end{bmatrix} + \begin{bmatrix} \mathbf{0}_p \\ \tilde{\theta}^\top \end{bmatrix} W^{-1} \begin{bmatrix} \mathbf{0}_p \\ \tilde{\theta}^\top \end{bmatrix}^\top \preceq 0,$$

where $\mathbf{0}_p$ is a matrix in $\mathbb{R}^{p \times p}$ with all zero entries. Using the Schur complement rule, this latter condition is equivalent to the following LMI in W, $\tilde{\theta}$, and $\tau_1, \ldots, \tau_{n_s}$:

$$\begin{bmatrix} W - \sum_{j=1}^{n_s} \tau_j (F_j^\top W_j F_j) & \tilde{\theta} - \sum_{j=1}^{n_s} \tau_j (F_j^\top W_j \hat{y}_j) & \mathbf{0}_p \\ * & -1 - \sum_{j=1}^{n} \tau_j (\hat{y}_j^\top W_j \hat{y}_j - 1) & \tilde{\theta}^\top \\ * & * & W \end{bmatrix} \preceq 0. \tag{7.6}$$

Then, the shape matrix W of the minimum volume ellipsoid $\mathcal{E} \supseteq \mathcal{I}$ can be computed by solving the following convex program:

$$\min_{\tilde{\theta}, W \succ 0, \tau_1 \geq 0, \ldots, \tau_{n_s} \geq 0} \ \texttt{log det}(W^{-1}) \quad \text{subject to (7.6).} \tag{7.7}$$

After obtaining the optimal solution of problem (7.7), the center of the minimum volume ellipsoid can be computed as $\hat{\theta} = W^{-1}\tilde{\theta}$.

Remark 9 (distributed estimation in heterogeneous network). A distributed implementation of the procedure previously described goes as follows. We assume that each node $i \in \{1, \ldots, n\}$ knows all the measurement models $\{F_1, \ldots, F_{n_s}\}$ and acquires N_i measurements according to its own model F_j; see (7.5). Each node i then maintains n_s different local constraint sets C_i^j, $j \in \{1, \ldots, n_s\}$, simultaneously and initializes the jth set C_i^j to the local measurements set of node i, if $i \in \mathcal{V}_j$, or to the empty set, otherwise. Then, each node runs a distributed constraints consensus algorithm (either ACC, or VCC, or qVCC) simultaneously on each of its local constraint sets. In this way, upon convergence, each node has all the optimal ellipsoids $\mathcal{E}_j$, $j \in \{1, \ldots, n_s\}$. Once this consensus is reached, each node can compute locally the enclosing ellipsoid $\mathcal{E} \supseteq \cap_{j=1}^{n_s} \mathcal{E}_j$ by solving the convex program (7.7).

7.1.4. Numerical results on distributed ellipsoid computation. We now elucidate on the distributed ellipsoid computation with some numerical examples. In particular, we demonstrate the effectiveness of our algorithms for (i) distributed computation of the enclosing ellipsoid when each node can measure the random parameter θ with the same measurement model, (ii) parallel computation of the enclosing ellipsoid, and (iii) distributed computation of the enclosing ellipsoid when each node can measure only some components of the random parameter θ.

Example 1 (distributed estimation in homogeneous sensor network). Consider the setup in which n sensors measure a random variable θ, using the same measurement model $y = F\theta$ (homogeneous sensor network), where we set for simplicity $F = I_p$ (the identity matrix of size p). We assumed $\theta \in \mathbb{R}^2$ to be distributed according to the following mixture distribution:

$$\theta = \begin{cases} \gamma_1 & \text{with probability } 0.95, \\ \gamma_2 + 10\gamma_1 & \text{with probability } 0.05, \end{cases}$$

where $\gamma_1 \in \mathbb{R}^2$ is a standard Normal random vector and $\gamma_2 \in \mathbb{R}^2$ is uniformly distributed in $[-1, 1]^2$. The overall number of measurements (acquired by all nodes) is $N = 20000$; the size of the local constraint sets is N/n. We consider the case in which the nodes in the network solve the RCP in (7.2) using one of the algorithms proposed in this paper. We consider two particular graph topologies: *chain graphs* and *geometric random graphs*. For the geometric random graph, we picked nodes uniformly in the square $[0, 1]^2$ and chose a communication radius $r_c > 2\sqrt{2}\sqrt{\log(n)/n}$ to ensure that the graph is strongly connected with high probability [25]. In Table 7.1 we report the maximum number of iterations and the maximum number of exchanged constraints for each algorithm. Statistics are computed over 20 experiments. The active constraints consensus algorithm requires nodes to exchange a small number of constraints, and it converges in a number of iterations that grows linearly in the graph diameter. For the VCC algorithm, the maximum number of iterations for convergence is equal to the graph diameter. For the considered problem instances, the number of constraints to be exchanged among the nodes is small. We picked $m = 5$ for the qVCC algorithm. Table 7.1 reports the number of iterations required by the qVCC to meet the halting conditions described in Corollary 5.4.

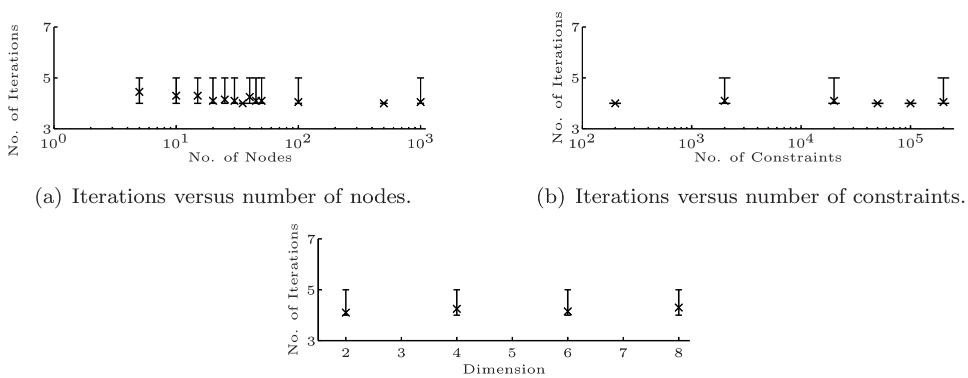
Example 2 (parallel computation of confidence ellipsoid). In this example we consider the same setup as in Example 1, but we solve the RCP (7.2) in a distributed fashion assuming a complete communication graph. A complete communication graph

Table 7.1

Distributed computation in homogeneous sensor network: maximum number of iterations, maximum number of exchanged constraints, and diameter for different graph topologies, and for each of the proposed algorithms.

	No. nodes	Diameter	ACC		VCC		qVCC	
			Iter.	Constr.	Iter.	Constr.	Iter.	Constr.
Geometric random graph	10	1	5	6	1	19	6	5
	50	2	7		2		8	
	100	3	10		3		9	
	500	5	16		5		13	
Chain graph	10	10	36	5	10	23	21	5
	50	50	187		50		101	
	100	100	375		100		200	
	500	500	1910		500		1000	

describes a parallel computation setup in which each processor can interact with every other processor. In this case, we focus on the active constraints consensus algorithm. In Figure 7.1 we report the dependence of the number of iterations on the number of nodes, number of constraints, and dimensions of the parameter $y = \theta$ to be estimated. In the considered problem instances the iterations of the active constraints consensus algorithm do not show any dependence on these three factors. In Figure 7.2 we show some statistics on the number of exchanged constraints. In particular, we compare the number of constraints exchanged among nodes at each communication round with the dimension $d = p(p+3)/2 + 1$ (recall that $p = q$ in this example) of the RCP (section 7.1.1): in Proposition 4.1 we concluded that the number of constraints exchanged at each communication round is bounded by d. Figure 7.2 shows that in the considered problem instances, the number of constraints is below this upper bound, which is shown as a dashed line. For space reasons we do not report results on the dependence of the number of exchanged constraints on the total number of constraints N and on the number of nodes n. In our test the number of exchanged constraints was practically independent of these two factors and remained below 5 in all tests.



(a) Iterations versus number of nodes.

(b) Iterations versus number of constraints.

(c) Iterations versus dimension of θ.

FIG. 7.1. *Parallel computation of confidence ellipsoid using the active constraints consensus algorithm.* (a) *Number of iterations required for convergence with different numbers of nodes, with fixed number of constraints $N = 20000$, and fixed dimension $p = 2$ of θ;* (b) *number of iterations for different numbers of constraints, with fixed number of nodes $n = 50$ and fixed dimension $p = 2$;* (c) *number of iterations for different dimensions of θ, with fixed number of nodes $n = 50$ and number of constraints $N = 20000$. In each figure the cross denotes the average number of iterations, whereas the bar defines the maximum and the minimum observed number of iterations.*

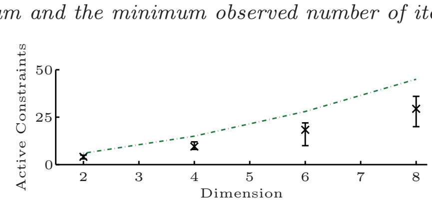
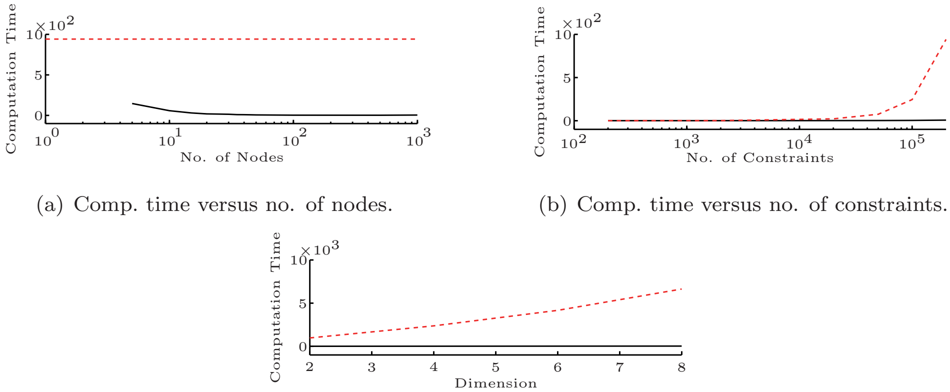


FIG. 7.2. *Parallel computation of confidence ellipsoid using the active constraints consensus algorithm. The bars represent number of constraints exchanged among nodes for different dimensions p of θ, with fixed number of constraints $N = 20000$ and fixed number of nodes $n = 50$. The cross denotes the average number of constraints, whereas the bar defines the maximum and the minimum observed numbers of exchanged constraints. The dashed line denotes maximum number of constraints in generic position $d = p(p+3)/2 + 1$.*



(a) Comp. time versus no. of nodes.

(b) Comp. time versus no. of constraints.

(c) Comp. time versus dimension of θ.

FIG. 7.3. *Parallel computation of confidence ellipsoid. The solid black line represents the parallel computation time required for solving the RCP using the active constraints consensus algorithm, and the dashed line represents the computation time required for centralized solution of the RCP.*

In Figure 7.3 we compare the computational effort required by the active constraints consensus algorithm in the parallel setup with a standard centralized solver in charge of solving the convex program (7.2). We used `CVX/SeDuMi` [26] as a centralized parser/solver, and we compared the computation times required to solve the problem, for different numbers of nodes, numbers of constraints, and dimensions of the parameter θ. The use of the active constraints consensus algorithm provides a remarkable advantage in terms of computational effort. For a large number of constraints, this advantage is significant even for a small number of processors.

Remark 10 (computational advantage of the active constraints consensus algorithm). The computational complexity of a centralized approach for convex programming depends on the actual algorithm used for the solution, and it is typically a function of the number of variables d and the number of constraints N. For a fixed dimension d, the complexity of a generic algorithm can be approximately estimated as $\mathcal{O}(N^\alpha)$, where the scalar α depends on the problem instance (see Lecture 5 in [27]). At each iteration of the active constraints consensus algorithm, instead, each node i solves a smaller convex program with at most $\frac{N}{n} + d(n-1)$ constraints (local constraints plus at most d constraints received by the remaining $n-1$ nodes in the network). As discussed in the previous section, numerical evidence shows that the number of iterations depends only on the diameter of the graph and, according to Figure 7.1, it is always below 5 in a parallel setup (complete graph). Therefore, the asymptotic complexity of the active constraints consensus algorithm in a parallel setup amounts to $\mathcal{O}\left((\frac{N}{n} + d(n-1))^\alpha\right)$. Assuming that $N \gg n$ (which is common in the RCP framework), the bound simplifies to $\mathcal{O}\left((\frac{N}{n})^\alpha\right)$. Therefore, the computation complexity reduces by order of $\frac{1}{n^\alpha}$ when passing from a centralized solver to a parallel implementation using the active constraints consensus algorithm.

Example 3 (distributed estimation in heterogeneous sensor network). We now consider the distributed computation of a parameter ellipsoid in a network with n nodes. We assume that half of the nodes in the network take measurements of $\theta \in \mathbb{R}^2$ according to the measurement model $y_1 = F_1\theta$, where $F_1 = [1\ 0]$; the remaining nodes use the measurement model $y_2 = F_2\theta$, where $F_2 = [0\ 1]$. We consider θ distributed according to a mixture distribution, as in Example 1. The nodes acquires 20000 mea-

TABLE 7.2
Distributed estimation in heterogeneous sensor network: maximum number of iterations, maximum number of exchanged constraints, and diameter for different graph topologies for the ACC and VCC algorithms.

	No. nodes	Diameter	ACC		VCC	
			Iter.	Constr.	Iter.	Constr.
Geometric random graph	10	1	4	4	1	4
	50	2	7		2	
	100	3	10		3	
	500	5	16		5	
Chain graph	10	10	28	4	10	4
	50	50	148		50	
	100	100	298		100	
	500	500	1498		500	

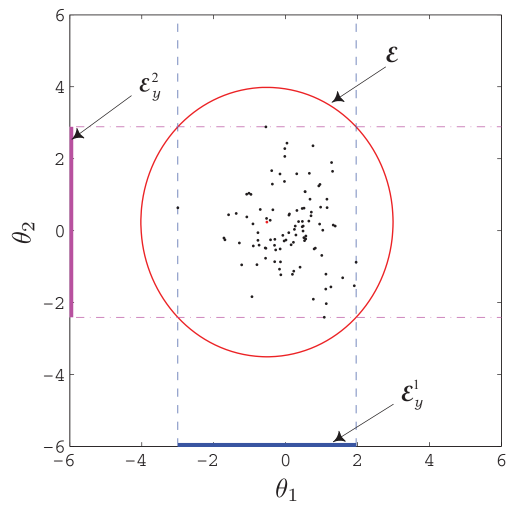


FIG. 7.4. *Distributed estimation in heterogeneous sensor network: the black dots are* 100 *realizations of the random parameter* $\theta = [\theta_1\ \theta_2]^\top$. *Nodes with measurement model* 1 *can measure* $y_1 = F_1\theta = [1\ 0]\ \theta = \theta_1$ *and compute the corresponding measurement set* $\mathcal{E}_y^1$ *(shown as a solid blue line) and the set* $\mathcal{E}_1$ *(the strip delimited by dashed blue lines) of parameters compatible with* $\mathcal{E}_y^1$. *Similarly, nodes with measurement model* 2 *can measure* $y_2 = F_2\theta = [0\ 1]\ \theta = \theta_2$ *from which the network builds the set* $\mathcal{E}_y^2$ *(shown as a solid magenta line) and the set* $\mathcal{E}_2$ *(the strip delimited by dashed magenta lines) of parameters compatible with* $\mathcal{E}_y^2$. *From the sets* $\mathcal{E}_1$ *and* $\mathcal{E}_2$, *each node can compute the bounding ellipsoid* $\mathcal{E} \supseteq \mathcal{E}_1 \cap \mathcal{E}_2$, *by solving problem* (7.7). *(See online version for color.)*

surements for each measurement model. They then estimate the set $\mathcal{E}$ according to Remark 9. In Table 7.2 we report some statistics related to the computation of the sets $\mathcal{E}_1$ and $\mathcal{E}_2$ using the ACC and the VCC algorithms; see Remark 9. After the computation of $\mathcal{E}_1$ and $\mathcal{E}_2$, each node can locally retrieve the set $\mathcal{E}$ solving problem (7.7); see Figure 7.4.

According to section 7.1.3 we can conclude that for $j \in \{1, 2\}$, with confidence level $1 - \beta = 1 - 10^{-8}$, $\mathcal{E}_j$ is a $(1 - \epsilon_j)$-confidence ellipsoid for θ, with $\epsilon_j = 2 \cdot 10^{-3}$. Then, with practical certainty the ellipsoid $\mathcal{E}$ is a μ-confidence ellipsoid for θ, with $\mu = (1 - \epsilon_1)(1 - \epsilon_2) \approx 0.995$.

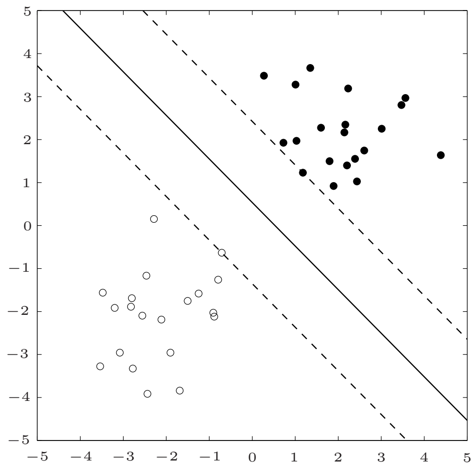


FIG. 7.5. *Binary linear classification: two clouds of points having labels* $+1$ *(full circles) and* -1 *(empty circles), respectively, need to be separated by a hyperplane* $\mathcal{H}$*, which maximizes the margin of separation between the classes.*

7.2. Distributed linear classification. A classical problem in binary linear classification is to determine a linear decision surface (a hyperplane) separating two clouds of binary labeled multidimensional points, so that all points with label $+1$ fall on one side of the hyperplane and all points with label -1 on the other side; see Figure 7.5. Formally, one is given a set of data points (*features*) $b_j \in \mathbb{R}^p$, $j \in \{1, \dots, N\}$, and the corresponding class label $l_j \in \{-1, +1\}$, and seeks a suitable hyperplane $\mathcal{H} = \{s \in \mathbb{R}^p : \theta^\top s + \rho = 0\}$, with $\theta \in \mathbb{R}^p$ and $\rho \in \mathbb{R}$, such that features with different labels belong to different half-spaces with respect to $\mathcal{H}$, and the margin of separation between the classes is maximized (maximum margin classifier; see [28]). If the features are linearly separable, then the optimal separating hyperplane solves the following minimization problem [29]:

$$\min_{\theta,\rho} \quad \|\theta\|_2 \qquad \text{subject to} \qquad \text{(7.8)}$$
$$l_j({b_j}^\top \theta + \rho) \geq 1 \quad \text{for each } j \in \{1, \dots, N\}.$$

To deal with possibly infeasible problem instances (i.e., nonlinearly separable data), it is common to include a slack variable, allowing (but penalizing) misclassification:

$$\min_{\theta,\rho,\nu\geq 0} \quad \|\theta\|_2 + \nu \qquad \text{subject to} \qquad \text{(7.9)}$$
$$l_j({b_j}^\top \theta + \rho) \geq 1 - \nu \quad \text{for each } j \in \{1, \dots, N\}.$$

If the observed datum/label pairs $\delta^{(j)} = (b_j, l_j)$, $j \in \{1, \dots, N\}$, are interpreted as realizations of a random datum/label variable $\delta = (b, l)$, then problem (7.9) is an instance of the following RCP in dimension $d = p + 3$:

$$\min_{\theta,\rho,\phi\geq 0,\nu\geq 0} \quad \phi \quad \text{subject to} \qquad \text{(7.10)}$$
$$l_j({b_j}^\top \theta + \rho) \geq 1 - \nu \quad \text{for each } j \in \{1, \dots, N\}, \qquad \text{(7.11)}$$
$$\|\theta\|_2 + \nu \leq \phi.$$

Such an RCP is always feasible, and it admits a unique optimal solution with probability one; see, e.g., [29]. Therefore, we can apply (3.3) to conclude that with practical certainty the hyperplane $\mathcal{H}$, obtained as a solution of (7.10), remains an optimal separating hyperplane also after adding a new realization to the training data.

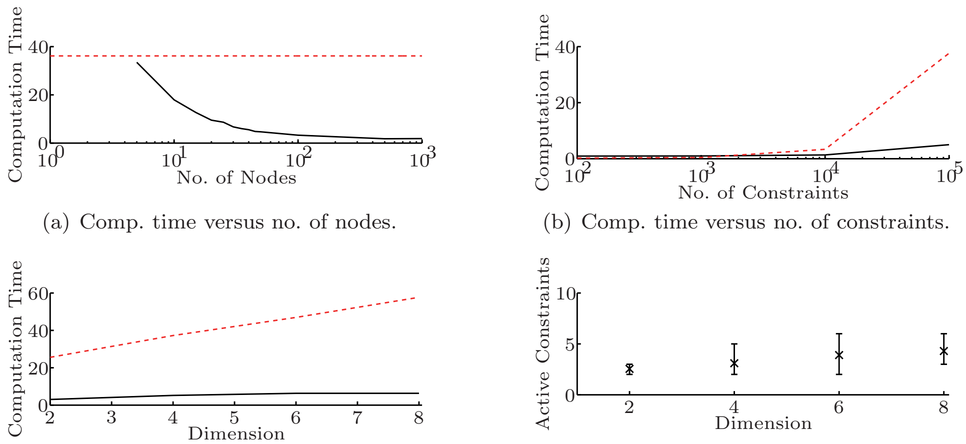
Problem (7.10) is readily amenable to a distributed solution via the active constraints consensus algorithm, by assuming that the N constraints in (7.11) are subdivided into n disjoint subsets of cardinality N_i each, $i \in \{1, \dots, n\}$, $\sum_{i=1}^{n} N_i = N$, and that each subset is assigned to a node as the local constraint set. The constraints in (7.11) are linear, and hence the problem can also be solved via the VCC or qVCC algorithm; see Remark 6.

7.2.1. Numerical results on distributed linear classification. We next present numerical examples of distributed and parallel computations of linear classifiers.

Example 4 (distributed linear classification). In this section we consider the case in which the training set $\delta^{(j)} = (b_j, l_j)$, $j \in \{1, \dots, N\}$, is not known at a central computational unit, but its knowledge is distributed among several nodes. An example of this setup can be the computation of a classifier for spam filtering [30], where the datum/label pairs are collected by the personal computers of n users, and the n computers may interact for computing the classifiers. For our numerical experiments we considered a problem in which the features with label "+1" are sampled from the normal distribution with mean $10 \times \mathbf{1}_p$, while features with label "-1" are sampled from the normal distribution with mean $-10 \times \mathbf{1}_p$. We used identity matrices as covariances for both distributions. After "sampling" the random constraints we distribute them among n nodes. Then we study the distributed computation of the solution to problem (7.10) on two network topologies: geometric random graphs and chain graphs. The performance of the ACC and VCC algorithms for $p = 4$ and $N = 20000$ total constraints is shown in Table 7.3. The values shown in the table are the worst-case values over 20 runs of the algorithms. It can be seen that the number of iterations required for the convergence of the active constraints consensus algorithm is linear in graph diameter, while it is equal to the graph diameter for the VCC algorithm. The number of constraints exchanged at each iteration among the nodes is small for the active constraints consensus algorithm, while this number is large for the VCC algorithm. Note that the local sets of constraints are initially disjoint (each node is entrusted with a different subset of constraints), but as the algorithms iterate, the same constraints may appear in different local sets, and upon convergence, the local constraints set at each node will contain the same subset of constraints, which defines the optimal solution of the RCP.

TABLE 7.3
Distributed linear classification: maximum number of iterations, maximum number of exchanged constraints, and diameter for different graph topologies for the ACC and VCC algorithms.

	No. nodes	Diameter	ACC		VCC	
			Iter.	Constr.	Iter.	Constr.
Geometric random graph	10	1	5	5	1	342
	50	2	11		2	
	100	3	11		3	
	500	5	24		5	
Chain graph	10	10	37	5	10	365
	50	50	177		50	
	100	100	319		100	
	500	500	1498		500	



(a) Comp. time versus no. of nodes.

(b) Comp. time versus no. of constraints.

(c) Comp. time versus dimension.

(d) Active constraints versus dimension.

FIG. 7.6. *Performance of the active constraints consensus algorithm for parallel computation of the solution of the linear classification problem.* (a)–(c) *The solid black and dashed lines represent parallel and centralized average computation time, respectively.* (d) *The cross represents the average number of active constraints, and the error bars represent the minimum and maximum numbers of active constraints for different problem dimensions.*

Example 5 (parallel linear classification). For the same set of data as in Example 4, we study the parallel computation of the optimal separating hyperplane. The parallel computation setup is modeled via a complete graph. The computation time of the active constraints consensus algorithm for parallel computation of the optimal separating hyperplane is shown in Figure 7.6. The computation time is averaged over 20 runs of the algorithm. The computation time is shown, respectively, as a function of number of processors for $p = 4$ and $N = 200000$ total constraints, as a function of total number of constraints for $p = 4$ and $n = 50$ processors, and as a function of dimension p for $N = 200000$ total constraints and $n = 50$ processors. In the first case, the minimum, average, and maximum numbers of active constraints are $2, 3.3$, and 5, respectively, while the minimum, average, and maximum numbers of iterations are $4, 4.04$, and 5, respectively. In the second case, the minimum, average, and maximum numbers of active constraints are $2, 3.09$, and 5, respectively, while the minimum, average, and maximum numbers of iterations are $4, 4.03$, and 6, respectively. In the third case, the minimum, average, and maximum numbers of iterations are $4, 4.04$, and 5, respectively, and the statistics of the constraints are shown in Figure 7.6. It can be seen that the parallel computation of the optimal solution via the active constraints consensus algorithm remarkably improves the computation time over the centralized computation. For a large number of constraints, this improvement is significant even for a small number of processors.

7.3. Parallel random model predictive control. Consider the LTI system

$$x_{t+1} = F(\xi)x_t + G(\xi)u_t + G_\gamma(\xi)\gamma_t, \tag{7.12}$$

where $t \in \mathbb{Z}_{\geq 0}$ is a discrete time variable, $x_t \in \mathbb{R}^p$ is the system state, $u_t \in \mathbb{R}^q$ is the control input, $\gamma_t \in \Gamma \subset \mathbb{R}^{q_\gamma}$ is an unmeasured disturbance vector, $\xi \in \Xi \subseteq \mathbb{R}^w$ is a vector of uncertain parameters, and $F(\xi) \in \mathbb{R}^{p\times p}, G(\xi) \in \mathbb{R}^{p\times q}, G_\gamma(\xi) \in \mathbb{R}^{p\times q_\gamma}$ are uncertain matrices. The design problem is to determine a control law that regulates the system state to some desired set, subject to some constraints on states and controls. In random model predictive control [31, 32], one picks a control law of the form $u_t = K_f x_t + v_t$, where $K_f \in \mathbb{R}^{q\times p}$ is the static linear terminal controller gain and $v_t \in \mathbb{R}^q$ is the design variable. The design variable v_t is picked to provide robustness with high probability. To determine the design variable that achieves such robustness, at each time t and for a given finite horizon length M, N realizations of the uncertain parameter ξ and disturbance vectors $(\gamma_t, \dots, \gamma_{t+M-1})$ are sampled and an optimization problem is solved. Let us denote these realizations by $(\xi^{(k)}, \gamma_t^{(k)}, \dots, \gamma_{t+M-1}^{(k)}), k \in \{1, \dots, N\}$, and define $g_t^{(k)} = [{\gamma_t^{(k)}}^\top \dots {\gamma_{t+M-1}^{(k)}}^\top]^\top$ for each $k \in \{1, \dots, N\}$. The design variable v_t is determined by the solution of the following optimization problem:

$$
\begin{aligned}
&\min_{\mathcal{V}_t} \quad \max_{k\in\{1,\dots,N\}} J(x_t, \mathcal{V}_t, \xi^{(k)}, g_t^{(k)}) \qquad \text{subject to} \\
&\qquad f_X(x_{t+j|t}^{(k)}) \leq 0, \\
&\qquad f_U(K_f x_{t+j|t}^{(k)} + v_{t+j-1}) \leq 0, \\
&\qquad f_{X_f}(x_{t+M|t}^{(k)}) \leq 0 \\
&\qquad \text{for each } j \in \{1, \dots, M\} \text{ and for each } k \in \{1, \dots, N\},
\end{aligned} \tag{7.13}
$$

where $J : \mathbb{R}^p \times \mathbb{R}^{qM} \times \mathbb{R}^w \times \mathbb{R}^{q_\gamma M} \to \mathbb{R}$ is a cost function that is convex in x_t and $\mathcal{V}_t$, $f_X : \mathbb{R}^p \to \mathbb{R}$, $f_U : \mathbb{R}^q \to \mathbb{R}$, and $f_{X_f} : \mathbb{R}^p \to \mathbb{R}$ are convex functions that capture constraints on the state at each time, the control at each time, and the final state, respectively, and

$$
\begin{aligned}
x_{t+j|t}^{(k)} &= (F_{\mathrm{cl}}(\xi^{(k)}))^{j-1} x_t + \Psi_j^{(k)} \mathcal{V}_t + \Upsilon_j^{(k)} g_t^{(k)}, \\
F_{\mathrm{cl}}(\xi^{(k)}) &= F(\xi^{(k)}) + G(\xi^{(k)}) K_f, \\
\Psi_j^{(k)} &= \left[(F_{\mathrm{cl}}(\xi^{(k)}))^{j-1} G(\xi^{(k)}) \dots F_{\mathrm{cl}}(\xi^{(k)}) G(\xi^{(k)}) \quad G(\xi^{(k)}) \quad 0 \dots 0 \right] \in \mathbb{R}^{p\times qM}, \\
\Upsilon_j^{(k)} &= \left[(F_{\mathrm{cl}}(\xi^{(k)}))^{j-1} G_\gamma(\xi^{(k)}) \dots F_{\mathrm{cl}}(\xi^{(k)}) G_\gamma(\xi^{(k)}) \quad G_\gamma(\xi^{(k)}) \quad 0 \dots 0 \right] \in \mathbb{R}^{p\times q_\gamma M}, \\
\mathcal{V}_t &= \left[v_t^\top \dots v_{t+M-1}^\top \right]^\top.
\end{aligned}
$$

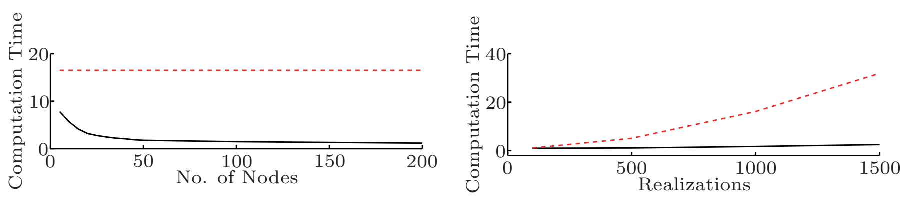
Problem (7.13) is an RCP of dimension $d = qM + 1$. Moreover, assuming that the problem admits a unique optimal solution with probability one and for $N > qM + 1$, for any realization of the parameter and the disturbance vector, the constraints on the state and the control are satisfied with expected probability at least $(N - qM)/(N + 1)$ [31]. Problem (7.13) is directly amenable to a distributed solution via the active constraints consensus algorithm. In the next section we consider the case in which the random constraints of the RCP are purposely distributed among n processors that have to solve the problem in parallel fashion.

7.3.1. Numerical results on parallel random model predictive control. In order to achieve robustness with high probability, a large number of realizations of the parameter and disturbances are needed in the RCP (7.13). This results in a large number of constraints and makes real-time centralized computation of the solution to the optimization problem (7.13) intractable. Therefore, we resort to the parallel computation of the solution to the optimization problem (7.13) via the active constraints consensus algorithm. We now apply the active constraints consensus algorithm to an example taken from [31] and show its effectiveness.

Example 6 (parallel random MPC). Consider the LTI system (7.12) with

$$F(\xi) = \begin{bmatrix} 1+\xi_1 & \frac{1}{1+\xi_1} \\ 0.1\sin(\xi_4) & 1+\xi_2 \end{bmatrix}, \quad G(\xi) = \begin{bmatrix} 0.3\arctan(\xi_5) \\ \frac{1}{1+\xi_3} \end{bmatrix}, \quad G_\gamma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

where each of the random parameters ξ_1, ξ_2, ξ_3 is uniformly distributed in the interval $[-0.1, 0.1]$, while ξ_4, ξ_5 are distributed according to Gaussian distributions with zero mean and unit variance. Let the horizon be $M = 10$ and the uncertainty γ be uniformly distributed over set $\Gamma = \{\gamma \in \mathbb{R}^2 \ : \ \|\gamma\|_\infty\} \leq 0.05$. Assume that $f_X(x) = \|x\|_\infty - 10$, $f_U(u) = |u| - 5$, and $f_{X_f}(z) = \|z\|_\infty - 1$. Consider the terminal controller gain $K_f = [-0.72 \ \ -1.70]$ and the cost function $J(x_t, \mathcal{V}_t, \xi^{(k)}, g_t^{(k)}) = \max_{j\in\{1,\dots,M\}} \max\{0, \|x_{t+j-1|t}^{(k)}\|_\infty - 1\} + \|\mathcal{V}_t\|_2^2$. For this set of data, the computation time of the active constraints consensus algorithm averaged over 20 runs of the algorithm for parallel computation of the solution to optimization problem (7.13) is shown in Figure 7.7. The computation time is shown, respectively, as a function of the number of processors for 1000 realizations of the random parameters and as a function of the number of realizations of the random parameters for 50 processors. In the first case, the minimum, average, and maximum numbers of active constraints are $2, 2.55$, and 6, respectively, while the minimum, average, and maximum numbers of iterations are $3, 3.73$, and 5, respectively. In the second case, the minimum, average, and maximum numbers of active constraints are $2, 2.18$, and 4, respectively, while the minimum, average, and maximum numbers of iterations are $3, 4.03$, and 5, respectively.



(a) Comp. time versus no. of nodes. (b) Comp. time versus no. of realizations.

FIG. 7.7. *Performance of the active constraints consensus algorithm for parallel random model predictive control. The solid black and dashed lines represent parallel and centralized average computation time, respectively.*

7.4. Example of distributed outliers rejection. We conclude the numerical part of this paper with a brief example of distributed constraints removal applied to the distributed estimation problem presented in section 7.1.1. We consider $n = 50$ sensors measuring a random variable θ, using the same measurement model of Example 1 (homogeneous sensor network). The overall number of measurements (acquired by all nodes) is $N = 3000$. The original scenario solution that satisfies all $N = 3000$ constraints can ensure a violation probability smaller than $\epsilon = 10^{-2}$ with confidence level greater than $1 - \beta = 1 - 2 \times 10^{-8}$. According to RCP theory we can remove $r = 165$ constraints, still guaranteeing that the violation probability is smaller than 10^{-1} with confidence level $1 - \beta$ close to 1. Therefore the nodes apply Algorithm 4 (the active constraints consensus algorithm is used within the removal strategy), computing a scenario solution which satisfies all but $r = 165$ constraints. Thus, with a little compromise over the bound on the violation probability, the constraints removal allows reducing the size of the ellipsoid, hence improving the informativeness of the confidence

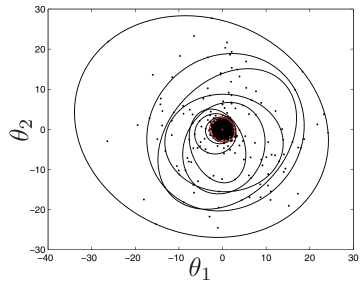


FIG. 7.8. *Measurements taken by all the sensors in the network (black dots) and confidence ellipsoids at one node after rejecting a number of outliers* $\eta = \{0, 20, 40, \dots, 140, 160\}$ *in Algorithm* 4. *The red ellipsoid is the one produced after discarding* $r = 165$ *measurements according to the* distributed constraints removal procedure *(See online version for color.)*

ellipsoid. In Figure 7.8, we report the confidence ellipsoids computed at one node using Algorithm 4, after rejecting a number of outliers $\eta = \{0, 20, 40, \dots, 140, 160\}$, together with the final ellipsoid satisfying all but $r = 165$ constraints.

8. Conclusion. In this paper, we studied distributed computation of the solution to random convex program (RCP) instances. We considered the case in which each node of a network of processors has local knowledge of only a subset of constraints of the RCP, and the nodes cooperate in order to reach the solution of the *global* problem (i.e., the problem including all constraints). We proposed two distributed algorithms, namely, the active constraints consensus algorithm and the vertex constraints consensus (VCC) algorithm. The active constraints consensus algorithm computes the solution in finite time and requires the nodes to exchange a small number of constraints at each iteration. Moreover, a parallel implementation of the active constraints consensus algorithm remarkably improves the computational effort compared to a centralized solution of the RCP. The VCC algorithm converges to the solution in a number of iterations equal to the graph diameter. We also developed a variant of the VCC algorithm, namely, quantized vertex constraints consensus (qVCC), that restricts the number of constraints to be exchanged at each iteration. We further proposed a distributed constraints removal strategy for outlier rejection within the framework of the RCP with violated constraints. Finally, we presented several applications of the proposed distributed algorithms, including estimation, classification, and random model predictive control.

Appendix.

A.1. Proof of Proposition 2.8. We start by establishing the first statement. Let c be a support constraint for a feasible problem in the form (2.1). Call $\hat{x}^* = x^*(C)$ and $\check{x}^* = x^*(C \backslash \{c\})$. From the definition of support constraints, it follows that $a^\top \check{x}^* < a^\top \hat{x}^*$. This inequality implies that $\check{x}^*$ does not satisfy c; otherwise we would have $\hat{x}^* = \check{x}^*$ and $a^\top \hat{x}^* = a^\top \check{x}^*$. Therefore, $f_c(\check{x}^*) > 0$. Assume by contradiction that c is not active at $\hat{x}^*$, i.e., that $f_c(\hat{x}^*) < 0$. Consider a point x on the segment connecting $\hat{x}^*$ and $\check{x}^*$: $x(\lambda) = (1 - \lambda)\check{x}^* + \lambda \hat{x}^*$, $\lambda \in [0,\ 1]$. It follows immediately that $a^\top x(\lambda) < a^\top \hat{x}^*$ for every $\lambda \in [0, 1)$. By convexity of the constraints it also holds that $f_c(x(\lambda)) = f_c\left((1 - \lambda)\check{x}^* + \lambda \hat{x}^*\right) \leq (1 - \lambda) f_c(\check{x}^*) + \lambda f_c(\hat{x}^*)$. For every $\bar{\lambda} \geq \frac{f_c(\check{x}^*)}{f_c(\check{x}^*) - f_c(\hat{x}^*)} \in (0, 1)$ the previous quantity is nonpositive, i.e., $f_c(x(\bar{\lambda})) \leq 0$; therefore the constraint c is satisfied at $x(\bar{\lambda})$. But then $x(\bar{\lambda})$ would satisfy all constraints and yield an objective value that improves upon that of $\hat{x}^*$. This contradicts optimality of $\hat{x}^*$, and hence proves that c must be active at $\hat{x}^*$.

We now establish the second statement. We first demonstrate that each essential set $E_i \doteq \mathtt{Es}_i(C)$ needs to be irreducible, i.e., $E_i = \mathtt{Sc}(E_i)$. By definition, each E_i is a minimum cardinality set satisfying $J^*(E_i) = J^*(C)$. Now assume by contradiction that there exists a constraint $c \in E_i$, such that $J^*(E_i) = J^*(E_i \backslash \{c\})$. This implies that there exists a set $E_i \backslash \{c\}$, which is also invariant for C, i.e., $J^*(E_i \backslash \{c\}) = J^*(E_i) = J^*(C)$, and has smaller cardinality than E_i, leading to a contradiction. Now we can prove the following statement: if each constraint in $\mathtt{Es}_i(C)$ is a support constraint for problem $P[\mathtt{Es}_i(C)]$, it needs to be active for the problem $P[\mathtt{Es}_i(C)]$; see claim (i). Consequently, if x_i^* is the optimal solution for $P[\mathtt{Es}_i(C)]$, then $f_j(x_i^*) = 0$ for all $c_j \in \mathtt{Es}_i(C)$. From the unique minimum condition and locality, it follows that

$$J^*(\mathtt{Es}_i(C)) = J^*(C) \implies x^*(\mathtt{Es}_i(C)) = x^*(C)$$

for each $i \in \{1, \ldots, n_e\}$. Therefore, $f_j(x^*(C)) = 0$ for each $c_j \in \mathtt{Es}_i(C)$, $i \in \{1, \ldots, n_e\}$, and $\mathtt{Ac}(C) \supseteq \cup_{i=1}^{n_e} \mathtt{Es}_i(C)$. ∎

A.2. Proof of Proposition 4.1. We start by establishing the first statement. According to the update rule of the active constraints consensus algorithm, the sequence of local optimal objective $J_i^*(t)$ satisfies

$$\begin{aligned} J_i^*(t+1) \doteq J^*\big(L_i(t+1)\big) &= J^*\big(A_i(t) \cup (\cup_{j \in \mathcal{N}_{\text{in}}(i)} A_j(t)) \cup C_i\big) \\ [\text{by monotonicity}] &\geq J^*\big(A_i(t)\big) \\ [\text{by Corollary 2.9}] &= J^*\big(L_i(t)\big) = J_i^*(t); \end{aligned}$$

then $J_i^*(t)$ is nondecreasing in t.

The proof of the second statement is more involved and works as follows. We first observe that, for any directed edge (i, j), it holds that

$$\begin{aligned} J_j^*(t+1) \doteq J^*\big(L_j(t+1)\big) &= J^*\big(A_j(t) \cup (\cup_{k \in \mathcal{N}_{\text{in}}(j)} A_k(t)) \cup C_j\big) \\ [\text{by monotonicity and } i \in \mathcal{N}_{\text{in}}(j)] &\geq J^*\big(A_i(t)\big) \\ [\text{by Corollary 2.9}] &= J^*\big(L_i(t)\big) = J_i^*(t), \end{aligned}$$

which can be easily generalized to a generic pair of nodes i, j connected by a directed path of length l_{ij} (such a path always exists for the hypothesis of strong connectivity):

$$J_j^*(t + l_{ij}) \geq J_i^*(t). \tag{A.1}$$

Moreover, we demonstrate that for any directed edge (i, j), it holds that

$$J_j^*(t+1) = J_i^*(t) \iff x_j^*(t+1) = x_i^*(t). \tag{A.2}$$

The reverse implication in (A.2) is straightforward, since the objective function is the same for both nodes. The direct implication is again trivial in the infeasible case, while for $J_j^*(t+1) = J_i^*(t) < \infty$ it can be proven as follows. For the uniqueness condition, adding a constraint c that is not satisfied at (or *violates*) $x_j^*(t+1)$ leads to an increase in the objective value, i.e., $J^*(L_j(t+1) \cup \{c\}) > J^*(L_j(t+1))$. Now, since $L_j(t+1) \supseteq A_i(t)$, and $J^*(L_j(t+1)) = J_j^*(t+1) = J_i^*(t) = J^*(A_i(t))$, by locality, if $J^*(L_j(t+1) \cup \{c\}) > J^*(L_j(t+1))$, then $J^*(A_i(t) \cup \{c\}) > J^*(A_i(t))$, which implies that also $x_i^*(t)$ is violated by c. Therefore, we concluded that every constraint that violates $x_j^*(t+1)$ also violates $x_i^*(t)$, and this may happen only if $x_j^*(t+1) = x_i^*(t)$. Again the correspondence between objective values and optimal solutions can be easily generalized to a generic pair of nodes i, j connected by a directed path of length l_{ij}:

$$J_j^*(t + l_{ij}) = J_i^*(t) \quad \Longleftrightarrow \quad x_j^*(t + l_{ij}) = x_i^*(t). \tag{A.3}$$

We now claim that the objective at one node cannot remain the same for $2\mathtt{diam}(\mathcal{G})+1$ iterations, unless the algorithm has converged. In the infeasible case the proof is trivial: according to the update rule of the active constraints consensus algorithm, if node i has detected an infeasible local problem, i.e., $J_i^*(t) = \infty$, it directly stops the execution of the algorithm since it is already sure of detaining the global solution. Let us instead consider the feasible case. We assume by contradiction that $J_i^*(t) = J_i^*(t + 2\mathtt{diam}(\mathcal{G})) < \infty$ and there exists a node j with at least a constraint that is not satisfied by $x_i^*(t) = x_i^*(t + 2\mathtt{diam}(\mathcal{G}))$. Let us consider a directed path of length l_{ij} from i to j: we already observed in (A.1) that $J_j^*(t + l_{ij}) \geq J_i^*(t)$. However, since there are constraints at node j that violate $x_i^*(t)$, equality cannot hold; see (A.3), and $J_j^*(t + l_{ij}) > J_i^*(t)$. By definition, the length l_{ij} of the path from i to j is bounded by the graph diameter and the local objective is nondecreasing; therefore $J_j^*(t + \mathtt{diam}(\mathcal{G})) > J_i^*(t)$. Now consider the path from j to i of length l_{ji}: according to (A.1) it must hold that $J_j^*(t+\mathtt{diam}(\mathcal{G})) \leq J_i^*(t+\mathtt{diam}(\mathcal{G})+l_{ji}) \leq J_i^*(t+2\mathtt{diam}(\mathcal{G}))$. Using the two inequalities found so far we obtain $J_i^*(t) < J_j^*(t + \mathtt{diam}(\mathcal{G})) \leq J_i^*(t + 2\mathtt{diam}(\mathcal{G}))$, which contradicts the assumption that the objective at node i remains constant for $2\mathtt{diam}(\mathcal{G}) + 1$ iterations. Therefore, before convergence the local objective $J_i^*(t)$ has to be strictly increasing every $2\mathtt{diam}(\mathcal{G}) + 1$ iterations. Moreover, the sequence $J_i^*(t)$ is upper bounded since, by monotonicity, for any $L \subseteq C$, $J^*(L) \leq J^*(C)$, and $J_i^*(t)$ can assume a finite number of values, i.e., $J^* \in \mathcal{J} \doteq \{J^*(L) \,:\, L \subseteq C\}$; therefore the sequence converges to a constant value, say $J_i^*(T)$, in finite time. We now demonstrate that after convergence, all nodes need to have the same local objective, i.e., $J_i^*(T) = \hat{J}$, for each $i \in \{1, \ldots, n\}$. For simplicity of notation, we drop the time index in the following discussion. Assume by contradiction that two nodes, say i and j, have different objective values, $J_i^* > J_j^*$. From the assumption of strong connectivity of the graph $\mathcal{G}$, there exists a directed path between i and j. Using relation (A.1) we obtain $J_i^* \leq J_j^*$, leading to a contradiction. Therefore, for any pair of nodes i and j, it must hold that $J_i^* = J_j^* = \hat{J}$, implying $J_i^* = \hat{J}$, for each $i \in \{1, \ldots, n\}$. With a similar reasoning, and using (A.3), we can also conclude that $J_i^* = \hat{J}$ for each $i \in \{1, \ldots, n\}$ implies $x_i^* = \hat{x}$ for each $i \in \{1, \ldots, n\}$. Now it remains to show that the local objectives $\hat{J}$ and the local solutions $\hat{x}$ actually coincide with $J^*(C)$ and $x^*(C)$. In the infeasible case this is again trivial: if the local objectives coincide with $\hat{J} = \infty$, by monotonicity the global problem cannot be other than infeasible, and then $J^*(C) = \hat{J} = \infty$ and $x^*(C) = \hat{x} = \mathtt{NaN}$. The feasible case can be proven as follows. If all nodes have the same local solution $\hat{x}$, it means that (i) $\hat{x}$ satisfies the local constraint set C_i, $i \in \{1, \ldots, n\}$, which implies that $\hat{x}$ is feasible for the global problem. Moreover, by monotonicity, $\hat{J} \leq J^*(C)$ (since $\hat{J}$ is the optimal value of a subproblem having constraint set $L \subseteq C$). Assume by contradiction that $\hat{J} < J^*(C)$, which implies that (ii) $\hat{J} = a^\top \hat{x} < a^\top x^*(C) = J^*(C)$; therefore $\hat{x}$ attains a smaller objective than $x^*(C)$ (see (ii)), and satisfies all constraints in C (see (i)), contradicting the optimality of $x^*(C)$. Therefore it must hold that $\hat{J} = J^*(C)$. With the same reasoning we used for proving (A.2), we also conclude that $\hat{x} = x^*(C)$.

To prove the third statement, we show that the set A_i contains all the constraints that are globally active for $P[C]$. If $J_i^* = J^*(C) = \infty$, the implication is trivial, since

$A_i = \mathtt{Ac}(C) = \emptyset$. In the feasible case the proof proceeds as follows. According to the second statement, we have $x_i^* = x^*(A_i) = x^*(C)$, $i \in \{1,\ldots,n\}$. By contradiction, let us suppose that there exists a globally active constraint c that is contained in the local constraint set C_i of a node i, but is not in the candidate set A_j of node j. Let us consider a directed path from i to j and relabel the nodes in this path from 1 to l. Starting from node 1 we observe that, since $x_1^* = x^*(C)$ and c is active for $P[C]$, $c \in A_1$. At each iteration of the active constraint consensus, node 2 in the path computes $A_2 = \mathtt{Ac}(A_2 \cup (\cup_{j\in\mathcal{N}_{\rm in}(2,t)} A_j) \cup C_2)$. Therefore, since $c \in A_1$ and $x_1^* = x_2^*$, it holds that $c \in A_2$. Iterating this reasoning along the path from i to j, we conclude that $c \in A_j$, leading to a contradiction.

To prove the fourth statement, we observe that if the local problem at node i is infeasible, then the node has only to transmit its local objective, $J_i(t)^* = \infty$, since the candidate set $A_i(t)$ is empty. If the local problem $P[L_i]$ is feasible, then the unique minimum condition ensures that the minimum is attained at a single point, say $x^*(L_i)$. If constraints are in general position, then no more than d constraints may be tight at $x^*(L_i)$, and hence at most d constraints are active. Therefore, in the feasible case, the number of constraints to be transmitted is upper bounded by d. ◻

A.3. Proof of Proposition 5.2. We start by recalling a basic property which is a direct consequence of the definition of the feasible set: for any set of constraints C_1 and C_2, it holds that

$$\mathtt{Sat}(C_1) \cap \mathtt{Sat}(C_2) = \mathtt{Sat}(C_1 \cup C_2). \tag{A.4}$$

To prove the first statement, we consider a generic node i. At time t node i receives the candidate sets from the incoming neighbors and computes $V_i(t{+}1) = \mathtt{vert}\big(L_i(t{+}1)\big) = \mathtt{vert}\big(V_i(t) \cup \big(\cup_{j\in\mathcal{N}_{\rm in}(i)} V_j(t)\big)\big)$. It follows that

$$\begin{aligned}\mathtt{Sat}(V_i(t+1)) &= \mathtt{Sat}\Big(\mathtt{vert}\big(V_i(t) \cup \big(\cup_{j\in\mathcal{N}_{\rm in}(i)} V_j(t)\big)\big)\Big)\\ \text{[by Lemma 5.1]} &= \mathtt{Sat}\big(V_i(t) \cup \big(\cup_{j\in\mathcal{N}_{in}(i,t)} V_j(t)\big)\big)\\ \text{[by equation (A.4)]} &= \mathtt{Sat}(V_i(t)) \cap \big(\cap_{j\in\mathcal{N}_{\rm in}(i)} \mathtt{Sat}(V_j(t))\big) \subseteq \mathtt{Sat}(V_i(t)).\end{aligned} \tag{A.5}$$

If $\mathtt{Sat}(V_i(t)) = \emptyset$ (infeasible local problem), then also $\mathtt{Sat}(V_i(t+1)) = \emptyset$, according to (A.5), then $J_i^*(t+1) = J_i^*(t) = \infty$, and the objective is nondecreasing. If $\mathtt{Sat}(V_i(t)) \neq \emptyset$ (feasible local problem), we can prove the statement as follows. Assume by contradiction that there exists $\bar{x} \in \mathtt{Sat}(V_i(t+1))$ such that $a^\top \bar{x} \doteq J^*(V_i(t+1)) < J^*(V_i(t))$. Equation (A.5) ensures that $\mathtt{Sat}(V_i(t+1)) \subseteq \mathtt{Sat}(V_i(t))$; therefore $\bar{x} \in \mathtt{Sat}(V_i(t))$ and there exists a point in the feasible set of problem $P[V_i(t)]$, whose value is smaller than $J^*(V_i(t))$. This contradicts the optimality of $J^*(V_i(t))$. Therefore, it must hold that $J^*(V_i(t+1)) \geq J^*(V_i(t))$.

To prove the second statement, we show that after $T \doteq \mathtt{diam}(\mathcal{G})$ iterations a generic node i satisfies $\mathtt{Sat}(V_i(T)) = \mathtt{Sat}(C)$. Consider a generic node j and a directed path from a node j to node i (this path does exist for the hypothesis of strong connectivity). We relabel the nodes on this path from 1 to l such that the last node is i. Node 1 initializes $V_1(0) = \mathtt{vert}(C_1)$, and then $\mathtt{Sat}(V_1(0)) = \mathtt{Sat}(C_1)$. At the first iteration, node 2 computes $V_2(1) = \mathtt{vert}\big(V_2(0) \cup \big(\cup_{j\in\mathcal{N}_{\rm in}(2)} V_j(0)\big)\big)$. Since node 1 is in $\mathcal{N}_{\rm in}(2)$, it follows from (A.5) that $\mathtt{Sat}(V_2(1)) \subseteq \mathtt{Sat}(V_1(0))$. Repeating the same reasoning along the path and for the original labeling, we can easily prove that $\mathtt{Sat}(V_i(l_{ij})) \subseteq \mathtt{Sat}(V_j(0)) = \mathtt{Sat}(C_j)$, where l_{ij} is the distance between i and j. Therefore, after a number of iterations equal to the distance between j and i, every feasible solution at node i satisfies the constraints of node j. Since the maximum distance between i and any other node is the diameter of the graph, in $T \doteq \mathtt{diam}(\mathcal{G})$ iterations, node i satisfies $\mathtt{Sat}(V_i(T)) \subseteq \mathtt{Sat}(C_j)$ for all j. Since this last property holds for all j, it also holds that $\mathtt{Sat}(V_i(T)) \subseteq \cap_{j\in\{1,\dots,n\}}\mathtt{Sat}(C_j) = \mathtt{Sat}(C)$. However, $V_i(T)$ is a subset of C, and it follows that $\mathtt{Sat}(V_i(T)) \supseteq \mathtt{Sat}(C)$. Thus, $\mathtt{Sat}(V_i(T)) = \mathtt{Sat}(C)$. Since the local problem $P[V_i(T)]$ and the global problem $P[C]$ have the same objective direction and the same feasible set, they attain the same (unique) solution, i.e., $x^*(V_i(T)) = x^*(C)$.

We now establish the third statement. We note that $V_i(T) = \mathtt{vert}(C)$ is a direct consequence of the update rule of the VCC algorithm. To prove the latter part of the statement, we assume by contradiction that $c \in C$ is a support constraint for $P[C]$ but $c \notin \mathtt{vert}(C)$. The relation $c \notin \mathtt{vert}(C)$ implies that $\mathtt{vert}(C) \subseteq C\backslash\{c\}$. It follows from monotonicity that (i) $J^*(\mathtt{vert}(C)) \leq J^*(C\backslash\{c\})$. According to Lemma 5.1 it also holds that (ii) $J^*(\mathtt{vert}(C)) = J^*(C)$. Combining (i) and (ii), we obtain $J^*(C) \leq J^*(C\backslash\{c\})$. By monotonicity, it cannot be $J^*(C) < J^*(C\backslash\{c\})$, and so $J^*(C) = J^*(C\backslash\{c\})$, but this contradicts the assumption that c is a support constraint. ∎

A.4. Proof of Proposition 5.3. The proof of the first and third statements follows similarly to the proof of the first and third statements in Proposition 4.1.

We now establish the second statement. Similarly to the VCC algorithm, we show that after $T \leq \sum_{k=0}^{\mathtt{diam}(\mathcal{G})-1} \lceil \frac{N_{\max}(d_{\max}+1)^k}{m} \rceil$ iterations a generic node i satisfies $\mathtt{Sat}(V_i(T)) = \mathtt{Sat}(C)$. Consider a generic pair of nodes i, j and a directed path of length l_{ji} from j to i (this path does exist for the hypothesis of strong connectivity). Relabel the nodes on this path from 1 to l such that the last node is i. We observe that, after the initialization, the local candidate set $V_1(0) = T_1(0) = \mathtt{vert}(C_1)$ has cardinality $|T_1(0)| \leq N_{\max}$. Since the transmission set is managed using an FIFO policy, after at most $\lceil \frac{N_{\max}}{m} \rceil$ communication rounds the node has transmitted all the constraints in $V_1(0)$ to node 2. Therefore, $\mathtt{Sat}\big(V_2(\lceil \frac{N_{\max}}{m} \rceil)\big) \subseteq \mathtt{Sat}(V_1(0)) = \mathtt{Sat}(C_1)$. Moreover, $\big|V_2(\lceil \frac{N_{\max}}{m} \rceil)\big| \leq \sum_{j\in\mathcal{N}_{\text{in}}(2)\cup\{2\}} N_j \leq N_{\max}(d_{\max}+1)$ (worst case, in which the incoming neighbors have to transmit all their local constraints, and all constraints are vertices of the convex hull). After at most $\lceil \frac{N_{\max}(d_{\max}+1)}{m} \rceil$ further iterations, node 2 has transmitted all constraints in $V_2\big(\lceil \frac{N_{\max}}{m} \rceil\big)$ to node 3. Therefore, $\mathtt{Sat}\big(V_3(\lceil \frac{N_{\max}}{m} \rceil + \lceil \frac{N_{\max}(d_{\max}+1)}{m} \rceil)\big) \subseteq \mathtt{Sat}(V_2(\lceil \frac{N_{\max}}{m} \rceil)) \subseteq \mathtt{Sat}(C_1)$. Also, $\Big|V_3\big(\lceil \frac{N_{\max}}{m} \rceil + \lceil \frac{N_{\max}(d_{\max}+1)}{m} \rceil\big)\Big| \leq \sum_{j\in\mathcal{N}_{\text{in}}(3)\cup\{3\}} \big|V_j(\lceil \frac{N_{\max}}{m} \rceil)\big| \leq N_{\max}(d_{\max}+1)^2$. Repeating the same reasoning along the directed path, for the original labeling, we obtain $\mathtt{Sat}\big(V_i\big(\sum_{k=0}^{l_{ji}-1} \lceil \frac{N_{\max}(d_{\max}+1)^k}{m} \rceil\big)\big) \subseteq \mathtt{Sat}(C_j)$. Therefore, every feasible solution at node i satisfies the constraints of node j at distance l_{ji} in a number of iterations no larger than $\sum_{k=0}^{l_{ji}-1} \lceil \frac{N_{\max}(d_{\max}+1)^k}{m} \rceil$. Since the maximum distance between i and any other node is the diameter of the graph, it follows that in $T \leq \sum_{k=0}^{\mathtt{diam}(\mathcal{G})-1} \lceil \frac{N_{\max}(d_{\max}+1)^k}{m} \rceil$ iterations node i satisfies $\mathtt{Sat}(V_i(T)) \subseteq \mathtt{Sat}(C_j)$ for all j. Since this property holds for all j, it also holds that $\mathtt{Sat}(V_i(T)) \subseteq \cap_{j\in\{1,\dots,n\}}\mathtt{Sat}(C_j) = \mathtt{Sat}(C)$. Since $V_i(T)$ is a subset of C, $\mathtt{Sat}(V_i(T)) \supseteq \mathtt{Sat}(C)$. Therefore, $\mathtt{Sat}(V_i(T)) = \mathtt{Sat}(C)$. Finally, T can be rewritten as

$$\sum_{k=0}^{\mathtt{diam}(\mathcal{G})-1} \left\lceil \frac{N_{\max}(d_{\max}+1)^k}{m} \right\rceil \leq \left\lceil \frac{N_{\max}}{m} \right\rceil \sum_{i=0}^{\mathtt{diam}(\mathcal{G})-1} (d_{\max}+1)^k$$
$$= \left\lceil \frac{N_{\max}}{m} \right\rceil \frac{1-(d_{\max}+1)^{\mathtt{diam}(\mathcal{G})}}{1-(d_{\max}+1)} = \left\lceil \frac{N_{\max}}{m} \right\rceil \frac{(d_{\max}+1)^{\mathtt{diam}(\mathcal{G})}-1}{d_{\max}},$$

which coincides with the bound in the second statement. Since the local problem $P[V_i(T)]$ and the global problem $P[C]$ have the same objective direction and the same feasible set, they attain the same (unique) solution, i.e., $x^*(V_i(T)) = x^*(C)$. ∎

Acknowledgment. The third author thanks Giuseppe Notarstefano for insightful discussions about abstract programming.

REFERENCES

[1] L. Carlone, V. Srivastava, F. Bullo, and G. C. Calafiore, *A distributed algorithm for random convex programming*, in Proceedings of the 5th International Conference on Network Games, Control and Optimization (NetGCooP), Paris, France, 2011, pp. 1–7.

[2] G. C. Calafiore, *Random convex programs*, SIAM J. Optim., 20 (2010), pp. 3427–3464.

[3] G. C. Calafiore and M. C. Campi, *The scenario approach to robust control design*, IEEE Trans. Automat. Control, 51 (2006), pp. 742–753.

[4] G. Notarstefano and F. Bullo, *Network abstract linear programming with application to minimum-time formation control*, in Proceedings of the 46th IEEE Conference on Decision and Control, New Orleans, LA, 2007, pp. 927–932.

[5] G. Notarstefano and F. Bullo, *Distributed abstract optimization via constraints consensus: Theory and applications*, IEEE Trans. Automat. Control, 56 (2011), pp. 2247–2261.

[6] A. Bental, L. El-Ghaoui, and A. Nemirovski, *Robust Optimization*, Princeton University Press, Princeton, NJ, 2009.

[7] G. C. Calafiore and M. C. Campi, *Uncertain convex programs: Randomized solutions and confidence levels*, Math. Programming Ser. A, 102 (2005), pp. 25–46.

[8] G. C. Calafiore and M. C. Campi, *Notes on the scenario design approach*, IEEE Trans. Automat. Control, 54 (2009), pp. 382–385.

[9] A. Nemirovski and A. Shapiro, *Scenario approximations of chance constraints*, in Probabilistic and Randomized Methods for Design under Uncertainty, G. C. Calafiore and F. Dabbene, eds., Springer, New York, 2006, pp. 3–47.

[10] L. S. Lasdon, *Optimization Theory for Large Systems*, The Macmillan Company, New York, 1970.

[11] D. A. Wismer, ed., *Optimization Methods for Large Scale Systems: With Applications*, McGraw–Hill, New York, 1971.

[12] J. N. Tsitsiklis, *Problems in Decentralized Decision Making and Computation*. Ph.D. thesis, Massachusetts Institute of Technology, Boston, MA, 1984. Available online at http://web.mit.edu/jnt/www/Papers/PhD-84-jnt.pdf.

[13] A. Nedić and A. Ozdaglar, *Distributed subgradient methods for multi-agent optimization*, IEEE Trans. Automat. Control, 54 (2009), pp. 48–61.

[14] M. Zhu and S. Martínez, *On distributed convex optimization under inequality and equality constraints*, IEEE Trans. Automat. Control, 57 (2012), pp. 151–164.

[15] E. Wei, A. Ozdaglar, and A. Jadbabaie, *A distributed Newton method for network utility maximization*—I: *Algorithm*, IEEE Trans. Automat. Control, 58 (2013), pp. 2162–2175.

[16] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Found. Trends Mach. Learn., 3 (2010), pp. 1–124.

[17] M. Bürger, G. Notarstefano, F. Bullo, and F. Allgöwer, *A distributed simplex algorithm for degenerate linear programs and multi-agent assignments*, Automatica J. IFAC, 48 (2012), pp. 2298–2304.

[18] M. Bürger, G. Notarstefano, and F. Allgöwer, *Locally constrained decision making via two-stage distributed simplex*, in Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference, Orlando, FL, 2011, pp. 5911–5916.

[19] A. B. Amenta, *Helly Theorems and Generalized Linear Programming*, Ph.D. thesis, Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA, 1993.

[20] B. Chazelle, *An optimal convex hull algorithm in any fixed dimension*, Discrete Comput. Geom., 10 (1993), pp. 377–409.

[21] R. Cowan, *Recurrence relationships for the mean number of faces and vertices for random convex hulls*, Discrete Comput. Geom., 43 (2010), pp. 209–220.

[22] R. A. Dwyer, *On the convex hull of random points in a polytope*, J. Appl. Probab., 25 (1988), pp. 688–699.

[23] C. Durieu, E. Walter, and B. Polyak, *Multi-input multi-output ellipsoidal state bounding*, J. Optim. Theory Appl., 111 (2001), pp. 273–303.

[24] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, Stud. Appl. Math. 15, SIAM, Philadelphia, 1994.

[25] T. Eren, D. K. Goldenberg, W. Whiteley, Y. R. Yang, A. S. Morse, B. D. O. Anderson, and P. N. Belhumeur, *Rigidity, computation, and randomization in network localization*, in Proceedings of the 23rd IEEE Annual Joint Conference of the IEEE Computer and Communications Societies, Hong Kong, China, 2004, pp. 2673–2684.

[26] M. Grant and S. Boyd, *CVX: MATLAB Software for Disciplined Convex Programming, Version* 2.0, http://cvxr.com/cvx (2013).

[27] A. Ben-Tal and A. Nemirovski, *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*, MOS-SIAM Ser. Optim. 2, SIAM, Philadelphia, 2001.

[28] B. E. Boser, I. M. Guyon, and V. N. Vapnik, *A training algorithm for optimal margin classifiers*, in Proceedings of the ACM Workshop on Computational Learning Theory, Pittsburgh, PA, 1992, pp. 144–152.

[29] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.

[30] H. Drucker, V. Vapnik, and D. Wu, *Support vector machines for spam categorization*, IEEE Trans. Neural Networks, 10 (2006), pp. 1048–1054.

[31] G. C. Calafiore and L. Fagiano, *Robust model predictive control via scenario optimization*, IEEE Trans. Automat. Control, 58 (2013), pp. 219–224.

[32] G. C. Calafiore and L. Fagiano, *Stochastic model predictive control of LPV systems via scenario optimization*, Automatica J. IFAC, 49 (2013), pp. 1861–1866.