A Distributed Algorithm for Random Convex Programming

Luca Carlone, Vaibhav Srivastava, Francesco Bullo, and Giuseppe C. Calafiore

***Abstract*— We study a distributed approach for solving random convex programs (RCP) for the case in which problem constraints are distributed among nodes in a processor network. We devise a distributed algorithm that allows network nodes to reach consensus on problem solution by exchanging a local set of constraints at each iteration. We prove that the algorithm assures finite-time convergence to problem solution and we provide explicit bounds on the maximum number of constraints to be exchanged among nodes at each communication round. Numerical experiments confirm the theoretical derivation and show that a parallel implementation of the proposed approach speeds-up the solution of the RCP with respect to centralized computation.**

I. Introduction

Optimization algorithms constitute the core of several problems in estimation, control and machine learning. In order to guarantee some sort of robustness of the optimal solution, the uncertainty in the input data must be taken into account. Particularly, if the constraints defining an optimization problem are functions of a random parameter, then the *robustness* of the optimal solution is measured by the capacity of the solution to preserve optimality under several realizations of the uncertain parameter. There are several approaches to ascertain robustness of the optimal solution. The *robust convex optimization* [1] enforces the optimality of the solution to the worst possible realization of the uncertainty. The *chance-constrained optimization* [2] provides an optimal solution with guaranteed bounds on the probability of constraints violation. In this context, *random convex programming* [3] approach enforces the optimality on a realization of the uncertainty (*scenario*) and provides probabilistic assessments on the robustness of the solution towards further *unseen* random constraints. The power of this approach to uncertain convex optimization lies in its *computational tractability* (they can be formulated in terms of finite-dimensional convex optimization problems) and in the possibility of trading-off between *robustness* (i.e., bounds on violation probability) and *performance* (i.e., value of the optimal objective). Although computing the solution for a given scenario reduces to solve a convex program, an RCP may have a large number of constraints and entrusting the computation to a single processor can be undesirable in terms of computational effort. Moreover, the constraints defining the scenario may be not known to a central unit, since each constraint can be function of local information acquired by different nodes in a network. For such reasons, in this context, we are interested in investigating a distributed approach to random convex programs.

Recently, there has been significant interest in distributed optimization. Widespread approaches for distributed optimization are based on *dual decomposition* [4], [5], [6], [7]. In dual decomposition-based approaches, the nodes communicate the estimate of the local algorithm and asymptotically achieve the global solution of the problem. An alternative approach to distributed optimization has been proposed in [8], [9], where the nodes exchange a small set of constraints at each iteration, and converge to a consensus set of constraints that determines the global solution of the optimization problem in finite time.

In this paper, we consider the case in which a network of processors is in charge of solving a random convex program. Each node knows a subset of constraints of the RCP and the network is required to compute the global solution of the RCP (i.e., the solution obtained by considering all the constraints) in a distributed fashion. First, we investigate some basic properties of the constraints sets in convex programs. Then, we exploit these properties to devise a distributed algorithm, namely *active constraints consensus* (ACC), that allows to solve the RCP in a finite number of iterations. The proposed algorithm is inspired by the recent work [8], and extends it to convex programming. The ACC algorithm can be also used for parallel computation of the solution of the RCP. For large scale problems, we numerically demonstrate that such parallel computation significantly improves the computation time over the centralized setup. We elucidate on the proposed theory showing an application to distributed classification.

The rest of this paper is organized as follows. Preliminaries on convex optimization are discussed in Section II. Certain properties of the constraints in convex programs are established in Section III. The ACC algorithm is presented in Section IV. We elucidate on the developed theory with some examples in Section V. Conclusions are presented in Section VI.

This work was funded by MACP4LOG grant (RU/02/26) from Piemonte Region, Italy, by PRIN grant n. 20087W5P2K from the Italian Ministry of University and Research, by NSF Award CPS 1035917 and by ARO Award W911NF-11-1-0092. The third author thanks Giuseppe Notarstefano for insightful discussions about abstract programming.

L. Carlone is with the Laboratorio di Meccatronica, Politecnico di Torino, Italy. luca.carlone@polito.it

V. Srivastava and F. Bullo are with the Center for Control, Dynamical Systems, and Computation, University of California Santa Barbara, United States of America. {vaibhav, bullo}@engineering.ucsb.edu

G. C. Calafiore is with the Dipartimento di Automatica e Informatica, Politecnico di Torino, Italy. giuseppe.calafiore@polito.it

II. Preliminaries

Consider a generic d-dimensional convex program

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

where $x \in X$ is the *optimization variable*, $X \subset \mathbb{R}^d$ is a closed and convex set, a is the *objective direction*, $f_j : \mathbb{R}^d \to \mathbb{R}$, $j \in C$ are convex *constraints*, and $C \subseteq \mathbb{N}$ is the finite set of constraints. We denote the solution of the problem $P[C]$ by $x^*(C)$ and the corresponding optimal value by $J^*(C)$. We now introduce some definitions:

Definition 1 (Support Constraints): The support constraints set $\mathtt{sc}(C) \subseteq C$ of problem $P[C]$ is the set of constraint $c \in C$ such that $J^*(C \backslash \{c\}) < J^*(C)$. The cardinality of the set of support constraints is upper bounded by $d+1$; moreover, the upper bound reduces to d, if the problem is feasible [3]. □

Definition 2 (Essential Constraint Set): An essential constraint set $\mathtt{es}(C) \subseteq C$ of problem $P[C]$ is a minimal subset such that $J^*(\mathtt{es}(C)) = J^*(C)$. If the essential constraint set $\mathtt{es}(C_i)$ is unique for any $C_i \subseteq C$, the optimization problem $P[C]$ is called *nondegenerate*. □

Definition 3 (Active Constraints): The active constraints set $\mathtt{ac}(C) \subseteq C$ of problem $P[C]$ is the set of constraints that are tight at the optimal solution $x^*(C)$, that is, $\mathtt{ac}(C) = \{j \in C : f_j(x^*(C)) = 0\}$. □

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 the problem $P[C_i]$ admits a unique solution, for any $C_i \subseteq C$. A convex program that does not satisfy unique minimum condition can be easily transformed to an equivalent problem that satisfies the unique minimum condition, by applying a suitable *tie-breaking rule* (e.g., choosing the lexicographic smallest solution $x^*(C')$, $\forall C' \subseteq C$ within the set of optimal solutions) [3]. We provide an example of a tie-breaking rule in Appendix I of the paper. Accordingly and without loss of generality, in the following we consider convex programs satisfying the unique minimum condition. We conclude this section with a last definition:

Definition 4 (General Position): A set of constraints in a feasible d-dimensional convex program is said to be in *general position* if no more than d constraints intersect at any point in the domain X. In the case of possibly unfeasible problems no more than $d+1$ constraints intersect at any point of the domain. □

III. Properties of the Constraints Sets

We now state some properties of convex programs pertinent to the discussion in this paper. We start by introducing some notations. Let the number of different essential sets in C be b and $\mathtt{es}_i(C)$ be the ith essential set. For the optimization problem $P[C]$, we define the *combinatorial dimension* d_{comb} as:

$$d_{\text{comb}} = \begin{cases} d, & \text{if } P[C] \text{ is feasible} \\ d+1, & \text{otherwise.} \end{cases}$$

We now state some properties of the problem $P[C]$:

Proposition 1 (Monotonicity & Locality, [3]): 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

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

Proposition 2 (Properties of the constraints): The following statements hold for the constraints sets in optimization problem $P[C]$:

i) The set of support constraints is the intersection of all the essential constraints sets, that is, $\mathtt{sc}(C) = \cap_{i=1}^{b} \mathtt{es}_i(C)$
ii) The set of active constraints contains the union of all the essential sets, that is, $\mathtt{ac}(C) \supseteq \cup_{i=1}^{b} \mathtt{es}_i(C)$.
iii) The cardinality of each essential constraints set is at most d_{comb}.

Proof. We start by establishing the first statement. Assume that $c \in \mathtt{sc}(C)$, that is, $J^*(C \backslash \{c\}) < J^*(C)$. Suppose now that $c \notin \cap_{i=1}^{b} \mathtt{es}_i(C)$, that is, $c \notin \mathtt{es}_i(C)$, for some $i \in \{1, \ldots, b\}$. Therefore,

$$J^*(\mathtt{es}_i(C) \backslash \{c\}) = J^*(\mathtt{es}_i(C)). \tag{2}$$

By the definition of an essential set, $J^*(C) = J^*(\mathtt{es}_i(C))$. It follows from the assumption and equation (2) that

$$J^*(C \backslash \{c\}) < J^*(C) = J^*(\mathtt{es}_i(C) \backslash \{c\}),$$

which contradicts monotonicity. Therefore, $c \in \mathtt{sc}(C)$ implies $c \in \cap_{i=1}^{b} \mathtt{es}_i(C)$, that is, $\mathtt{sc}(C) \subseteq \cap_{i=1}^{b} \mathtt{es}_i(C)$.

We now claim that $\mathtt{sc}(C) \supseteq \cap_{i=1}^{b} \mathtt{es}_i(C)$. Let us assume that $J^*(C \backslash \{c\}) = J^*(C)$, that is, $c \notin \mathtt{sc}(C)$. It follows that there exists $\mathtt{es}(C \backslash \{c\})$ such that $J^*(C) = J^*(\mathtt{es}(C \backslash \{c\}))$. Therefore, $\mathtt{es}(C \backslash \{c\})$ is also an essential set for C and $c \notin \cap_{i=1}^{b} \mathtt{es}_i(C)$. This concludes the proof of the first statement.

We now establish the second statement. From the minimality of the essential set, it follows that each constraint in $\mathtt{es}_i(C)$ is a support constraint for the problem $P[\mathtt{es}_i(C)]$, and they need to be active for the problem $P[\mathtt{es}_i(C)]$, see Appendix II. As a consequence, if $x^*(\mathtt{es}_i(C))$ is the optimal solution for $P[\mathtt{es}_i(C)]$, and $f_j(x_i^*) = 0$, for each $j \in \mathtt{es}_i(C)$. 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 $i \in \{1, \ldots, b\}$. Therefore, $f_j(x^*) = 0$, for each $j \in \mathtt{es}_i(C), i \in \{1, \ldots, b\}$, and $\mathtt{ac}(C) \supseteq \cup_{i=1}^{b} \mathtt{es}_i(C)$.

To prove the last statement, we note from [3] that the maximum cardinality of the support set of $P[C]$ is d_{comb}.

Assume that $P[C]$ is feasible. Now consider by absurd that $P[C]$ has at least one essential set, say $\mathtt{es}_i(C)$ with cardinality $\hat{d} > d$. Notice that $\mathtt{es}_i(C)$ is the unique essential set for $P[\mathtt{es}_i(C)]$. It follows from the first statement that $\mathtt{es}_i(C) = \mathtt{sc}(\mathtt{es}_i(C))$. This means that the support constraint set $\mathtt{sc}(\mathtt{es}_i(C))$ has cardinality $\bar{d}$, which is a contradiction. Similar argument applies in the case when $P[C]$ is infeasible. This concludes the proof. □

Corollary 3 (Constraints set in nondegenerate programs): For nondegenerate convex programs, $\mathtt{es}(C_i) = \mathtt{sc}(C_i)$, for any $C_i \subseteq C$.

Proof. As a consequence of nondegeneracy, the essential set of any $C_i \subseteq C$ is unique and the statement follows immediately from Proposition 2. □

IV. Distributed Random Convex Programs

A. Problem Statement

Given a function $f : (\mathbb{R}^d \times \Delta) \to \mathbb{R}$, where $\Delta \subseteq \mathbb{R}^\ell$ is the sample space from which a vector of random parameters δ is drawn, the associated *random constraint* is defined by $f(x, \delta) \leq 0$. A random constraint is said to be *convex* if $x \mapsto f(x, \delta)$ is convex for all $\delta \in \Delta$.

Consider a networked system with n interacting nodes (e.g., *processors*, *sensors*). We model node communication by a digraph G with vertex set $\{1, \dots, n\}$ and edge set $\mathcal{E}$. Let $\mathcal{N}_{\text{in}}(i)$ and $\mathcal{N}_{\text{out}}(i)$ be the set of incoming and outgoing neighbors of node i, respectively. Denote the *diameter* of the graph G by $\mathtt{diam}(G)$. Suppose that node i has a local set of random constraints, in the form $f_j(x) \doteq f(x, \delta^{(j)})$, $j \in C_i$, where $\delta^{(j)}$ are realizations of some random variable and C_i is the set of indexes specifying the local constraint set, with $N_i \doteq |C_i|$. Define $C = \cup_{i=1}^n C_i$, $N \doteq |C|$, and assume that $\delta^{(j)} \in \Delta$, $j = 1, \dots, N$, are *independent and identically distributed*. Finally, denote with $\omega \doteq (\delta^{(1)}, \dots, \delta^{(N)}) \in \Delta^N$, with $\Delta^N = \Delta \times \Delta \cdots \Delta$ (N times).

We define the *distributed random convex program* (DRCP) over graph G as:

$$P[C(\omega)] \doteq P[\omega] : \quad \min_{x \in X} \ a^\top x \quad \text{subject to:} \quad f_j(x) \leq 0, \quad j \in \cup_{i=i}^n C_i \tag{3}$$

where node i knows set of random constraints C_i, it can receive information from nodes in the set $\mathcal{N}_{\text{in}}(i)$ and can transmit information to nodes in the set $\mathcal{N}_{\text{out}}(i)$. Since each node has only a partial knowledge of problem constraints, it needs to exploit local computation and inter-nodal communication to compute the global solution of $P[\omega]$. We refer to the solution of $P(\omega)$ obtained by considering only the local constraints at a given node by the *local solution* and the associated value of the objective function by the *local optimal value.* We denote the local solution and the local optimal value at node i by x_i^* and J_i^*, respectively. We refer to the solution and the optimal value of $P[\omega]$ obtained by considering all the constraints by the *global solution* and the *global optimal value*, respectively.

We conclude this section with two definitions:

Definition 5 (Helly's dimension [3]): Let $C(\omega)$ be the set of N random constraints associated with realization ω. The *Helly's dimension* of a random convex program $P[\omega]$ is the least integer ζ such that

$$\operatorname*{ess\,sup}_{\omega \in \Delta^N} |\mathtt{sc}(C(\omega))| \leq \zeta$$

holds for any finite $N \geq 1$. □

Definition 6 (Violation probability [3]): For a random convex program $P[\omega]$, the *violation probability* $V^*(\omega)$ is defined as

$$V^*(\omega) \doteq \mathbb{P}\{\delta \in \Delta : \ J^*(\omega \cup \{\delta\}) > J^*(\omega)\}.$$

where $J^*(\omega \cup \{\delta\})$ is the optimal value of $P[\omega]$ subject to an additional random constraint associated with the realization δ of the random parameter. □

B. Distributed Computation of the Scenario Solution

We now study the distributed computation of the scenario solution of random convex program $P[\omega]$ under the following assumptions:

Assumption 1: The digraph G is *strongly connected*, that is, the digraph G contains a directed path from each vertex to another. □

Assumption 2: The constraints in the random convex program $P[\omega]$ are in general position almost surely. □

The global optimal objective value $J^*(\omega)$ and the solution $x^* = x^*(\omega)$ are random variables, whose value depend on the realization of the uncertain parameter ω. Moreover, for each realization of ω, random convex program $P[\omega]$ is a convex optimization problem. In the following we present a distributed algorithm that computes the solution to such convex optimization problem. We assume that a generic node i at time t has knowledge of the local set C_i and can store a small *candidate set* that we call $A_i(t)$. Each node initializes the candidate set to $A_i(0) = \mathtt{ac}(C_i)$, by solving the local convex program $P[C_i]$. Then, at each iteration t of the algorithm, node i receives the candidate sets from the incoming neighbors, i.e., $A_j(t)$, $j \in \mathcal{N}_{\text{in}}(i)$, and updates its candidate set with the following rule: $A_i(t+1) = \mathtt{ac}\big(A_i(t) \cup \big(\cup_{j \in \mathcal{N}_{\text{in}}(i)} A_j(t)\big) \cup C_i\big)$. The algorithm, named *active constraints consensus*, is summarized in Algorithm 1. We now state some important properties of the algorithm in the following proposition.

Algorithm 1 Active Constraints Consensus (ACC)

1: *Input:* c, G, and $C_i, \forall i \in \{1, \dots, n\}$
2: *Output:* Scenario solution $x^*(C)$, active set $\mathtt{ac}(C)$
% Initialization
3: Set $A_i(0) = \mathtt{ac}(C_i)$, for each $i \in \{1, \dots, n\}$; $t = 1$;
% Update
4: **if** $A_i(t-l) = A_i(t), \forall l \in \{1, \dots, 2\mathtt{diam}(G) + 1\}$
set $\mathtt{ac}(C) = A_i(t)$, $x^*(C) = x^*(A_i(t))$; exit;
5: **else**
$A_i(t+1) = \mathtt{ac}\big(A_i(t) \cup \big(\cup_{j \in \mathcal{N}_{\text{in}}(i)} A_i(t)\big) \cup C_i\big)$;
$t = t + 1$; **go** to 4:;

Proposition 4: Consider a processor network defined over a graph G and the distributed random convex program $P[\omega]$. The following statements hold under Assumptions 1 and 2:

i) At each iteration of the Algorithm 1 each node transmits at most d_{comb} constraints to the outgoing neighbors;
ii) The local optimal objective $J^*(A_i(t))$ is monotonically non-decreasing in the iterations t and converges in a finite number of iterations, say T, to the global optimal value $J^*(C)$;
iii) At iteration T, the local solution at a generic node i is the same as the global solution;
iv) For each node i the local candidate set $A_i(T)$ at time T coincides with the global active set $\texttt{ac}(C)$;
v) If the distributed random convex program $P[\omega]$ is almost surely nondegenerate, then the local probability of violation $V^*(A_i(T))$ is equal to $V^*(\omega)$, the violation probability of the global solution obtained in [3].

Proof. From Assumption 2 it follows that no point in X lies on the boundary of more than d_{comb} constraints. For the unique minimum condition, the minimum is attained at a single point, therefore, no more than d_{comb} constraints can be active for any subproblem $P[H]$, with $H \subseteq C$, and the number of constraints to be transmitted is upper-bounded by d_{comb}.

To prove the second statement we first observe that (a) $J^*(\texttt{ac}(H)) = J^*(\texttt{es}(H)) = J^*(H)$, for any $H \in C$. The equality can be proved as follows. $J^*(\texttt{ac}(H)) \geq J^*(\texttt{es}(H))$ is a consequence of monotonicity and Lemma 2 ($\texttt{ac}(H) \supseteq \texttt{es}(H)$). Assume now the contradiction that $J^*(\texttt{ac}(H)) > J^*(\texttt{es}(H))$; by monotonicity and by definition of essential set it follows $J^*(H) \geq J^*(\texttt{ac}(H)) > J^*(\texttt{es}(H)) = J^*(H)$, leading to contradiction. We can now use this basic property for analyzing the iterations of the active constraints consensus:

$$J^*(A_i(t+1)) = J^*(\texttt{ac}(A_i(t) \cup (\cup_{j \in \mathcal{N}_{\text{in}}(i)} A_j(t)) \cup C_i)) =$$
$$\overset{\text{fact (a)}}{=} J^*(A_i(t) \cup (\cup_{j \in \mathcal{N}_{\text{in}}(i)} A_j(t)) \cup C_i) \overset{\text{monotonicity}}{\geq} J^*(A_i(t)).$$

Thus, the sequence of objective values is non-decreasing in t. Moreover, the sequence is upper bounded, since for any $H \subseteq C$, $J^*(H) \leq J^*(C)$. Finally, we notice that $J^*(A_i(t))$ can assume a finite number of values, i.e., $J^* \in \mathcal{J} := \{J^*(H) \mid H \subseteq C\}$, hence the sequence has to converge to a constant value, say J_i^* in finite time. It now remains to demonstrate that after all the nodes in the network have reached convergence, the local objective $J_i^* \doteq J^*(A_i(T))$ at a generic node i is equal to the global objective $J^*(C)$. For the hypothesis of convergence, in the following development we drop the time indices. We first demonstrate that after convergence all the nodes need to have the same local objective, i.e., $J_i^* = \hat{J}$, for each $i \in \{1, \ldots, n\}$. Assume by absurd that two nodes, say i and j, have different objective values, $J_i^* > J_j^*$. From the assumption of strongly connectivity of the graph G, there exist a directed path between i and j. Because of the update law of algorithm, along this path, monotonicity holds, i.e., for any directed edge (i_1, i_2), that belongs to the path from i to j, it holds:

$$J_{i_2}^* = J^*(A_{i_2}) = J^*(\texttt{ac}(A_{i_2} \cup (\cup_{j \in \mathcal{N}_{\text{in}}(i)} A_{i_1}) \cup C_{i_2}))$$
$$\overset{\text{fact (a)}}{=} J^*(A_{i_2} \cup (\cup_{j \in \mathcal{N}_{\text{in}}(i)} A_j) \cup C_{i_2})$$
$$\overset{\text{monotonicity}}{\geq} J^*(A_{i_2} \cup A_{i_1} \cup C_{i_2}) \overset{\text{monotonicity}}{\geq} J^*(A_{i_1}).$$

Iterating this reasoning along the path we arrive to the contradiction $J_i^* \leq J_j^*$. Therefore, for any pair of nodes i and j, it must hold $J_i^* = J_j^* = \hat{J}$, implying $J_i^* = \hat{J}$, for each $i \in \{1, \ldots, n\}$.

Then, we want to show that the local objective $\hat{J}$ actually corresponds to $J^*(C)$. Let us consider a directed path connecting all nodes and ending at a generic node i. Now re-label the nodes in this path from 1 to n and start by considering the last node. For the hypothesis that the node reached convergence and for the presence of the edge $(n-1, n)$ it holds (b) $J_n^* = J^*(A_n) = J^*(A_n \cup A_{n-1} \cup C_n)$. Similarly for node $n-1$ it holds (c) $J_{n-1}^* = J^*(A_{n-1}) = J^*(A_{n-1} \cup A_{n-2} \cup C_{n-1})$. But we demonstrated that $J_n^* = J_{n-1}^*$, hence from (b) it follows (d) $J^*(A_n \cup A_{n-1} \cup C_n) = J^*(A_{n-1})$. Since from (c), C_{n-1} does not contain violators of A_{n-1}, applying locality to (d) if holds $J_n^* = J^*(A_n) = J^*(A_n \cup A_{n-1} \cup C_n) = J^*(A_n \cup A_{n-1} \cup C_n \cup C_{n-1})$. Repeating the same considerations for node $n-1$ and $n-2$, we conclude that C_{n-2} does not contain violators of A_{n-1}, hence by locality $J_n^* = J^*(A_n \cup A_{n-1} \cup C_n) = J^*(A_n \cup A_{n-1} \cup C_n \cup C_{n-1} \cup C_{n-2})$. Iterating such reasoning along the directed path spanning all nodes, we conclude $J_n^* = J^*(A_n \cup A_{n-1} \cup_{i=1}^n C_i)$; finally we observe that by assumption $\cup_{i=1}^n C_i = C$ and $A_n, A_{n-1} \subseteq C$, therefore $J_n^* = J_i^* = J^*(C)$, and this proves point (ii) of the proposition.

Point (iii) claims that after convergence is attained the local solution $x^*(A_i)$ coincides with the centralized solution of the RCP. This is a consequence of the fact that $J^*(A_i) = J^*(C)$: for the unique minimum condition the optimal objective value is attained at a single point, moreover by locality every constraints that violates A_i has to violate also C and this can happen only if $x^*(A_i) = x^*(C)$.

For proving point (iv) we have to show that the set A_i contains all the constraints that are globally active for $P[C]$. According to the claim (iii) we have that $x^* = x_i^* = x^*(A_i)$, $i \in \{1, \ldots, n\}$ and this solution coincides with $x^*(C)$. By contradiction let us suppose that there exists a globally active constraint c_k 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 re-label the nodes in this path from 1 to l. Starting from node 1 we observe that, since $x_1^* = x^*(C)$ and c_k is active for $P[C]$, then $c_k \in A_1$. According to the update rule of the active constraint consensus, node 2 in the path, computes $A_2 = \texttt{ac}(A_2 \cup (\cup_{j \in \mathcal{N}_{\text{in}}(2,t)} A_j) \cup C_2)$. Therefore, if $c_k \in A_1$ and $x_1^* = x_2^*$, then $c_k \in A_2$. Iterating this reasoning along the path from i to j we conclude that $c_k \in A_j$ leading to contradiction.

The proof of claim (v) is a straightforward consequence of the results on random convex programs, see [3]. □

It is worth noticing that the proof of claim (ii) proceeds along the same lines of the proof of Theorem IV.4 in [8]. Moreover, it is easy to prove that, if the candidate set at one node does not change for $2\texttt{diam}(G)+1$ iterations, then the local solution at the node has converged to the global solution (the demonstration is similar to the proof of claim (ii) of Proposition 4).

Remark 1: The active constraints consensus algorithm can be used for the distributed computation of the solution of any convex program. It is particularly suited for random convex programs because they have a large number of constraints which are in general position almost surely. □

Remark 2: As a by-product of this work we notice that the approach proposed in [8] can be applied as a distributed algorithm for convex programming (according to Proposition 1 convex programs are *abstract optimization problems*). However, [8] would require the nodes to compute an essential set of the local set of constraints at each iteration, and such computation can be expensive in practice. On the other hand, the ACC algorithm only requires the computation of the active constraints set, which can be easily obtained by solving the local convex problem. Moreover, when the local solution $x_i^* \doteq x^*(A_i(t))$ is not violated by the incoming neighbors constraints, $\cup_{j\in\mathcal{N}_{\text{in}}(i)} A_i(t)$, the update of ACC algorithm only requires to check if for some $c \in \cup_{j\in\mathcal{N}_{\text{in}}(i)} A_i(t)$, it holds $f_c(x_i^*) = 0$. □

V. Numerical Example

We now briefly describe an application of the proposed approach to *robust linear classification*.

A. Robust Linear Classification

Let us consider a set of features $\{a_k \in \mathbb{R}^p \mid k \in \{1,\dots,m\}\}$ and the corresponding class labels $b_k \in \{-1,+1\}$. The linear classification problem determines a hyperplane that separates the features with different labels. In general, the features with different labels are not linearly separable, and a convex loss function penalizing *misclassifications* is minimized. Suppose that each node in a processor network measures a subset of the features; each feature is measured with uncertainty and the associated distribution is unknown. Let Δ be the sample space and $\delta \in \Delta$ represent a particular realization of the uncertainty. The robust linear classification problem minimizes the loss due to *misclassification* in one of the following ways: it minimizes the worst loss due to mis-classifications

$$\begin{aligned} &\min_{\theta,\lambda,\rho} \quad g(\theta)+\lambda \qquad \text{subject to :} \\ &\max_{\delta\in\Delta} \max_{k\in\{1,\dots,m\}} h(b_k(a_k^T(\delta)\theta+\rho)) - \lambda \le 0 \\ &\theta\in\mathbb{R}^p, \rho\in\mathbb{R}, \lambda\in\mathbb{R}_{\ge 0}, \end{aligned} \tag{4}$$

or it minimizes the sum of the losses due to misclassifications

$$\begin{aligned} &\min_{\theta,\lambda_k,\rho} \quad g(\theta)+\sum_{k=1}^m \lambda_k \qquad \text{subject to :} \\ &\max_{\delta\in\Delta} \max_{k\in\{1,\dots,m\}} h(b_k(a_k^T(\delta)\theta+\rho)) - \lambda_k \le 0 \\ &\theta\in\mathbb{R}^p, \rho\in\mathbb{R}, \lambda_k\in\mathbb{R}_{\ge 0}, \end{aligned} \tag{5}$$

where $g:\mathbb{R}^p\to\mathbb{R}_{\ge 0}$ is a convex *regularization function*, $h:\mathbb{R}\to\mathbb{R}_{\ge 0}$ is a convex *loss function*, $\lambda,\lambda_k\in\mathbb{R}_{\ge 0}$ are some parameters.

In these formulations, the regularization function imparts special structure to the parameter θ, while the loss function penalizes for misclassification of some feature into wrong label. Problems (4) and (5) correspond to the robust formulations of the classification problem; we leave to the reader the modifications that allow to retrieve the RCP counterpart.

Example 1: We study the following robust linear classification problem:

$$\begin{aligned} &\min_{\theta,\lambda,\rho} \quad \lambda \qquad \text{subject to :} \\ &x_i^\top\theta+\rho-\lambda\le 0, \quad \forall i\in\{1,\dots,N_x\} \\ &y_j^\top\theta+\rho+\lambda\ge 0, \quad \forall j\in\{1,\dots,N_y\} \\ &\|\theta\|_2\le 1 \\ &\theta,x_i,y_j\in\mathbb{R}^p, \rho\in\mathbb{R}, \lambda\in\mathbb{R}_{\ge 0}, \end{aligned}$$

where p is lately referred to as *dimension* of the classification problem, x_i are the N_x features with label $+1$ and y_i are the N_y features with label -1. For sake of simplicity we assume $N_x = N_y$. In our experiments, x_i and y_i are sampled from normal distributions, $\mathcal{N}(\mathbf{0}_p, 25\boldsymbol{I}_p)$ and $\mathcal{N}(50\cdot\mathbf{1}_p, 25\boldsymbol{I}_p)$, respectively. Notice that the problem is feasible with probability 1, hence the combinatorial dimension is $d_{\text{comb}} = p+2$ (the decision variable also comprises the scalars ρ and λ). Moreover the problem satisfies the unique minimum condition, which is enforced by the constraint $\|\theta\|_2\le 1$, [10].

We first study the parallel solution of the robust classification problem on a complete graph for different number of processors and for 20 different realizations of the constraints (*scenarios*). For this test we consider $p=4$ and $N_x = 10^5$. The simulation results are shown in Figure 1. The top figure shows the average computation time for different numbers of processors. It can be seen that the average computation time decreases with the number of processors. The average processing time saturates for number of processors larger than 50. The bottom figure shows the average number of iterations required for convergence. The average number of active constraints exchanged at each communication round was 4.8.

In the second experiments we investigate the scalability of the approach with respect to the number of constraints in the global RCP. We consider 20 different realizations of the constraints, dimension $p=4$ and number of processors $n=50$. In Figure 2 we show the computational time, the number of iterations and the average number of active constraints

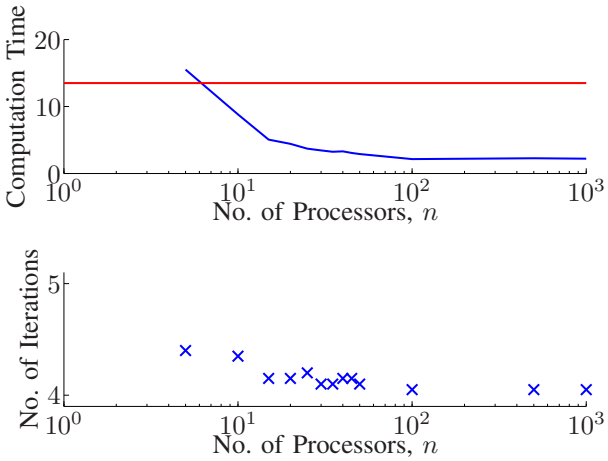

Fig. 1. Computation time and number of iterations versus number of nodes for the parallel solution of the robust classification problem. Results are averaged over 20 tests. In the top figure, the red line represents the centralized solution while the blue line represents the proposed parallel solution.

exchanged among nodes for $N_x = \{10^2, 10^3, 10^4, 10^5\}$. It is possible to notice that the computational effort of the parallel approach has better scalability properties with respect to the centralized implementation, which shows a sharp increase of the CPU time for $N_x > 10^3$. Therefore, the proposed approach has a clear computational advantage over the centralized solution for problems with large number of constraints. Moreover, the number of iterations shows low sensitivity with respect to the number of constraints of the RCP.

As a last test regarding the parallel implementation, we study the performance of the algorithm for different problem dimensions. In Figure 3 we report the computational effort of the proposed approach compared with the centralized implementation for different values of p. Also in this case, the ACC algorithm shows low sensitivity to problem dimension if compared with its centralized counterpart. The lower plots in Figure 3 show the number of iterations and the average number of constraints exchanged by the nodes at each communication round.

As a conclusion of this section we report further numerical results concerning other graph topologies, which can be of interest for distributed applications in sensor networks. We consider $p = 2$ and $N_x = 10^3$. In Table I we report number of iterations, number of active constrains and average graph diameter for *geometric random graph* and *chain graph*.

VI. Conclusion

We considered a setup in which the constraints of a random convex program are distributed among several nodes in a processor network and the network is in charge of solving the global RCP, satisfying all problem constraints.

We first analyzed the properties of the constraint sets in a convex program and then we exploited such properties for devising a distributed algorithm, named *active constraints consensus*. The algorithm allows network nodes to reach consensus on problem solution by exchanging a local set

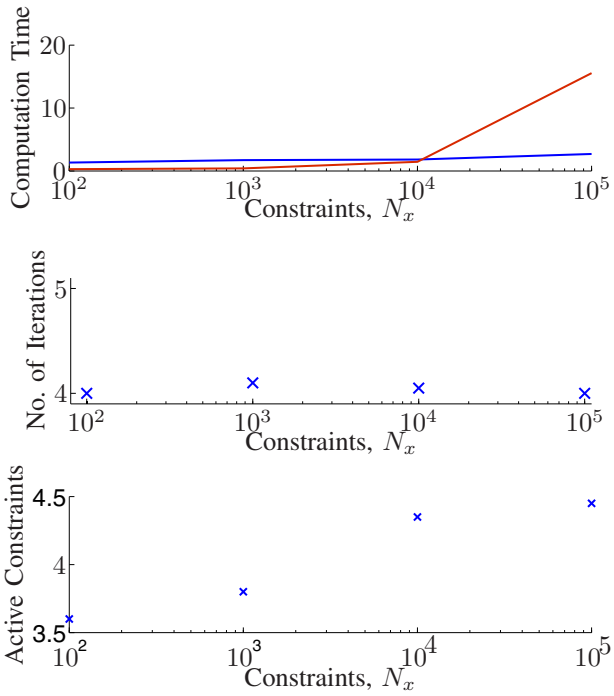

Fig. 2. Computation time, number of iterations and number of active constraints versus N_x for the parallel solution of the robust classification problem. Results are averaged over 20 tests. In the top figure, the red line represents the centralized solution while the blue line represents the proposed parallel solution.

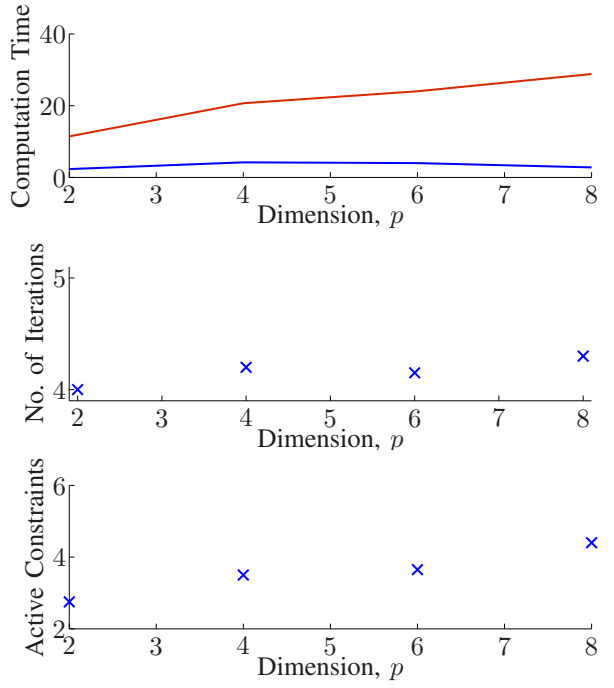

Fig. 3. Computation time, number of iterations and number of active constraints versus problem dimension for the parallel solution of the robust classification problem. Results are averaged over 20 tests. In the top figure, the red line represents the centralized solution while the blue line represents the proposed parallel solution.

	No. Processors	Diameter	Iterations	Active Constraints
Geometric	10	1	4.1	
random	30	1.95	6.45	2.44
graph	50	2	6.85	
Chain	10	10	30.5	
graph	30	30	89.35	2.45
	50	50	151.1	

TABLE I

NUMBER OF ITERATIONS, NUMBER OF ACTIVE CONSTRAINTS AND DIAMETER FOR DIFFERENT GRAPH TOPOLOGIES.

of constraints at each iteration. The algorithm assures finite-time convergence to the *scenario solution*, which preserves the probabilistic guarantees of the RCP theory. The theoretical derivation was then validated by means of numerical experiments on a distributed classification problem.

VII. APPENDIX

Appendix I: Lexicographic Solution

We here discuss an approach for computing the unique lexicographic solution for a given convex optimization problem $P[C]$. Consider a generic convex problem, possibly not satisfying the unique minimum condition. We denote with $Opx[C]$ the optimal solutions set, i.e., $Opx[C] = \{x \mid a^\top x = J^*(C) \text{ and } f_j(x) \leq 0, \forall j \in C\}$. We now want to compute the unique solution $x^* = [x_1^*, \ldots, x_d^*]^\top \in Opx[C],\ d > 1$ satisfying $x^* \leq_l x$, for any $x \in Opx[C]$; the symbol $\leq_l$ denotes the lexicographic ordering: $x^* \leq_l x$ if and only if $x_1^* \leq x_1$, or $x_1^* = x_1$ and $x_2^* \leq x_2$, etc.

Proposition 5: Given the optimal objective $J^*(C)$ of a generic convex problem $P[C]$, the lexicographic solution x^* can be obtained by solving the following convex program:

$$\begin{aligned} P_l[C]: \quad \min_{x \in X} \quad & \sum_{j=1}^{d} \lambda^{j-1} x_j \quad \text{subject to :} \\ & f_j(x) \leq 0, \quad \forall j \in C \\ & a^\top x = J^*(C) \end{aligned} \tag{6}$$

for some arbitrary small positive λ.

Proof. We start by noticing that the feasible set of problem (6) corresponds to the optimal solutions set of the original program $P[C]$; this is consequence of the fact that the additional constraint $a^\top x = J^*(C)$ restricts the feasible set to the x attaining the optimal value $J^*(C)$. Therefore it only remains to demonstrate that the coefficients of the optimization variables force the lexicographic ordering. For this purpose assume to have two solutions $\hat{x}^*$ and $\bar{x}^*$, such that $\hat{x}_1^* - \bar{x}_1^* \doteq \gamma > 0$. Let us consider the Euclidean ball of radius M enclosing the compact domain X. If we call $J_l(x)$ the objective value of problem (6) at x, it is then easy to see that selecting $0 < \lambda < \frac{\gamma}{2(d-1)M} < 1$ assures that $J_l(\hat{x}^*) > J_l(\bar{x}^*)$:

$$J_l(\hat{x}^*) - J_l(\bar{x}^*) = \sum_{j=1}^{d} \lambda^{j-1} \hat{x}_j^* - \sum_{j=1}^{d} \lambda^{j-1} \bar{x}_j^* =$$

$$= \gamma + \sum_{j=2}^{d} \lambda^{j-1} (\hat{x}_j^* - \bar{x}_j^*) \geq \gamma - 2M \sum_{j=2}^{d} \lambda^{j-1} \overset{\lambda<1}{\geq}$$

$$\overset{\lambda<1}{\geq} \gamma - 2\lambda M(d-1) \overset{\lambda < \frac{\gamma}{2(d-1)M}}{>} 0.$$

Repeating the same reasoning for the remaining variables (e.g., comparing the objectives in the case in which $\hat{x}_1^* = \bar{x}_1^*$) it is possible to show that, for some small λ, problem (6) imposes the lexicographic ordering. □

We notice that this approach is convenient if some reasonable bound M is *a priori* known. In case this bound is not known or it may cause numerical problems (e.g., λ is comparable with the machine precision), it can be preferable to consider alternative approaches for computing the lexicographic solution, as the one presented in [11].

Appendix II: Support Constraints of Convex Programs

In this appendix we prove that every support constraints of a convex program has to belong to the active constraints set. Let c be a support constraint for problem (1). Call $\hat{x}^* = x^*(C)$ and $\bar{x}^* = x^*(C \backslash \{c\})$. From the definition of support constraints, it holds that $a^\top \bar{x}^* < a^\top \hat{x}^*$. Suppose then, for the purpose of contradiction, that c is not active at $\hat{x}^*$, i.e. that $f_c(\hat{x}^*) < 0$. Consider a point z on the segment connecting $\hat{x}^*$ and $\bar{x}^*$: $z(\lambda) = \lambda \hat{x}^* + (1-\lambda)\bar{x}^*$, $\lambda \in [0,\ 1]$. It is immediate to check that $a^\top z(\lambda) < a^\top \hat{x}^*$ for all $\lambda > 0$. Moreover, by convexity, $z(\lambda)$ is feasible for all constraints, except possibly for c. However, since $\hat{x}^*$ is in the interior of the convex set defined by $f_c \leq 0$, there must exist values of λ sufficiently small such that $z(\lambda)$ satisfies also $f_c(z(\lambda)) \leq 0$. But then $z(\lambda)$ 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 $\hat{x}^*$.

REFERENCES

[1] A. Ben-Tal and A. Nemirovski. Robust optimization: Methodology and applications. *Math. Program.*, 92:453–480, 2002.

[2] A. Prékopa. *Probabilistic programming*. Elsevier, 2003.

[3] G.C. Calafiore. Random convex programs. *SIAM J. Optim.*, 20(6):3427–3464, 2010.

[4] J.N. Tsitsiklis. Problems in decentralized decision making and computation. *PhD thesis, Massachusetts Institute of Technology*, 1984.

[5] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall, 1989.

[6] A. Nedíc and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[7] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–124, 2010.

[8] G. Notarstefano and F. Bullo. Distributed abstract optimization via constraints consensus: Theory and applications. *IEEE Transactions on Automatic Control*, 56(10):2247–2261, 2011.

[9] M. Burger, G. Notarstefano, F. Bullo, and F. Allgower. A distributed simplex algorithm for degenerate linear programs and multi-agent assignment. *Automatica*, May 2011. Submitted.

[10] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[11] G. Calafiore and M.C. Campi. The scenario approach to robust control design. *IEEE Transactions on Automatic Control*, 51(5):742–753, 2006.