# Resilience for Distributed Consensus With Constraints

Xuan Wang, Shaoshuai Mou, *Member, IEEE*, and Shreyas Sundaram, *Senior Member, IEEE*

***Abstract*—This article proposes a new approach that enables multiagent systems to achieve resilient *constrained* consensus in the presence of Byzantine attacks, in contrast to existing literature that is only applicable to *unconstrained* resilient consensus problems. The key enabler for our approach is a new device called a $(\gamma_i, \alpha_i)$-*Resilient Convex Combination*, which allows normal agents in the network to utilize their locally available information to automatically isolate the impact of the Byzantine agents. Such a resilient convex combination is computable through linear programming, whose complexity scales well with the size of the overall system. By applying this new device to multiagent systems, we introduce network and constraint redundancy conditions under which resilient constrained consensus can be achieved with an exponential convergence rate. We also provide insights on the design of a network such that the redundancy conditions are satisfied. Finally, numerical simulations and an example of safe multiagent learning are provided to demonstrate the effectiveness of the proposed results.**

***Index Terms*—Multiagent consensus, resilient distributed coordination.**

## I. Introduction

THE emergence of *multiagent* systems is motivated by modern applications that require more efficient data collection and processing capabilities in order to achieve a higher level of autonomy [1]. For multiagent systems, the main challenge lies in the requirement of a scalable control algorithm that allows all agents in the system to interact in a cooperative manner. Toward this end, traditional approaches equipped with a centralized coordinator can suffer from poor performance due to limitations on the coordinator's computation and communication capabilities. Motivated by this, significant research effort has been devoted to developing *distributed* control architectures [2], [3], [4], which enable multiagent systems to achieve global objectives through only local coordination. A specific example is *consensus*, where each agent controls a local state, which it repeatedly updates as a weighted average of its neighbors' states [5], [6], [7], [8], [9]. In this way, the states of all agents will gradually approach the same value [10], [11], [12], allowing the system to achieve certain global control or optimization goals. Recently, the consensus approach has powered many applications, including motion synchronization [13], multirobot path planning/formation control [14], [15], flocking of mobile robots [16], and cooperative sensing [17], [18], across the overall network.

While consensus algorithms are simple and powerful, their effectiveness heavily depends on the assumption that all agents in the network correctly execute the algorithm. In practice, however, the presence of large numbers of heterogeneous agents in multiagent systems (provided by different vendors, with differing levels of trust) introduces the potential for some of the agents to be malicious. These agents may inject misinformation that propagates throughout the network and prevent the entire system from achieving proper consensus-based coordination. As shown in [19], even one misbehaving agent in the network can easily lead the states of other agents to any value it desires. Traditional information-security approaches [20] (built on the pillars of confidentiality, integrity, and availability) focus on protecting the data itself, but are not directly applicable to consensus-based distributed algorithms for cooperative decision-making, as such processes are much more sophisticated and vulnerable than simple data transmission. Because of this, instead of narrowly focusing on techniques, such as data encryption and identity verification [20], researchers are motivated to seek fundamentally new approaches that allow the network to reliably perform consensus even after the adversary has successfully compromised certain agents in the system. Toward this end, the main challenges arise from the special characteristics of distributed systems where no agent has access to global information. As a consequence, classical paradigms, such as fault-tolerant control [21], are usually not sufficient to capture the sophisticated actions (i.e., Byzantine failures[1]) the adversarial agents take to avoid detection.

Received 7 July 2024; revised 15 February 2025; accepted 18 April 2025. Date of publication 29 April 2025; date of current version 29 September 2025. This work was supported in part by NSF ECCS under Award 2332210, in part by ARL Award under Grant W911NF-25-2-0011, in part by NASA-ULI under Award 80NSSC20M0161, and in part by NSF CAREER under Award 1653648. Recommended by Associate Editor Z. Li. *(Corresponding author: Xuan Wang.)*

Xuan Wang is with the Department of Electrical and Computer Engineering, George Mason University, Fairfax, VA 22030 USA (e-mail: xwang64@gmu.edu).

Shaoshuai Mou is with the School of Aeronautics and Astronautics, Purdue University, West Lafayette, IN 47906 USA (e-mail: mous@purdue.edu).

Shreyas Sundaram is with the Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47906 USA (e-mail: sundara2@purdue.edu).

Digital Object Identifier 10.1109/TAC.2025.3565681

[1] Byzantine failures are considered the most general and most difficult class of failures among the failure modes. They imply no restrictions, which means that the failed agent can generate arbitrary data, including data that makes it appear like a functioning (normal) agent.

*Literature review:* Motivated by the need to address security for consensus-based distributed algorithms, recent research efforts have aimed to establish more advanced approaches that can minimize the impact of Byzantine attacks. Filtering-based anomaly detection [22], [23] has been successfully developed to identify and mitigate the impact of false data injected by attackers to maintain the normal functionality of a network system, by assuming the attackers follow certain stochastic models. Complementary to anomaly detection, in this article, we seek to develop control algorithms that are inherently resilient to Byzantine attacks, i.e., allowing the normal agents to utilize only their local information to automatically isolate misinformation from attackers without the need to identify them. Along this direction, relevant literature dates back to the results in [24] and [25], which reveal that a resilient consensus can be achieved among $m$ agents with at most $(m-1)/3$ Byzantine attackers. However, these techniques require significant computational effort by the agents (and global knowledge of the topology). To address this, the mean subsequence reduced algorithms were developed [26], [27], based on the idea of letting each normal agent run distributed consensus by excluding the most extreme values from its neighbor sets at each iteration. Under certain conditions on the network topology, the effectiveness of these algorithms can be theoretically validated. Followed by [26] and [27], similar approaches have been further applied to resilient distributed optimization [28], [29], [30], [31] and multiagent machine learning [32]. Note that in the results of [26] and [27], the local states for all agents must be scalars. If the agents control multidimensional states, one possible solution is to run the scalar consensus separately on each entry of the state. However, such a scheme will violate the convex combination property of distributed consensus (i.e., the agents' states in the new time step must be located within the convex hull of its neighbors' states at the current time step,) and thus, lead to a loss of the spatial information encoded therein. In order to maintain such convex combination property among agents' states, methods based on Tverberg points have been proposed for both centralized [33] and distributed cases [34], [35], [36], [37]. However, according to the authors in [38] and [39], the computational complexity of calculating exact Tverberg points is typically high. To reduce the computational complexity, new approaches based on *resilient convex combinations* [33], [40], [41], [42], [43] have been developed, which can be computed via linear or quadratic programming with lower complexity. Apart from studying the computational complexity for each iteration of the resilient consensus algorithms, there are also results focusing on the convergence rate of consensus. For example, Vaidya [34] used a multiset generalization to iteratively compute the resilient convex combination, and Yan et al. [43] introduced an entrywise ordering mechanism to design a special safe region for the resilient convex combination. Both approaches can achieve exponential convergence rates.

It is worth pointing out that the results mentioned above are applicable to *unconstrained* resilient consensus problems. To the best of the authors' knowledge, there is no existing literature that expands these results to constrained consensus problems where the consensus value must remain within a certain set. Such problems play a significant role in the field of distributed control and optimization [44], [45], [46]. This motivates the goal of this article: developing a fully distributed algorithm that can achieve resilient constrained consensus for multiagent systems with exponential convergence rate. The key challenges in such settings are as follows. 1) To satisfy local constraints, each agent must update its local state by a resilient convex combination of its neighbors' states, precluding the use of entrywise consensus. 2) The proposed approach should guarantee an exponential convergence rate. 3) For constrained consensus problems, to isolate the impact of malicious information from Byzantine agents, in addition to the network redundancy, one also needs constraint redundancy.

*Statement of contributions:* We study the resilient consensus algorithm for multiagent systems with local constraints. Originated from our previous preliminary work in [41], we introduce a new device named $(\gamma_i, \alpha_i)$-*Resilient Convex Combination*. Given a maximum number of Byzantine neighbors for each agent, we introduce an approach that allows the multiagent system to achieve resilient constrained consensus with an exponential convergence rate. We provide conditions on network and constraint redundancies under which the effectiveness of the proposed algorithm can be guaranteed. We observe that the proposed network redundancy condition is difficult to verify directly. Thus, we introduce a sufficient condition, which is directly verifiable and offers an easy way to design network typologies satisfying the required condition. All the results in the article are validated with examples in unconstrained consensus, constrained consensus, and safe multiagent learning. Our result differs from existing literature in three major aspects. First, inspired by [34], this article formalizes the concept of $(\gamma_i, \alpha_i)$-Resilient Convex Combination. It introduces tunable parameters $\gamma_i$ and $\alpha_i$, which quantify how many (i.e., $\gamma_i$) and how much weight (i.e., $\alpha_i$) of the information each agent uses from normal agents to update its state. In contrast, most existing works [33], [35], [36], [37], [41] only ensure the obtained fusion vector to be a resilient convex combination; Vaidya[34] was able to quantify $\gamma_i$ and $\alpha_i$ but the values are not tunable. Second, our approach has an exponential convergence rate, which was not guaranteed in our previous work [41]. The related work in [34] first introduced a special way of using Tverberg points for resilient consensus and achieved an exponential convergence rate. While our approach shares similar properties with the one in [34], the key mechanism differs from using Tverberg points. Furthermore, the method proposed in this article has a less restrictive safe area compared with the use of the Tverberg point [34] and the safe region in [43], which uses an entrywise ordering mechanism. Third, as far as we know, this work is the first to study resilient consensus with constraints, in contrast to [33], [34], [35], [36], [40], [41], [42], and [37], which are designed only for unconstrained consensus. The involvement of constraints requires the introduction of extra redundancy conditions to guarantee convergence. It is worth mentioning that the redundancy conditions we derived to handle constraints are not restricted to the $(\gamma_i, \alpha_i)$-*resilient convex combination* proposed in this article but can also be integrated into the results

in [33], [34], [35], [36], [37], [40], [41], and [42] by applying the same redundancy conditions accordingly.

The rest of this article is organized as follows. Section II presents the mathematical formulation of the distributed resilient constrained-consensus problem. In Section III, we introduce a new device, named $(\gamma_i, \alpha_i)$-Resilient Convex Combination, and propose an algorithm that allows the normal agents in the network to obtain such a resilient convex combination only based on its local information. Section IV presents how the $(\gamma_i, \alpha_i)$-Resilient Convex Combination can be applied to solve resilient constrained consensus problems and specifies under which conditions (network and constraint redundancies) the effectiveness of the proposed algorithm can be theoretically guaranteed. Section V presents numerical simulations and an example of safe multiagent learning to validate the results. Finally, Section VI concludes this article.

*Notations:* Let $\mathbb{Z}$ denote the set of integers and $\mathbb{R}$ denote the set of real numbers. Let $x_{\mathcal{A}} = \{x_j \in \mathbb{R}^n, j \in \mathcal{A}\}$, denote a set of states in $\mathbb{R}^n$, where $\mathcal{A}$ is a set of agents. Given $x_1, x_2, \ldots, x_r \in \mathbb{R}^n$, let $\text{col}\{x_j | j = 1, 2, \ldots, r\} = [x_1^\top \;\; x_2^\top \cdots x_r^\top]^\top \in \mathbb{R}^{rn}$ denote a column stack of vectors; let $\text{row}\{x_j | j = 1, 2, \ldots, r\} = [x_1 \;\; x_2 \cdots x_r] \in \mathbb{R}^{n \times r}$ denote a row stacked matrix of vectors. Let $\mathcal{V}(\mathbb{G})$ denote the agent (vertex) set of a graph $\mathbb{G}$. Let $\mathbf{1}_r$ denote a vector in $\mathbb{R}^r$ with all its components equal to 1. Let $I_r$ denote the $r \times r$ identity matrix. For a vector $\beta$, by $\beta > 0$ and $\beta \geq 0$, we mean that each entry of vector $\beta$ is positive and nonnegative, respectively.

## II. Problem Formulation

Consider a network of $m$ agents in which each agent $i$ is able to receive information from certain other agents, called agent $i$'s neighbors. Let $\mathcal{N}_i(t), i \in \{1, \ldots, m\}$, denote the set[2] of agent $i$'s neighbors at time $t$. The neighbor relations can be characterized by a *time-dependent* graph $\mathbb{G}(t)$ such that there is a directed edge from $j$ to $i$ in $\mathbb{G}(t)$ if and only if $j \in \mathcal{N}_i(t)$. For all $i$ and $t$, we assume that $i \in \mathcal{N}_i(t)$. Let $m_i(t) = |\mathcal{N}_i(t)|$, which is the cardinality of the neighbor set. Based on network $\mathbb{G}(t)$, in a consensus-based distributed algorithm, each agent is associated with a local state $x_i \in \mathbb{R}^n$; the goal is to reach a consensus for all agents, such that

$$x_1 = \cdots = x_m = x^*. \tag{1}$$

### A. Consensus-Based Distributed Algorithms

In order to achieve the $x^*$ in (1), consensus-based distributed algorithms usually require each agent to update its state by an equation of the form

$$x_i(t+1) = f_i(v_i(t), x_i(t)) \tag{2}$$

where $f_i(\cdot)$ is an operator associated with the problem being tackled by the agents. For example, it may arise from constraints (as a projection operator), objective functions (as a gradient descent operator), and so on. The quantity $v_i(t)$ in (2) is a convex combination of agent $i$'s neighbors' states, i.e.,

$$v_i(t) = \sum_{j \in \mathcal{N}_i(t)} w_{ij}(t) x_j(t). \tag{3}$$

Here, $w_{ij}(t) \geq 0$ is a weight that agent $i$ places on its neighboring agent $j \in \mathcal{N}_i(t)$, with $\sum_{j \in \mathcal{N}_i(t)} w_{ij}(t) = 1$ for all $i \in \{1, \ldots, m\}$ and $t = 1, 2, 3, \ldots$. Note that different choices of $w_{ij}(t)$ can lead to different $v_i(t)$, which can influence the convergence of the algorithm. One feasible choice of $w_{ij}(t)$ can be obtained by letting $w_{ij}(t) = \frac{1}{|\mathcal{N}_i(t)|}$, which uniformly shares the weights among the neighbors [45].

With the $v_i(t)$ in (3), we further specify the form of update (2) for solving constrained consensus problems. Suppose the agents of the network are subject to local convex constraints $x_i \in \mathcal{X}_i$, $i \in \{1, \ldots, m\}$, with $\bigcap_{i=1}^m \mathcal{X}_i \neq \emptyset$. To obtain a consensus value in (1) such that

$$x^* \in \bigcap_{i=1}^m \mathcal{X}_i \tag{4}$$

update (2) can take the form

$$x_i(t+1) = \mathcal{P}_i(v_i(t)) \tag{5}$$

where $\mathcal{P}_i(\cdot)$ is a projection operator that projects any state to the local constraint $\mathcal{X}_i$ [45]. Using this update, if all agents reach a consensus, i.e., $x_i(t) = v_i(t) = x_i(t+1) = x^* \; \forall i$, then one has $x^* = \mathcal{P}_i(x^*)$. This yields $x^* \in \bigcap_{i=1}^m \mathcal{X}_i$ satisfying all local constraints. Note that if the local constraint does not exist, i.e., $\mathcal{X}_i = \mathbb{R}^n \; \forall i \in \{1, \ldots, m\}$, then $\mathcal{P}_i(v_i(t)) = v_i(t)$ is an identity-mapping.

### B. Distributed Consensus Under Attack

Suppose the network $\mathbb{G}(t)$ is attacked by a number of Byzantine agents. Each Byzantine agent can connect itself to the normal agents in $\mathbb{G}(t)$ in a time-varying manner, and can send arbitrary state information to those connected agent(s). Note that if one Byzantine agent is simultaneously connected to multiple normal agents, then it does not necessarily have to send the same state information to all of those agents. While the goal of the Byzantine agents is to prevent the normal agents from reaching a consensus, their behaviors are not restricted to a specific strategy. Instead, we assume that Byzantine agents have superior knowledge of the normal agents' behaviors and can always cause the worst-case impacts to try to prevent normal agents from reaching constrained consensus. We use $\mathbb{G}^+(t)$ to characterize the new graph where both normal and Byzantine agents are involved. Accordingly, for normal agents $i = 1, \ldots, m$, we use $\mathcal{M}_i(t)$ to denote the set of its Byzantine neighbors. For all $t = 1, 2, 3, \ldots$, we assume that the number of each agent's Byzantine neighbors is $|\mathcal{M}_i(t)| = \kappa_i(t)$. Further for all $t = 1, 2, \ldots$ and $i \in \{1, \ldots, m\}$, we assume that $\kappa_i(t) \leq \bar{\kappa} \in \mathbb{Z}_+$ is bounded. Note that the normal agents do not know which, if any, of their neighbors are Byzantine.

Under network $\mathbb{G}^+(t)$, if one directly applies the constrained consensus algorithm, then the malicious states sent by Byzantine

[2] Here, the system is not under Byzantine attack. We use $\mathcal{N}_i(t)$ to denote all (normal) neighbors of agent $i$. Later, when Byzantine agents are present, we will use $\mathcal{N}_i^+(t)$ to include both normal and Byzantine agents.

agents will be injected into the convex combination (3) as

$$v_i(t) = \sum_{j\in\mathcal{N}_i(t)} w_{ij}(t)x_j(t) + \sum_{k\in\mathcal{M}_i(t)} w_{ik}(t)x_{ik}(t) \quad (6)$$

where $x_{ik}(t)$ is the state information sent by the Byzantine agent $k$ to the normal agent $i$, $w_{ij}(t), w_{ik}(t) \geq 0, i \in \{1,\ldots,m\}, j \in \mathcal{N}_i(t)$, $k \in \mathcal{M}_i(t)$, and $\sum_{j\in\mathcal{N}_i(t)} w_{ij}(t) + \sum_{k\in\mathcal{M}_i(t)} w_{ik}(t) = 1$. A simple way of choosing the weights is by letting $w_{ij}(t) = w_{ik}(t) = \frac{1}{|\mathcal{N}_i(t)|+|\mathcal{M}_i(t)|}$. According to (6), since $x_{ik}(t)$ can take arbitrary values, no matter how $w_{ij}$ and $w_{ik}$ are chosen, as long as $w_{ik}(t)$ are not all equal to zero, the Byzantine agents can fully manipulate $v_i(t)$ to any desired value and, therefore, prevent the normal agents from reaching a consensus.

### C. Problem: Resilient Constrained Consensus

In this article, *the problem of interest* is to develop a novel algorithm such that each normal agent in the system can mitigate the impact of its (unknown) Byzantine neighbors when calculating its local consensus combination; we call this a resilient convex combination. Then, by employing such a resilient convex combination into update (5), all agents in the system are able to achieve resilient *constrained* consensus with *exponential* convergence rate even in the presence of Byzantine attacks.

## III. Algorithm to Achieve the $(\gamma_i, \alpha_i)$-Resilient Convex Combination

Since the convex combination $v_i(t)$ in (6) is the key reason why Byzantine agents can compromise a consensus-based distributed algorithm, it motivates us to propose a distributed approach for the normal agents in the network to achieve the following objectives. 1) The approach should allow the normal agents to choose a $v_i^\star(t)$ subject to (6), such that $w_{ik}^\star(t) = 0$, for $\forall k \in \mathcal{M}_i(t)$. This guarantees that all information from Byzantine agents is automatically removed from $v_i^\star(t)$. 2) The obtained $v_i^\star(t)$ should guarantee that, for normal neighbors of agent $i$, $j \in \mathcal{N}_i(t)$, a certain number of $w_{ij}^\star(t)$ is lower bounded by a positive constant. This guarantees that a consensus among normal agents can be reached with a proper convergence rate.

### A. Resilient Convex Combination

In line with the two objectives proposed above, for each normal agent $i$, we define the following concept, named a *$(\gamma_i, \alpha_i)$-resilient convex combination.*

*Definition 1 ($(\gamma_i, \alpha_i)$-Resilient Convex Combination):* A vector $v_i^\star(t)$ computed by agent $i \in \{1,\ldots,m\}$ is called a *resilient convex combination* of its neighbor's states if it is a convex combination of its *normal* neighbors' states, i.e., then there exists $w_{ij}^\star(t) \in [0,1]$, $j \in \mathcal{N}_i(t)$, such that

$$v_i^\star(t) = \sum_{j\in\mathcal{N}_i(t)} w_{ij}^\star(t)x_j(t) \quad (7)$$

with $\sum_{j\in\mathcal{N}_i(t)} w_{ij}^\star(t) = 1$. Furthermore, for $\gamma_i \in \mathbb{N}$ and $\alpha_i \in (0,1)$, the resilient convex combination is called a $(\gamma_i, \alpha_i)$-*resilient*, if at least $\gamma_i$ of $w_{ij}^\star(t)$ satisfy $w_{ij}^\star(t) \geq \alpha_i$.

Moving forward, let $\mathcal{N}_i^+(t) \triangleq \mathcal{N}_i(t) \bigcup \mathcal{M}_i(t)$ be the neighbor set of $i$, which is composed of both normal and Byzantine agents.

*Assumption 1:* Suppose for $\forall t = 1,2,3,\ldots,$ and $\forall i \in \{1,\ldots,m\}$, the neighbor sets satisfy $|\mathcal{N}_i^+(t)| \geq (n+1)\kappa_i(t) + 2$, where $n$ is the dimension of the state and $\kappa_i(t)$ is the number of the Byzantine neighbors of agent $i$.

Note that this assumption is a local condition in terms of a minimal cardinality of neighbor sets for each normal agent to compute a resilient convex combination. It will be used later in the theorems to guarantee the existence of the resilient convex combination. This assumption is mild when $n$ (state dimension) is small, and $\kappa_i(t)$ is bounded. The same bound has been used in the existing literature [33], [34], [35], [43] and has been proven necessary and sufficient to compute a resilient convex combination. In addition, if $\kappa_i(t)$ is unknown, one can replace the $\kappa_i(t)$ by an upper bound of $\kappa_i(t)$ $\forall i,t$, denoted by $\bar{\kappa}$. To continue, define the following concept.

*Definition 2 (Convex hull):* Given a set of agents $\mathcal{A}$, and the corresponding state set $x_\mathcal{A}$, the convex hull of vectors in $x_\mathcal{A}$ is defined as

$$\mathcal{H}(x_\mathcal{A}) = \left\{ \sum_{j=1}^{|x_\mathcal{A}|} \omega_j x_j : x_j \in x_\mathcal{A}, \omega_j \geq 0, \sum_{j=1}^{|x_\mathcal{A}|} \omega_j = 1 \right\}. \quad (8)$$

### B. Algorithm for Resilient Convex Combination

The above definition allows us to present Algorithm 1 for agent $i$ to achieve a $(\gamma_i, \alpha_i)$-resilient convex combination, which is only based on the agent's locally available information. Note that since all the results in this section only consider the states for a particular time step $t$, for the convenience of presentation, we *omit* the time step notation $(t)$ in the following algorithm description.

*Theorem 1. (Validity of 1):* Suppose Assumption 1 holds. For agent $i$, given an integer $\sigma_i \in [n\kappa_i + 2 \;\; d_i - \kappa_i]$, by 1, the obtained $v_i^\star$ is a $(\gamma_i, \alpha_i)$-resilient convex combination, where $\gamma_i = d_i - \kappa_i - \sigma_i + 2$ and $\alpha_i = \frac{1}{(s_i+1)(n+1)}$. Here, $d_i = |\mathcal{N}_i^+|$ and $s_i = \binom{d_i-1}{\sigma_i-1}$.

*Remark 1:* Note that for a certain time step $t$, if agent $i$ has $d_i$ neighbors, then by running Algorithm 1, the obtained $v_i^\star$ is a convex combination of at least $\gamma_i = d_i - \kappa_i - \sigma_i + 2$ normal neighbors. This means in order to completely isolate the information from $\kappa_i$ Byzantine agents, the agent $i$ may simultaneously lose the information from $\sigma_i - 2$ of its normal neighbors. Then, the $\bigcap_{k=1}^{b_i} \mathcal{H}(x_{\mathcal{B}_{ijk}})$ defined in line 11 of Algorithm 1 guarantees no matter which neighboring agents are Byzantine, the obtained vector $v_i$ can always be represented as a convex combination of its normal neighbors states. ■

To better understand Algorithm 1 before proving Theorem 1, we provide Fig. 1 to visualize its execution process. In a 2-D ($n = 2$) space, consider an agent $i = 1$, which has $d_i = 6$ neighbors, denoted by agents 1–6. Suppose $\kappa_i = 1$, and let $\sigma_i = 5 \in [n\kappa_i + 2 \;\; d_i - \kappa_i]$. Then, for step 4 of

**Algorithm 1:** Compute a $(\gamma_i, \alpha_i)$-Resilient Convex Combination $v_i^\star$ for Agent $i$.

1 **Input** $\mathcal{N}_i^+$, $x_{\mathcal{N}_i^+}$, and $\kappa_i$.
2 *Let* $d_i = |\mathcal{N}_i^+|$.
3 *Choose* an integer $\sigma_i \in [n\kappa_i + 2\ ,\ d_i - \kappa_i]$ ; // Note that $d_i \geq (n+1)\kappa_i + 2$, so the set is non-empty.
4 *Let* $s_i = \binom{d_i - 1}{\sigma_i - 1}$. Construct all possible subsets $\mathcal{S}_{ij} \subset \mathcal{N}_i^+$, $j \in \{1, \cdots, s_i\}$, such that $i \in \mathcal{S}_{ij}$, and $|\mathcal{S}_{ij}| = \sigma_i$ ; // An illustration of sets $\mathcal{S}_{ij}$ is given in Fig. 1.
5 **for** $j = 1, \cdots, s_i$ **do**
6 *Define* $\widehat{\mathcal{S}}_{ij} \triangleq \mathcal{S}_{ij} \setminus \{i\}$.
7 **if** $x_i \in \mathcal{H}(x_{\widehat{\mathcal{S}}_{ij}})$ **then**
8 *Let* $\phi_{ij} = x_i$.
9 **else**
10 *Let* $b_i = \binom{\sigma_i - 1}{\sigma_i - \kappa_i - 1} = \binom{\sigma_i - 1}{\kappa_i}$. For $k \in \{1, \cdots, b_i\}$, construct sets $\mathcal{B}_{ijk}$, whose elements are from $\mathcal{S}_{ij}$, i.e., $\mathcal{B}_{ijk} \subset \mathcal{S}_{ij}$, while ensuring that $i \in \mathcal{B}_{ijk}$, and $|\mathcal{B}_{ijk}| = \sigma_i - \kappa_i$.
11 *Let* $\phi_{ij} \in \left(\bigcap_{k=1}^{b_i} \mathcal{H}(x_{\mathcal{B}_{ijk}})\right) \bigcap \mathcal{H}(x_{\widehat{\mathcal{S}}_{ij}})$ ; // Later, we will show that the set defined here is non-empty.
12 **end**
13 **end**
14 *Compute* $v_i^\star = \dfrac{\sum_{j=1}^{s_i} \phi_{ij} + x_i}{s_i + 1}$.

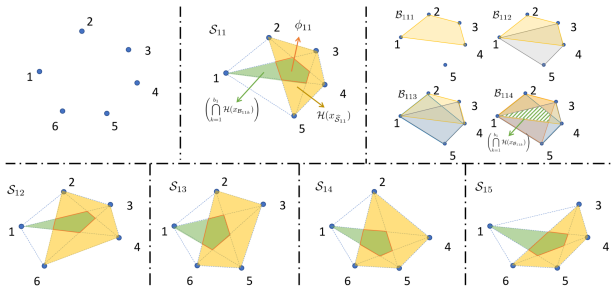


Fig. 1. Visualizations of Algorithm 1 for computing intersections of convex hulls. Green shades: $\bigcap_{k=1}^{b_i} \mathcal{H}(x_{\mathcal{B}_{ijk}})$. Yellow shades: $\mathcal{H}(x_{\widehat{\mathcal{S}}_{ij}})$. Orange outline: intersected area for choosing $\phi_{ij}$. The example focuses on agent 1 with $(i = j = 1)$.

Algorithm 1, one can construct $s_i = \binom{5}{4} = 5$ sets $\mathcal{S}_{ij}$. For example, $\mathcal{S}_{11} = \{1, 2, 3, 4, 5\}$, $\mathcal{S}_{12} = \{1, 2, 3, 4, 6\}$, and so forth. Consider $j = 1$ in step 5. Then, $\widehat{\mathcal{S}}_{11} = \mathcal{S}_{11} \setminus \{i\} = \{2, 3, 4, 5\}$, and accordingly, the set $\mathcal{H}(x_{\widehat{\mathcal{S}}_{11}})$ must be the yellow area in the top middle figure of Fig. 1. Since in the given example, one has $x_i \notin \mathcal{H}(x_{\widehat{\mathcal{S}}_{11}})$, then by running step 10, we have $b_1 = \binom{\sigma_i - 1}{\kappa_i} = \binom{4}{1} = 4$, and there holds $|\mathcal{B}_{11k}| = \sigma_i - \kappa_i = 4$, for $k \in \{1, 2, 3, 4\}$. Specially, $\mathcal{B}_{111} = \{1, 2, 3, 4\}$, $\mathcal{B}_{112} = \{1, 3, 4, 5\}$, $\mathcal{B}_{113} = \{1, 2, 4, 5\}$, and $\mathcal{B}_{114} = \{1, 2, 3, 5\}$. Consequently, $\bigcap_{k=1}^{b_i} \mathcal{H}(x_{\mathcal{B}_{ijk}})$ will be the green area in the top middle figure of Fig. 1. The intersection $\left(\bigcap_{k=1}^{b_i} \mathcal{H}(x_{\mathcal{B}_{ijk}})\right) \bigcap \mathcal{H}(x_{\widehat{\mathcal{S}}_{ij}})$ is nonempty. One can pick any point in this intersection to be $\phi_{ij}$. With the obtained $\phi_{ij}$, the output of Algorithm 1 can be computed as $v_i^\star = \frac{\sum_{j=1}^{s_i} \phi_{ij} + x_i(t)}{s_i + 1}$.

As a further remark, each iteration of Algorithm 1 (from step 5 to step 13) aims to find a vector $\phi_{ij} \in \left(\bigcap_{k=1}^{b_i} \mathcal{H}(x_{\mathcal{B}_{ijk}})\right) \bigcap \mathcal{H}(x_{\widehat{\mathcal{S}}_{ij}})$. As will be shown in the following subsection, the set $\left(\bigcap_{k=1}^{b_i} \mathcal{H}(x_{\mathcal{B}_{ijk}})\right)$ guarantees that the chosen $\phi_{ij}$ is a resilient convex combination defined in definition 1, and the set $\mathcal{H}(x_{\widehat{\mathcal{S}}_{ij}})$ guarantees certain bounds on the values of $\gamma_i$ and $\alpha_i$.

*Proof of Theorem 1:* Ahead of proving Theorem 1, we first present a lemma to summarize some useful results with proofs given in the Appendix.

*Lemma 1:* 1 has the following properties.

a) [Existence] For the step 11 of 1

$$\left(\bigcap_{k=1}^{b_i} \mathcal{H}(x_{\mathcal{B}_{ijk}})\right) \bigcap \mathcal{H}(x_{\widehat{\mathcal{S}}_{ij}}) \neq \emptyset. \tag{9}$$

b) [Resilient combination] For all $j \in \{1, 2, \ldots, s_i\}$, the vector $\phi_{ij}$ is a resilient convex combination such that

$$\phi_{ij} = \sum_{\ell \in \widetilde{\mathcal{S}}_{ij}} \xi_{ij\ell} x_\ell, \quad \sum_{\ell \in \widetilde{\mathcal{S}}_{ij}} \xi_{ij\ell} = 1, \quad \xi_{ij\ell} \geq 0 \tag{10}$$

where $\widetilde{\mathcal{S}}_{ij} \triangleq \mathcal{S}_{ij} \setminus \mathcal{M}_i$. Note that $\widetilde{\mathcal{S}}_{ij}$ precludes any Byzantine agents from the neighbor set $\mathcal{S}_{ij}$.

c) [Weighted combination] For all $j \in \{1, 2, \ldots, s_i\}$, the vector $\phi_{ij}$ can be written as

$$\phi_{ij} = \sum_{\ell \in \widehat{\mathcal{S}}_{ij}} \beta_{ij\ell} x_\ell, \quad \sum_{\ell \in \widehat{\mathcal{S}}_{ij}} \beta_{ij\ell} = 1, \quad \beta_{ij\ell} \geq 0 \tag{11}$$

such that there exists at least one $\ell^* \in \widehat{\mathcal{S}}_{ij}$ with $\beta_{ij\ell^*} \geq \frac{1}{n+1}$. Here, as defined in 1, $\widehat{\mathcal{S}}_{ij} \triangleq \mathcal{S}_{ij} \setminus \{i\}$.

*Remark 2:* Note that Lemma 1(a) establishes the existence of the vector $\phi_{ij}$. Then, (b) and (c) represent this same vector using different combinations of $x_\ell$ to characterize the different properties of $\phi_{ij}$. In (10), the set $\widetilde{\mathcal{S}}_{ij}$ includes agent $i$ and precludes Byzantine agents. This set aims to obtain a resilient convex combination. In contrast, in (11), the set $\widehat{\mathcal{S}}_{ij}$ precludes agent $i$ but may potentially involve the Byzantine agents. This set aims to obtain a convex combination with certain weights that are bounded away from zero for some $\ell \neq i$. Specifically, if $\widehat{\mathcal{S}}_{ij} \subset \widetilde{\mathcal{S}}_{ij}$, then the $\xi_{ij\ell}$ in (10) can correspondingly be replaced by the $\beta_{ij\ell}$ in (11), and leaving the remaining weights for $\widetilde{\mathcal{S}}_{ij} \setminus \widehat{\mathcal{S}}_{ij}$ to be 0. It is worth mentioning that the first intersection of convex hulls in Lemma 1(a) can be considered as generalizations of the safe region in [33, Lemma 3] and the Tverberg point-based region in [34, Thm. 1]. More specifically, [33] and [34] do not use each agent's own state (i.e., $x_i$) to construct a resilient convex combination. Using the notation in this article, Mendes et al. [33] defined a safe region $\bigcap_{k=1}^{b_i} \mathcal{H}(x_{\widehat{\mathcal{B}}_{ijk}})$, where $\widehat{\mathcal{B}}_{ijk} = \mathcal{B}_{ijk} \setminus \{i\}$. It can be observed that $\forall i, j$, $\widehat{\mathcal{B}}_{ijk} \subset \mathcal{B}_{ijk}$, and $\widehat{\mathcal{B}}_{ijk} \subset \widehat{\mathcal{S}}_{ij}$. Thus, this safe region in [33] is a subset of the safe region $\left(\bigcap_{k=1}^{b_i} \mathcal{H}(x_{\mathcal{B}_{ijk}})\right) \bigcap \mathcal{H}(x_{\widehat{\mathcal{S}}_{ij}})$ defined in 1. In addition, it has been proven in [33] that the region in [34] is a subset of its safe region, making it also a subset of the one defined

in this article. Compared with [33] and [34], the relaxation is achieved by taking advantage of the fact that normal agents' can always trust their own states $x_i$. In the following proof of Theorem 1, this difference also leads to a quantifiable bound on $\gamma_i$ and $\alpha_i$. ■

*Proof of Theorem 1:* Recall $v_i^\star = \frac{1}{s_i+1}(\sum_{j=1}^{s_i}\phi_{ij} + x_i)$ and $\forall j \in \{1,2,\ldots,s_i\}$, $\widetilde{\mathcal{S}}_{ij} \subset \mathcal{N}_i$, where $\mathcal{N}_i$ is the set of normal neighbors of agent $i$. Then, based on Lemma 1(b), one has

$$v_i^\star = \sum_{\ell\in\mathcal{N}_i} w_{i\ell}x_\ell, \quad \sum_{\ell\in\mathcal{N}_i} w_{i\ell} = 1, \quad w_{i\ell} \geq 0. \tag{12}$$

Equation (12) means that $v_i^\star$ is a resilient convex combination, and particularly, for the weights $w_{i\ell}$, one has $w_{ii} \geq \frac{1}{s_i+1} > \frac{1}{(s_i+1)(n+1)}$. For $l \neq i$, from (10), one has

$$w_{i\ell} = \frac{1}{s_i+1}\sum_{j\in\{1,2,\ldots,s_i\}} \xi_{ij\ell} \tag{13}$$

where for $\ell \in \widetilde{\mathcal{S}}_{ij}$, $\xi_{ij\ell}$ is the weight defined in (10), and for $\ell \notin \widetilde{\mathcal{S}}_{ij}, \xi_{ij\ell} = 0$. To complete the proof of Theorem 1, we need to further show that for $\ell \neq i$, at least $\gamma_i = d_i - \kappa_i - \sigma_i + 1$ of weights $w_{i\ell}$ are no less than $\alpha_i = \frac{1}{(s_i+1)(n+1)}$. Recall that $\xi_{ij\ell} \geq 0$, and thus, $\forall j \in \{1,\ldots,s_i\}\forall \ell \in \mathcal{N}_i$, one has

$$w_{i\ell} \geq \frac{1}{s_i+1}\xi_{ij\ell}. \tag{14}$$

The key idea is to find the lower bound of $w_{i\ell}$ by determining the lower bound of the corresponding $\xi_{ij\ell}$. To proceed, define a set $\Gamma_i(0) \triangleq \{\mathcal{S}_{ij} \,|\, \mathcal{S}_{ij} \subset \mathcal{N}_i \text{ and } j \in \{1,\ldots,s_i\}\}$. Note that in Algorithm 1, $\mathcal{S}_{ij}$ is originally defined as subsets of $\mathcal{N}_i^+ = \mathcal{N}_i \bigcup \mathcal{M}_i$. Thus, here, $\Gamma_i(0)$ is composed of the sets $\mathcal{S}_{ij}$ by precluding any $\mathcal{S}_{ij}$ that includes at least one Byzantine agent. We have $|\Gamma_i(0)| = \binom{d_i-\kappa_i-1}{\sigma_i-1}$. Now consider the following process.

♣ *For $\tau = 0, 1, 2, \ldots$, if the set $\Gamma_i(\tau)$ is nonempty:*

Pick an arbitrary $\mathcal{S}_{ij^\star}$ from $\Gamma_i(\tau)$. From Lemma 1(c), the corresponding $\phi_{ij^\star}$ can be written as

$$\phi_{ij} = \sum_{\ell\in\widehat{\mathcal{S}}_{ij^\star}} \beta_{ij^\star\ell}x_\ell, \quad \sum_{\ell\in\widehat{\mathcal{S}}_{ij^\star}} \beta_{ij^\star\ell} = 1, \quad \beta_{ij^\star\ell} \geq 0 \tag{15}$$

such that there exists at least one $\ell^* \in \widehat{\mathcal{S}}_{ij^*}$ with $\beta_{ij^\star\ell^*} \geq \frac{1}{n+1}$. Here, since $i \notin \widehat{\mathcal{S}}_{ij^*}$, one has $\ell^* \neq i$. Further note that $\mathcal{S}_{ij^\star} \subset \mathcal{N}_i$, then $\widetilde{\mathcal{S}}_{ij^\star} = \mathcal{S}_{ij^\star} \setminus \mathcal{M}_i = \mathcal{S}_{ij^\star}$. Thus, $\widehat{\mathcal{S}}_{ij^\star} \subset \widetilde{\mathcal{S}}_{ij^\star}$ and according to Remark 2, one can replace the $\xi_{ij^\star\ell}$ in (10) by the $\beta_{ij^\star\ell}$ in (15). Bringing this into (14) leads to, for $\ell^* \neq i$

$$w_{i\ell^*} \geq \frac{\xi_{ij^\star\ell^*}}{s_i+1} = \frac{\beta_{ij^\star\ell^*}}{s_i+1} \geq \frac{1}{(s_i+1)(n+1)}. \tag{16}$$

Now, update $\Gamma_i(\tau+1) = \Gamma_i(\tau) \setminus \{\mathcal{S}_{ij} \,|\, \ell^* \in \mathcal{S}_{ij}\}$, which removes any $\mathcal{S}_{ij}$ composed of agent $\ell^*$. Here, $|\Gamma_i(\tau)| = \binom{d_i-\kappa_i-1-\tau}{\sigma_i-1}$. Note that by doing this, the new $\ell^*$ to be obtained in the new iteration will be different from the ones obtained in previous iterations. Update $\tau$ with $\tau+1$.

*End, and iterate the process from ♣:* Now, consider $\widehat{\tau} = (d_i - \kappa_i - \sigma_i)$, one has $|\Gamma_i(\widehat{\tau})| = \binom{d_i-\kappa_i-1-(d_i-\kappa_i-\sigma_i)}{\sigma_i-1} = 1$. Thus, there is only one $\mathcal{S}_{ij}$ in set $\Gamma_i(\widehat{\tau})$. Clearly, $\Gamma_i(\widehat{\tau}+1) = \emptyset$.

**Algorithm 2:** Compute $\phi_{ij}$ by Linear Programming.

1: **Input** $x_{\mathcal{B}_{ijk}}$, $b_i$ and $x_{\widehat{\mathcal{S}}_{ij}}$.
2: Define variables $\boldsymbol{\mu}_{ij} = \text{col}\,\{\mu_{ijk} | \, k = 1,\ldots,b_i\} \in \mathbb{R}^{b_i|\mathcal{B}_{ijk}|}$ with each $\mu_{ijk} \in \mathbb{R}^{|\mathcal{B}_{ijk}|}$; and $\boldsymbol{\lambda}_{ij} \in \mathbb{R}^{|\widehat{\mathcal{S}}_{ij}|}$.
3: Define matrix $\boldsymbol{X} = \text{row}\,\{x_\ell | \, \ell \in \widehat{\mathcal{S}}_{ij}\} \in \mathbb{R}^{n\times|\widehat{\mathcal{S}}_{ij}|}$. For all $k \in \{1,\ldots,b_i\}$, define matrices $\boldsymbol{X}_k = \text{row}\,\{x_\ell | \, \ell \in \mathcal{B}_{ijk}\} \in \mathbb{R}^{n\times|\mathcal{B}_{ijk}|}$.
4: Set equality constraint $\mathbf{1}^\top\boldsymbol{\lambda}_{ij} = 1$. For all $k \in \{1,\ldots,b_i\}$, set equality constraints $\mathbf{1}^\top\mu_{ijk} = 1$ and $\boldsymbol{X}\boldsymbol{\lambda}_{ij} - \boldsymbol{X}_k\mu_{ijk} = 0$.
5: Set inequality constraints $\boldsymbol{\mu}_{ij} \geq 0$; and $\boldsymbol{\lambda}_{ij} \geq 0$.
6: Choose arbitrary vectors $\eta_1 \in \mathbb{R}^{b_i|\mathcal{B}_{ijk}|}$ and $\eta_2 \in \mathbb{R}^{|\widehat{\mathcal{S}}_{ij}|}$ such that $||\eta_1||_2 = ||\eta_2||_2 = 1$. Define a linear objective function $g(\boldsymbol{\mu}_{ij}, \boldsymbol{\lambda}_{ij}) = \eta_1^\top\boldsymbol{\mu}_{ij} + \eta_2^\top\boldsymbol{\lambda}_{ij}$.
7: Run linear programming with objective function $g(\boldsymbol{\mu}_{ij}, \boldsymbol{\lambda}_{ij})$ subject to the defined equality/inequality constraints.
8: Compute $\phi_{ij} = \sum_{\ell\in\widehat{\mathcal{S}}_{ij}}[\boldsymbol{\lambda}_{ij}]_\ell x_\ell$.

This indicates that the process described in ♣ can be repeated for $d_i - \kappa_i - \sigma_i + 1$ times, from $\tau = 0$ to $\tau = d_i - \kappa_i - \sigma_i$. Consequently, in (12), for $\ell \neq i$, there exist at least $d_i - \kappa_i - \sigma_i + 1$ number of weights $w_{i\ell} \geq \frac{1}{(s_i+1)(n+1)}$. This together with the fact that $w_{ii} \geq \frac{1}{s_i+1}$ yields $\gamma_i = d_i - \kappa_i - \sigma_i + 2$ and completes the proof of Theorem 1. ■

*Remark 3:* In Theorem 1, the values of $\gamma_i$ and $\alpha_i$ can be changed by tuning $\sigma_i$. If a particular $\gamma_i^*$ is desired, one can use equation $\gamma_i = d_i - \kappa_i - \sigma_i + 2$ and the lower bound $\sigma_i \geq n\kappa_i + 2$ to derive a condition $|\mathcal{N}_i| = d_i \geq (n+1)\kappa_i + \gamma_i^*$. Note that for the purpose of achieving constrained consensus, we generally require $\gamma_i^* \geq 2$ (so that agents move their states toward each other). In this case, $d_i \geq (n+1)\kappa_i + 2$ aligns with Assumption 1 in our article. Furthermore, the choice of $\sigma_i$ also influences the computational complexity for calculating $v_i^\star$. We will provide more discussion on this in the next section. ■

### D. Computational Complexity Analysis

We analyze the computational complexity denoted by $C_P$ of Algorithm 1, which is mainly determined by steps 5–13. Among these steps, the major computation comes from the selection of vector $\phi_{ij}$ from the set $(\bigcap_{k=1}^{b_i}\mathcal{H}(x_{\mathcal{B}_{ijk}}))\bigcap\mathcal{H}(x_{\widehat{\mathcal{S}}_{ij}})$ in step 10. To do this, one can directly compute the vertex representation of the set by intersecting convex hulls in the $n$-dimensional space. However, given a large $b_i$, the convex combination may have numerous vertices. To avoid computing all of them, here, we show that a feasible $\phi_{ij}$ can be obtained by solving a linear programming problem, as shown in Algorithm 2.

*Remark 4:* In Algorithm 2, we propose a linear program subject to the constraints of two convex polytopes, namely, $\bigcap_{k=1}^{b_i}\mathcal{H}(x_{\mathcal{B}_{ijk}})$ and $\mathcal{H}(x_{\widehat{\mathcal{S}}_{ij}})$. Note that the equality constraints in line 4 and the inequality constraints in line 5 allow the elements in $\mu_{ijk}$, $k \in \{1,\ldots,b_i\}$, and $\lambda_{ij}$ to specify convex

combinations of $\boldsymbol{X}_k$ and $\boldsymbol{X}$, respectively. Then, the intersection is characterized by $\boldsymbol{X}\lambda_{ij} - \boldsymbol{X}_k\mu_{ijk} = 0$. Since we only need to obtain a feasible point from the intersection of the two polytopes, the objective function $g(\boldsymbol{\mu}_{ij}, \lambda_{ij})$ can be chosen arbitrarily. ■

Now, let $C_L$ denote the computational complexity for this linear program. According to Lee and Sidford [47], one has

$$C_L = \mathcal{O}\left((\text{nz}(W) + d_v^2)\sqrt{d_v}\right)$$

where $d_v$ is the number of variables and $\text{nz}(W)$ is the nonzero entries of a constraint matrix $W$ that encodes both equality and inequality constraints. From Algorithm 2, the definitions of $|\mathcal{B}_{ijk}|$ and $b_i$, we know

$$d_v = b_i|\mathcal{B}_{ijk}| + |\widehat{\mathcal{S}}_{ij}| = \binom{\sigma_i - 1}{\kappa_i}(\sigma_i - \kappa_i) + (\sigma_i - 1) \quad (17)$$

and

$$\text{nz}(W) = 2d_v + nb_i(|\mathcal{B}_{ijk}| + |\widehat{\mathcal{S}}_{ij}|)$$

where $2d_v$ entries are associated with equality and inequality constraints on $\mu_{ijk}$, $k \in \{1, \ldots, b_i\}$, and $\lambda_{ij}$, and $nb_i(|\mathcal{B}_{ijk}| + |\widehat{\mathcal{S}}_{ij}|)$ entries are associated with equality constraints $\boldsymbol{X}\lambda_{ij} - \boldsymbol{X}_k\mu_{ijk} = 0$ for all $k \in \{1, \ldots, b_i\}$. Thus

$$\begin{aligned}\mathcal{C}_L &= \mathcal{O}\left((2d_v + nb_i(|\mathcal{B}_{ijk}| + |\widehat{\mathcal{S}}_{ij}|) + d_v^2)\sqrt{d_v}\right)\\ &= \mathcal{O}\left((2d_v + 2nd_v + d_v^2)\sqrt{d_v}\right)\\ &= \mathcal{O}\left((n + d_v)d_v^{1.5}\right).\end{aligned}$$

The second equation holds because $\sigma_i \geq n\kappa_i + 2$, thus $\mathcal{O}(b_i|\mathcal{B}_{ijk}|) = \mathcal{O}(b_i(\sigma_i - \kappa_i)) = \mathcal{O}(d_v)$ and $\mathcal{O}(b_i|\widehat{\mathcal{S}}_{ij}|) = \mathcal{O}(b_i(\sigma_i - 1)) = \mathcal{O}(b_i(\sigma_i - \kappa_i)) = \mathcal{O}(d_v)$. In addition, since steps 5–13 have to be repeated $s_i = \binom{d_i - 1}{\sigma_i - 1}$ times, one has

$$\mathcal{C}_P = s_i\mathcal{C}_L = \mathcal{O}\left(\binom{d_i - 1}{\sigma_i - 1}(n + d_v)d_v^{1.5}\right). \quad (18)$$

To continue, from Theorem 1, one has $\sigma_i \in [n\kappa_i + 2, d_i - \kappa_i]$ and $\gamma_i = d_i - \kappa_i - \sigma_i + 2$. Thus, choosing $\sigma_i = n\kappa_i + 2$ leads to larger $\gamma_i$. This means the obtained $v_i^\star$ uses more information from its normal neighbors, which can potentially increase the convergence rate of the consensus-based distributed algorithm. As will be shown in the next section, such a choice of $\sigma_i$ will be used to establish a theoretical guarantee on an exponential convergence rate. By taking $\sigma_i = n\kappa_i + 2$, (18) yields

$$\mathcal{C}_P = \mathcal{O}\left(\binom{d_i - 1}{n\kappa_i + 1}(n + d_v)d_v^{1.5}\right) \quad (19)$$

and (17) yields

$$d_v = \binom{n\kappa_i + 1}{\kappa_i}((n-1)\kappa_i + 2) + (n\kappa_i + 1). \quad (20)$$

Clearly, when $n$ and $\kappa_i$ are constants, the complexity $\mathcal{C}_P$ of calculating a resilient convex combination is polynomial in the number of agent's neighbors $d_i$.

*Remark 5:* Since $\mathcal{C}_P$ is exponential in $n$, and the condition in Assumption 1 grows linearly with $n$, our approach is more suitable for problems with low dimensions, such as the resilient multirobot rendezvous problem $n = 2$ or 3 with local constraints. By observing (17) and (18), the complexity $\mathcal{C}_P$ will decrease as $\sigma_i \to (d_i - \kappa_i)$, due to the fast decay of $\binom{d_i - 1}{\sigma_i - 1}$ for $\sigma_i - 1 \geq \frac{d_i - 1}{2}$. However, doing so leads to small $\gamma_i$ and can potentially decrease the convergence speed of the consensus-based algorithm. ■

## IV. Resilient Constrained Consensus

By using the $(\gamma_i, \alpha_i)$-resilient convex combination developed in the previous section, we introduce an approach that allows multiagent systems to achieve resilient constrained consensus in the presence of Byzantine attacks. This distinguishes this article from existing ones that are only applicable to unconstrained consensus problems [26], [27], [30], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41].

### A. Consensus Algorithm Under Byzantine Attack

By recalling Section II-A, consider a multiagent network with $m$ agents characterized by the time-varying network $\mathbb{G}(t)$. In distributed consensus problems, each agent is associated with a local state $x_i \in \mathbb{R}^n$ and a constraint set $\mathcal{X}_i \subseteq \mathbb{R}^n$. Our goal is to find a common point $x^* \in \mathbb{R}^n$ such that

$$x_1 = \cdots = x_m = x^*, \qquad x^* \in \bigcap_{i=1}^{m} \mathcal{X}_i. \quad (21)$$

Based on [48], for constrained consensus problems, where $\exists i \in \{1, \ldots, m\}$, $\mathcal{X}_i \subsetneq \mathbb{R}^n$, the update (5) is effective if there exists a sequence of contiguous uniformly bounded time intervals such that in every interval, $\mathbb{G}(t)$ is strongly connected for at least one time step.[3] In addition, to guarantee an exponential convergence rate, the weights of the spanning graph must be strictly positive and bounded away from 0.

Now, we assume that the network $\mathbb{G}(t)$ suffers a Byzantine attack, as described in Section II-B, where the number of Byzantine neighbors of each normal agent is $\kappa_i(t)$, and $\kappa_i(t) \leq \bar{\kappa}$ $\forall i \in \{1, \ldots, m\}$, $t = 1, 2, \ldots$. The new network is characterized by $\mathbb{G}^+(t)$. To guarantee that the normal agents in $\mathbb{G}^+(t)$ are not influenced by the Byzantine agents, in the following, we introduce a way to incorporate the approach proposed in Algorithm 1 to ensure resilient constrained consensus. Along with this, instead of directly using the result in [48], i.e., relaying on strongly connected graphs, we will introduce concepts of network redundancy and constraint redundancy (cf., Assumption 2) to derive a relaxed topological condition.

### B. Main Result: Resilient Constrained Consensus

For update (5), substitute the $v_i(t)$ with the $v_i^\star(t)$ of Algorithm 1. Then, the vector $v_i^\star(t)$ is a convex combination that automatically removes any information from Byzantine agents; however, as shown in Remark 1, $v_i^\star(t)$ also excludes some information from certain normal neighbors of agent $i$. In other words, substituting the $v_i(t)$ with $v_i^\star(t)$ is equivalent to a classical consensus

[3] A network is called strongly connected if every agent is reachable through a path from every other agent in the network.

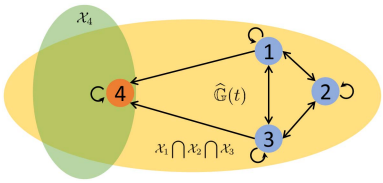



Fig. 2. Resilient subgraph of four agents with $\gamma_i = 3$. Yellow shade denotes the constraint of agents 1–3, green shade denotes the constraint of agent 4.

update on a graph $\widehat{\mathbb{G}}(t)$, where all the Byzantine agents (and some edges from normal agents) are removed. In particular, the network $\widehat{\mathbb{G}}(t)$ is defined as follows.

*Definition 3 (Resilient subgraph):* Define a graph $\widehat{\mathbb{G}}(t)$ with $\mathcal{V}(\widehat{\mathbb{G}}(t)) = \mathcal{V}(\mathbb{G}(t))$ (same agent set).[4] For each time instance $t$, associate the edges of $\widehat{\mathbb{G}}(t)$ with the weights $w^\star_{i\ell}$ $\forall i, \ell \in \{1, \ldots, m\}$ defined in (7), which is the $((\gamma_i, \alpha_i)$-resilient convex combination obtained by Algorithm 1 with parameter $\sigma_i(t) = n\kappa_i(t) + 2$.

Obviously, $\widehat{\mathbb{G}}(t)$ is composed of only normal agents. Furthermore, according to Theorem 1, each agent in $\widehat{\mathbb{G}}(t)$ retains at least $\gamma_i(t) = d_i(t) - \kappa_i(t) - \sigma_i(t) + 2$ of its incoming edges with other normal agents, with weights no less than $\alpha_i(t)$. Thus, by associating the edges of $\widehat{\mathbb{G}}(t)$ with the weights in (7), $\widehat{\mathbb{G}}(t)$ must be an edge-induced subgraph of $\mathbb{G}(t)$. We choose $\sigma_i(t) = n\kappa_i(t) + 2$ in order to maximize the number of retained edges in $\widehat{\mathbb{G}}(t)$. In this case, $\gamma_i(t) = d_i(t) - (n+1)\kappa_i(t)$.

Now, to see whether the consensus update running on $\widehat{\mathbb{G}}(t)$ is able to solve the problem (4), according to Lin and Ren [49], we need to check whether there exists a sequence of contiguous uniformly bounded time intervals such that the union of network $\widehat{\mathbb{G}}(t)$ across each interval is strongly connected. However, since the edge elimination for each agent in $\mathbb{G}(t)$ is a function of the unknown (and arbitrary) actions of Byzantine agents, it can be difficult to draw any conclusion on the strong-connectivity of the union of $\widehat{\mathbb{G}}(t)$, based on the union of $\mathbb{G}(t)$. Actually, even if the $\mathbb{G}(t)$ is fully connected, the obtained $\widehat{\mathbb{G}}(t)$ may still be nonstrongly connected.

Here, we provide a toy example to show why strong-connectivity is critical. Consider a fully connected network $\mathbb{G}(t)$ with $m = 4$ agents, where each agent possesses a local constraint $\mathcal{X}_i$. Suppose for all $i \in \{1, 2, 3, 4\}$, $\kappa_i = 1$ and $n = 1$. Thus, $\sigma_i = n\kappa_i(t) + 2 = 3$, $d_i = m + \kappa_i = 5$, and $\gamma_i = d_i - \kappa_i - \sigma_i + 2 = 5 - 1 - 3 + 2 = 3$. This guarantees that each agent has three neighbors (including itself), i.e., each agent excludes one of its neighbors from a fully connected graph. Then, as shown in Fig. 2, if all agents 1–3 exclude agent 4 from their neighbor sets, then the obtained $\widehat{\mathbb{G}}(t)$ cannot be strongly connected because there is no path to deliver information from agent 4 to agents 1–3. Note that, for the $\widehat{\mathbb{G}}(t)$ in Fig. 2, agents 1–3 are strongly connected, and by running update (5), they are able to reach a consensus within the intersection of $\bigcap_{i=1}^3 \mathcal{X}_i$. However, since agents 1–3 are not able to receive any information from agent 4, they will not move toward agent 4 and, therefore, have no awareness of the constraint $\mathcal{X}_4$. On the other hand for agent 4, even though it can receive information from agents 1 and 3, the existence of constraint $\mathcal{X}_4$ prevents agent 4 from moving toward agents 1–3; thus, a consensus cannot be reached, even though there are no Byzantine agents in the network.

As validated by the example in Fig. 2, if one directly replaces the $v_i(t)$ in (5) with the $v_i^\star$ in Algorithm 1, the underlying network $\widehat{\mathbb{G}}(t)$ may not be sufficient for the agents in the network to reach resilient constrained consensus. This phenomenon necessitates the following assumption, where we introduce redundancies for both the network and the constraint sets.

*Assumption 2 (Redundancy conditions):* We assume that there exists $\varphi \in \mathbb{Z}_+$ such that the following conditions hold.

a) *[Network Redundancy]:* By Theorem 1, one has $\alpha_i(t) = \frac{1}{(s_i(t)+1)(n+1)}$. Define $\underline{\alpha} = \min\{\alpha_i(t)\}$ $\forall t \in \{1, 2, \ldots\}$ and $\forall i \in \{1, 2, \ldots, m\}$. Such $\underline{\alpha} \in \mathbb{R}_+$ must exist and is bounded away from zero due to the boundedness of $s_i(t)$. Now, consider the network $\widehat{\mathbb{G}}(t)$ defined in Definition 3. Suppose $\forall t = 1, 2, \ldots$, $\widehat{\mathbb{G}}(t)$ is spanned by a rooted directed graph[5] $\mathbb{N}(t)$, where $\mathcal{V}(\mathbb{N}(t)) = \mathcal{V}(\widehat{\mathbb{G}}(t))$ and $\mathbb{N}(t)$ has the edges of $\widehat{\mathbb{G}}(t)$ with weights no less than $\underline{\alpha}$. Furthermore, let $\mathbb{S}(t)$ be the root strongly connected component of $\mathbb{N}(t)$, and suppose there exists an infinite sequence of contiguous uniformly bounded time intervals such that in every interval, $|\mathcal{V}(\mathbb{S}(t))| \geq \varphi$ for at least one time step.

b) *[Constraints Redundancy]:* Let $\mathcal{V} = \{1, \ldots, m\}$ denote the agent set of network $\mathbb{G}(t)$. Suppose that for all $\bar{\mathcal{V}} \subset \mathcal{V}$ with $|\bar{\mathcal{V}}| \geq \varphi$, there holds

$$\bigcap_{i \in \bar{\mathcal{V}}} \mathcal{X}_i = \bigcap_{i=1}^{m} \mathcal{X}_i. \tag{22}$$

The proposed assumption leads to the following result.

*Theorem 2. (Resilient constrained consensus):* Consider a network $\mathbb{G}(t)$ of normal agents. Suppose $\mathbb{G}(t)$ experiences a Byzantine attack such that each normal agent in $\mathbb{G}(t)$ has $\kappa_i(t)$ Byzantine neighbors. Suppose Assumptions 2 (a) and (b) hold. By replacing the $v_i(t)$ in (5) with the $v_i^\star$ in 1 and setting $\sigma_i(t) = n\kappa_i(t) + 2$, the update (5) will drive the states of all agents in $\mathbb{G}(t)$ to converge exponentially fast to a common state, which satisfies the constrained consensus (4). Note that if $\kappa_i(t)$ is not known by each agent, then it can be replaced by an upper bound $\bar{\kappa}$.

*Proof of Theorem 2:* Here, we prove Theorem 2 by referring to some existing results in [49], [50], [51], [52], and [53]. Let $\widehat{W}(t)$ be the (row stochastic) weight matrix corresponding to the network $\widehat{\mathbb{G}}(t)$. Define matrix $\widehat{U}(T)$ as follows:

$$\widehat{U}(T) \triangleq \widehat{W}(T) \cdot \widehat{W}(T-1) \cdots \widehat{W}(1) \cdot \widehat{W}(0). \tag{23}$$

Since all $\widehat{W}(t)$ are row stochastic and all $\widehat{\mathbb{G}}(t)$ are spanned by a rooted directed graph with edge weights no less than $\underline{\alpha}$ [see Assumption 2(a)], then according to the authors in[50],

[4] To clarify, $\mathbb{G}(t)$ is the graph without Byzantine agents.

[5] A spanning rooted graph contains a least one agent that can reach every other agent in the network through a path.

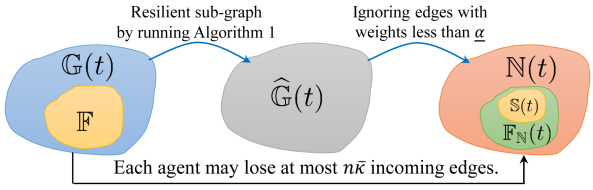



Fig. 3. Demonstration of $\mathbb{G}(t)$, $\widehat{\mathbb{G}}(t)$, $\mathbb{N}(t)$, $\mathbb{F}$, and $\mathbb{F}_{\mathbb{N}}(t)$, where $\mathbb{G}(t)$, $\widehat{\mathbb{G}}(t)$, and $\mathbb{N}(t)$ share the same agent set. $\mathbb{F}$ and $\mathbb{F}_{\mathbb{N}}(t)$ share the same agent set.

[51], and [52], as $T \to \infty$, the $\widehat{U}(T)$ in (23) must converge exponentially fast to a constant matrix $\widehat{U}^*$ such that all rows of $\widehat{U}^*$ are identical, i.e., $\widehat{U}^* = \mathbf{1}_m \widehat{u}^*$, $\widehat{u}^* \in \mathbb{R}^{1\times m}$. In addition, from Assumption 2(a), $\widehat{\mathbb{G}}(t)$ contains a component $\mathbb{S}(t)$ with at least $\varphi$ agents [containing the root of $\mathbb{N}(t)$], and for $t = 1, 2, \ldots$, $\mathbb{S}(t)$ is strongly connected infinitely often. Furthermore, since $\mathbb{N}(t)$ has a finite number of agents, by Infinite Pigeonhole principle [54], there must exist a fixed set of $\varphi$ agents [containing the root of $\mathbb{N}(t)$] such that the graph induced by these agents is strongly connected infinitely often. Corresponding to this set of agents, at least $\varphi$ columns of $\widehat{U}^*$ are strictly positive [55]. Furthermore, since $\widehat{U}^* = \mathbf{1}_m \widehat{u}^*$, at least $\varphi$ entries of $\widehat{u}^*$ must be strictly positive. For $i \in \{1, \ldots, m\}$, define

$$e_i(t) = [\mathcal{P}_i(v_i(t)) - v_i(t)]. \tag{24}$$

Accordingly, update (5) can be rewritten as

$$x_i(t+1) = v_i(t) + e_i(t) \tag{25}$$

where $v_i(t) = v_i^\star = \sum_{\ell\in\widehat{\mathcal{N}}_i} w_{i\ell}(t) x_\ell(t)$ and the $w_{i\ell}(t)$ is the $i,\ell$th entry of the adjacency matrix $\widehat{W}(t)$.

To proceed, recall our above proof that the $\widehat{U}(T)$ defined in (23) converges exponentially fast to the matrix $\mathbf{1}_m \widehat{u}^*$, and the entries of $\widehat{u}^*$ associated with $\mathcal{V}(\mathbb{S}(t))$ must be strictly positive. Then, by utilizing [49, Lemma 8],[6] for $i \in \mathcal{I}$, there holds $\|e_i(t)\| \to 0$ as $t \to \infty$. This further implies that for $i \in \mathcal{V}(\mathbb{S}(t))$, the states $x_i(t)$ will reach a consensus at a point $x^*$, such that $\mathcal{P}_i(x^*) = x^*$, $i \in \mathcal{I}$. Finally, by Assumption 2(b), since $|\mathcal{V}(\mathbb{S}(t))| \geq \varphi$, one has $x^* \in \bigcap_{i\in\mathcal{V}(\mathbb{S}(t))} \mathcal{X}_i = \bigcap_{i=1}^m \mathcal{X}_i$.

Now, to complete the proof of Theorem 2, we only need to show that for agents $j \notin \mathcal{V}(\mathbb{S}(t))$, the states $x_j(t)$ also converge to $x^*$. Note that $x^* \in \bigcap_{i=1}^m \mathcal{X}_i \subset \bigcap_{j\notin\mathcal{V}(\mathbb{S}(t))} \mathcal{X}_j$. Thus, for agents $j \notin \mathcal{V}(\mathbb{S}(t))$, their dynamics are dominated by a leader–follower consensus process within the domain of $\bigcap_{j\notin\mathcal{I}} \mathcal{X}_j$, where any agent in $\mathcal{V}(\mathbb{S}(t))$ can be considered as the leader. Since from Assumption 2(a), $\mathbb{N}(t)$ is a rooted graph and the root is in $\mathcal{V}(\mathbb{S}(t))$, one has $x_j(t)$, $j \notin \mathcal{V}(\mathbb{S}(t))$, also converge exponentially to $x^*$ [53]. This completes the proof. ■

*Remark 6: (Justification of Assumption 2):* While the example in Fig. 2 explains why Assumption 2 is required, the above proof demonstrates that this assumption is sufficient for using Algorithm 1 to achieve resilient constraint consensus. Specifically, it ensures normal agents use a sufficient amount of other normal agents' information to update their states. Without the network redundancy, the states in all agents may not reach a consensus; without the constraint redundancy, the consensus point may not satisfy the constraints in all agents.

Obviously, for Assumption 2(b), one can guarantee that (22) holds by introducing overlaps on the local constraint $\mathcal{X}_i$ of each agent. However, for Assumption 2(a), since $\widehat{\mathbb{G}}(t)$ is an edge-induced subgraph of $\mathbb{G}(t)$, it not straightforward to see what $\mathbb{G}(t)$ can lead to a $\widehat{\mathbb{G}}(t)$ that satisfies the network redundancy conditions in Assumption 2(a). Motivated by this, in the following, we propose a sufficient condition for network redundancy. Based on this, one can easily design a network $\mathbb{G}(t)$, which always guarantees that Assumption 2(a) holds for certain $\varphi$. ■

*Definition 4: (r-reachable set [26]):* Given a directed graph $\mathbb{G}$ and a nonempty subset $\mathcal{A}_s$ of its agents $\mathcal{V}(\mathbb{G})$, we say $\mathcal{A}_s$ is an r-reachable set if $\exists i \in \mathcal{A}_s$ such that $|\mathcal{N}_i \setminus \mathcal{A}_s| \geq r$.

*Corollary 1: (Sufficient condition for network redundancy):* Suppose for all $t = 1, 2, \ldots$, the network $\mathbb{G}(t)$ has a fully connected subgraph[7] $\mathbb{F}$ (time-invariant) whose number of agents satisfies $|\mathcal{V}(\mathbb{F})| = f \geq 2n\bar{\kappa} + 1$. Furthermore, we assume that any subset of $\mathcal{V}(\mathbb{G}(t)) \setminus \mathcal{V}(\mathbb{F})$ (set of agents that are in $\mathbb{G}(t)$ but not in $\mathbb{F}$) is $(n\bar{\kappa}+1)$-reachable in $\mathbb{G}(t)$. Now, if the network $\mathbb{G}(t)$ is under Byzantine attack and each agent adds $\kappa_i(t)$ Byzantine neighbors in each time step, such that $\kappa_i(t) \leq \bar{\kappa}$, we let each agent in the system run Algorithm 1 by choosing $\sigma_i(t) = n\bar{\kappa}_i + 2$. Then, Assumption 2(a) must hold for $\varphi = f - n\bar{\kappa}$.

*Proof:* Recall from Definition 3 that $\widehat{\mathbb{G}}(t)$ is the network associated with the weight coefficients of (7) obtained in Algorithm 1. According to Theorem 1, each agent $i$ of $\widehat{\mathbb{G}}(t)$ must have at least $\gamma_i(t) = d_i(t) - \kappa_i(t) - \sigma_i(t) + 2$ neighbors (including itself) with weights no less than $\alpha_i(t) = \frac{1}{(s_i(t)+1)(n+1)}$. Here, $d_i(t) = m_i(t) + \kappa_i(t)$ with $d_i(t) = |\mathcal{N}_i^+(t)|$ and $m_i(t) = |\mathcal{N}_i(t)|$. Thus, we can easily derive $\gamma_i(t) = m_i(t) - n\kappa_i(t) \geq m_i(t) - n\bar{\kappa}$. To continue, let $\underline{\alpha} = \min\{\alpha_i(t) \,|\, i = 1, \ldots, m;\, t = 1, 2 \ldots\}$. Followed by Assumption 2(a), for each time step, we define the edge-induced subgraph $\mathbb{N}(t)$ of $\widehat{\mathbb{G}}(t)$, which removes all the edges of $\widehat{\mathbb{G}}(t)$ with weights less than $\underline{\alpha}$. Since $\alpha_i(t) \geq \underline{\alpha}$ and $\gamma_i(t) \geq m_i(t) - n\bar{\kappa}$, each agent in $\mathbb{N}(t)$ loses at most $n\bar{\kappa}$ incoming edges compared with that of the original graph $\mathbb{G}(t)$.

We start by proving the second statement of Assumption 2(a). This will be done by ensuring a sufficient condition such that for all $t$, a subgraph of $\mathbb{N}(t)$ always contains a strongly connected component $\mathbb{S}(t)$ with at least $\varphi = f - n\bar{\kappa}$ agents. Since $\mathbb{G}(t)$ has a fully connected component $\mathbb{F}$, let $\mathbb{F}_{\mathbb{N}}(t)$ denote the agent-induced subgraph of $\mathbb{N}(t)$ such that $\mathcal{V}(\mathbb{F}_{\mathbb{N}}(t)) = \mathcal{V}(\mathbb{F})$. Then, $|\mathcal{V}(\mathbb{F}_{\mathbb{N}}(t))| = |\mathcal{V}(\mathbb{F})| = f$, and each agent in $\mathbb{F}_{\mathbb{N}}(t)$ loses at most $n\bar{\kappa}$ incoming edges from $\mathbb{F}$ (this property follows from the relation between $\mathbb{G}(t)$ and $\mathbb{N}(t)$, as visualized in Fig. 3). As a consequence, every agent in $\mathbb{F}_{\mathbb{N}}(t)$ has at least $f - n\bar{\kappa} = \varphi$ incoming edges with the agents in $\mathbb{F}_{\mathbb{N}}(t)$ (including self-loops). Now, for

[6] The main result in [49] considers the convergence of states to the intersection of constraints in all agents, which requires *all* entries in $\widehat{u}^*$ to be strictly positive. Here, we utilize [49, Lemma 8] to prove the convergences of states to the intersection of constraints possessed by agents in $\mathbb{S}(t)$, since only the corresponding entries in $\widehat{u}^*$ are strictly positive.

[7] A fully connected component is a complete graph, i.e., all agents in the component have incoming edges from all other agents in that component.

time step $t$, suppose $\mathbb{F}_{\mathbb{N}}(t)$ has $N(t)$ maximal strongly connected components (disjoint), denoted by $\mathbb{S}_j(t)$, $j \in \{1, \ldots, N(t)\}$. Clearly, there exists at least one root strongly connected component $\mathbb{S}(t) \in \{\mathbb{S}_j(t)\}$ that does not have incoming edges from other components in $\mathbb{F}_{\mathbb{N}}(t)$ [56] because, otherwise, there will be loops among $\mathbb{S}_j(t)$, leading to a larger component, which contradicts with its definition. Based on this, recall the fact that every agent in $\mathbb{F}_{\mathbb{N}}(t)$, as well as every agent in $\mathbb{S}(t)$, has at least $f - n\bar{\kappa} = \varphi$ incoming edges from the agents in $\mathbb{F}_{\mathbb{N}}(t)$. Thus, all incoming edges of agents in $\mathbb{S}(t)$ must come from $\mathbb{S}(t)$ itself (including self-loops). Consequently, $|\mathcal{V}(\mathbb{S}(t))| \geq \varphi$.

In the following, we establish the remaining statements of 2(a), namely, $\mathbb{N}(t)$ is a rooted (connected) graph and one root of the graph is in $\mathbb{S}(t)$. We start by showing that $\mathbb{F}_{\mathbb{N}}(t)$ is a rooted graph. Note that from the above derivation, $\mathbb{S}(t)$ is a strongly connected component with $|\mathcal{V}(\mathbb{S}(t))| \geq \varphi = f - n\bar{\kappa}$. Thus, $|\mathcal{V}(\mathbb{F}_{\mathbb{N}}(t))| - |\mathcal{V}(\mathbb{S}(t))| \leq n\bar{\kappa}$, meaning that all agents in $\mathbb{F}_{\mathbb{N}}(t)$ but not in $\mathbb{S}(t)$ must have at least one incoming edge from $\mathbb{S}(t)$. Therefore, $\mathbb{F}_{\mathbb{N}}(t)$ is a rooted graph and one root of the graph is in $\mathbb{S}(t)$. Now, we show that $\mathbb{N}(t)$ is a rooted graph and one of its roots is in $\mathbb{S}(t)$. Recall that any subset of agents in $\mathcal{V}(\mathbb{G}(t)) \setminus \mathcal{V}(\mathbb{F})$ is $(n\bar{\kappa}+1)$-reachable in $\mathbb{G}(t)$. Since each agent in $\mathbb{N}(t)$ loses at most $n\bar{\kappa}$ incoming edges from $\mathbb{G}(t)$, any subset of $\mathcal{V}(\mathbb{N}(t)) \setminus \mathcal{V}(\mathbb{F}_{\mathbb{N}}(t))$ is 1-reachable in $\mathbb{N}(t)$. Let us decompose $\mathbb{N}(t)$ into its strongly connected components. If $\mathbb{N}(t)$ is not a rooted graph, then there must exist at least two components, denoted by $\mathbb{P}_1$ and $\mathbb{P}_2$, that have no incoming edges from any other components. Since $\mathbb{F}_{\mathbb{N}}(t)$ is a rooted graph, any agents in $\mathbb{F}_{\mathbb{N}}(t)$ can only be associated with one of such root component, say $\mathbb{P}_1$. Then, $\mathbb{P}_2$ must be composed of agents only in $\mathcal{V}(\mathbb{N}(t)) \setminus \mathcal{V}(\mathbb{F}_{\mathbb{N}}(t))$. This, however, contradicts with the fact that any subset of $\mathcal{V}(\mathbb{N}(t)) \setminus \mathcal{V}(\mathbb{F}_{\mathbb{N}}(t))$ is 1-reachable in $\mathbb{N}(t)$ (every subset must has a neighbor outside the set). Thus, $\mathbb{N}(t)$ must be a rooted graph. Further, since there can be no root components in $\mathcal{V}(\mathbb{N}(t)) \setminus \mathcal{V}(\mathbb{F}_{\mathbb{N}}(t))$, the root component of $\mathbb{N}(t)$ must be in $\mathbb{F}_{\mathbb{N}}(t)$, which is then rooted by agents in $\mathbb{S}(t)$. This completes the proof.

Note that if Assumption 2(a) holds for $\varphi = f - n\bar{\kappa}$, then it also holds for any smaller $\varphi$.

### C. Special Case: Resilient Unconstrained Consensus

Note that if the agents in the system are not subject to local constraints, i.e., $\mathcal{X}_i = \mathbb{R}^n \, \forall i = 1, \ldots, m$ and $\mathcal{P}_i(v_i(t)) = v_i(t)$, then update (5) degrades to

$$x_i(t+1) = v_i(t). \tag{26}$$

As pointed in [48], for unconstrained consensus, a sufficient condition for the effectiveness of update (26) is the existence of a uniformly bounded sequence of contiguous time intervals such that in every interval, $\widehat{\mathbb{G}}(t)$ has a rooted directed spanning tree for at least one time step. Obviously, this connectivity condition is weaker than the one proposed in Section IV-A for the constrained consensus case. Correspondingly, we introduce a weaker version of Assumption 2 as follows.

*Definition 5: (r-robustness [26]):* A directed graph $\mathbb{G}$ is $r$-robust, with $r \in \mathbb{Z}_{\geq 0}$, if for every pair of nonempty, disjoint subsets of $\mathbb{G}$, at least one of the subsets is $r$-reachable.

*Assumption 3:* Consider a network $\mathbb{G}(t)$. Suppose $\mathbb{G}(t)$ is $(n\bar{\kappa}+1)$-robust.

This assumption leads to the following lemma, which can be obtained by [26, Lemma 7] or by combining the [34, Lemmas 3 and 4].

*Lemma 2:* For any network $\mathbb{G}(t)$ satisfying Assumption 3, if each agent in $\mathbb{G}(t)$ removes up to $n\bar{\kappa}$ incoming edges, then the obtained edge-induced network is still spanned by a rooted directed tree.

Consequently, one has the following.

*Theorem 3:* Consider a network $\mathbb{G}(t)$ of normal agents. For $\mathbb{G}(t)$, there exists a sequence of uniformly bounded contiguous time intervals such that in each interval, at least one $\mathbb{G}(t)$ satisfies Assumption 3. Suppose $\mathbb{G}(t)$ is affected by a Byzantine attack such that each normal agent in $\mathbb{G}(t)$ has at most $\bar{\kappa}$ Byzantine neighbors. Then, by replacing the $v_i(t)$ in (26) with the $v_i^\star$ in Algorithm 1 with $\sigma_i(t) = n\kappa_i(t) + 2$, update (26) will drive the states in all agents of $\mathbb{G}(t)$ to reach a consensus exponentially fast.

The proof of this theorem can directly be obtained by combining the Theorem 1, Lemma 2 in this article, and the result in [48].

*Remark 7:* Note that the results in this section can be considered as the special case of Section IV-B. Particular, compared with Assumption 2(a), unconstrained consensus no longer requires a root strongly connected component such that $|\mathcal{V}(\mathbb{S}(t))| \geq \varphi$, but only $|\mathcal{V}(\mathbb{S}(t))| = 1$, which reduced to a rooted spanning tree. Such condition can be guaranteed by $(n\bar{\kappa}+1)$-robustness in Assumption 3 based on Lemma 2. The constraints redundancy Assumption 2(b) is completely removed due to the nonexistence of local constraints. In addition, compared with existing results, Theorem 3 applies to the resilient unconstrained consensus for multidimensional state vectors under time-varying networks. If the network is time-invariant, then our result matches Theorem 1 (under Condition SC) proposed in [34] and Theorem 5.1 (under Assumption 5.2) proposed in [35]. If the agents' states are scalars, then the result matches [26, Thm. 2] and [19, Thm. 6.1]. ■

## V. Simulation and Application Examples

In this section, we validate the effectiveness of the proposed algorithm via both numerical simulations and an application example of safe multiagent learning.

### A. Resilient Consensus in Multiagent Systems

For resilient consensus, consider the following update:

$$x_i(t+1) = \mathcal{P}_i(v_i^\star(t)) \tag{27}$$

where $v_i^\star(t)$ is a resilient convex combination that can be obtained from Algorithm 1 with $x_{\mathcal{N}_i^+}(t)$; $\mathcal{P}_i(\cdot)$ is a projection operator that projects any state to the local constraint $\mathcal{X}_i$. If the constraint does not exist, then $\mathcal{P}_i(v_i^\star) = v_i^\star$. Here, we validate resilient consensus problems for both unconstrained and constrained cases.

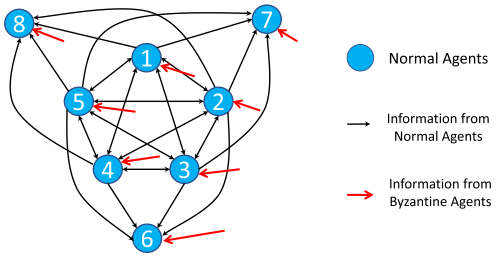



Fig. 4. Network of eight agents, under Byzantine attack.

*Unconstrained consensus:* Consider a multiagent system composed of eight normal agents. Let $n = 2$ and $\bar{\kappa} = 1$, which means that each agent holds a state $x_i \in \mathcal{R}^2$ and has at most one Byzantine neighbor. Suppose the communication network of the system is time-varying, but every three time steps, at least one graph satisfies the connectivity condition described in Assumption 3. Fig. 4 is an example of the network at a certain time step, which is constructed by the *preferential-attachment* method [57]. Note that in Fig. 4, we do not draw concrete Byzantine agents as these agents can send different misinformation to different normal agents. Specifically, the red arrows can be considered either different information for one Byzantine agent or different information for different Byzantine agents.

From Fig. 4, each agent has $d_i = 6$ (including the edge from Byzantine agents and the agent's self-loop). According to Theorem 1, by taking $\sigma_i = n\kappa_i + 2$, each normal agent can locally compute a $(\gamma_i, \alpha_i)$-resilient convex combination such that for all $i = 1, \ldots, 8$, there holds $\gamma_i = d_i - \kappa_i - \sigma_i + 2 = 3$. Note that the network in Fig. 4 is constructed based on the condition described in Assumption 3. Although Assumption 2(a) is not required for unconstrained consensus, it is worth mentioning that Fig. 4 also satisfies the condition described in Corollary 1 with a fully connected component of five agents. This leads to the hold of Assumption 2(a). Specifically, if each agent in Fig. 4 keeps at least three edges from their normal neighbors and removes the other edges, the obtained subgraph is spanned by a rooted directed tree.

Initialize the normal agents' states $x_i(0)$ by vectors randomly chosen from $[-2, 2] \times [-2, 2]$. At each time step, let Byzantine agents send states to normal agents, which are randomly chosen from $[-4, 4] \times [-5, 5]$. By running update (27), one has the following simulation result. Since the solutions for consensus problems are nonunique, in the figure, $V(t) \triangleq \sum_{i=1}^{m} \sum_{j=1}^{m} \|x_i(t) - x_j(t)\|_2^2$ measures the closeness among all the normal neighbors. In particular, $V(t) = 0$ if and only if a consensus is reached. It can be seen that compared with traditional nonresilient consensus update (5), where the states of the normal agents will be misled by their Byzantine neighbors, the proposed approach allows all the normal agents to achieve a resilient consensus. In addition, we note that the topology in Fig. 4 also satisfies the network condition in [33]. The algorithm in [33] achieves resilient consensus with a slightly different convergence rate. The improved convergence rate of our algorithm might be attributed to the special use of agents' own states, as we have discussed in Remark 2.

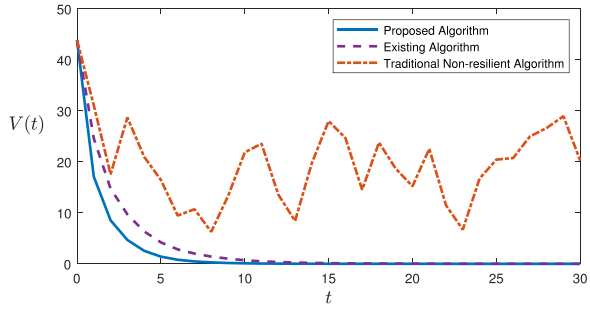



Fig. 5. Algorithm proposed in this article versus the result of [33] for unconstrained consensus versus tradition nonresilient consensus algorithm.

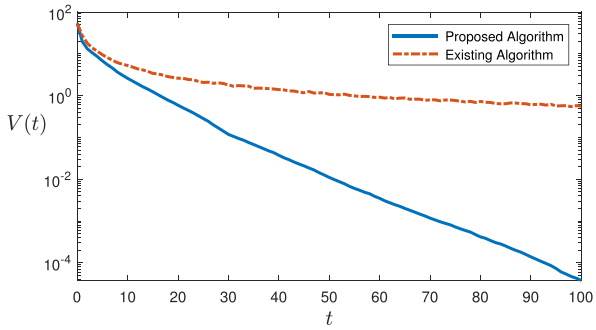



Fig. 6. Algorithm proposed in this article versus the result of [41] for constrained consensus.

*Constrained consensus:* It has been shown in Section IV-B that a resilient constrained consensus requires both network and constraint redundancy. In this simulation, we employ a network $\mathbb{G}(t)$ of $m = 30$ agents. Let $n = 2$. Suppose $\mathbb{G}(t)$ is affected by Byzantine attack such that each normal agent in $\mathbb{G}(t)$ has at most $\bar{\kappa} = 2$ Byzantine neighbors. By choosing $\varphi = 11$, we construct the network such that $\mathbb{G}(t)$ has a fully connected component of $f = 15$ agents, and any other agents of $\mathbb{G}(t)$ have at least six neighbors in the fully connected component. This setup guarantees all conditions in Corollary 1 hold, i.e., $\varphi = f - n\bar{\kappa}$ and $f \geq 2n\bar{\kappa} + 1$. Thus, the network redundancy in Assumption 2(a) must be satisfied. To continue, for all the agents, define their local constraints as follows:

$$\mathcal{X}_i = \{x | \, \|x\|_2 \leq 1, \, l_1 \cdot x \geq 0\} \text{ for } i = 1, \ldots, 10$$
$$\mathcal{X}_i = \{x | \, l_1 \cdot x \geq 0, \, l_2 \cdot x \geq 0\} \text{ for } i = 11, \ldots, 20$$
$$\mathcal{X}_i = \{x | \, l_2 \cdot x \geq 0, \, \|x\|_2 \leq 1\} \text{ for } i = 21, \ldots, 30$$

where $l_1 = [1 \; 1]$, $l_2 = [1 \; 0]$, and $\cdot$ is the inner product. Clearly, the intersection of constraints of any 11 agents equals to that of all the agents. Thus, the constraint redundancy in Assumption 2(b) is also satisfied for $\varphi = 11$.

Under the above setup, the simulation result for this example is shown in Fig. 6. Particularly, one has $x_i = x^*$ for all $i = 1, \ldots, 30$, where $x^* = [0.7071 \; 0.7071]^\top$. Such $x^*$ satisfies all the local constraints. Furthermore, by the $\log(\cdot)$ scale on $Y$-axis, one can observe that compared with the result of [41], the approach proposed in this article has an exponential convergence rate.

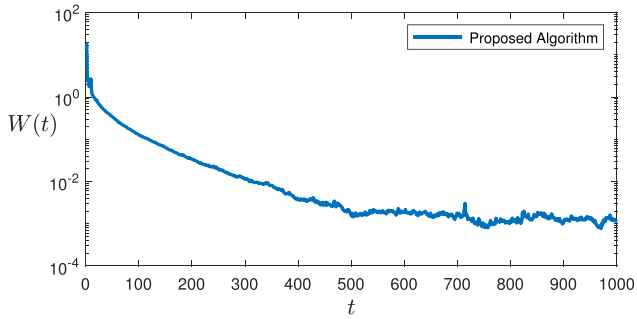



Fig. 7. Multiagent machine learning based on the resilient constrained consensus approach proposed in this article.

### B. Safe Multiagent Learning Based on Resilient Consensus

This section provides an example by employing the resilient convex combination into an application of safe multiagent learning. Consensus-based multiagent learning algorithms [58] allow multiple agents to cooperatively handle complex machine learning tasks in a distributed manner. By combining the idea of consensus and machine learning techniques, the system can synthesize agents' local data samples in order to achieve a globally optimal model. However, similar to other consensus-based algorithms, multiagent learning algorithms are also vulnerable to malicious attacks. These attacks can either be caused by the presence of Byzantine agents or normal functioning agents with corrupted data.

Here, we consider a multiagent value-function approximation problem, which is a fundamental component in both supervised learning algorithms [59] (e.g., Regression and SVM) and reinforcement learning algorithms [60] (e.g., deep $Q$-learning, actor–critic, proximal policy optimization). Suppose we have the following value function $Q_\theta(z)$:

$$Q_\theta(z) = \psi(z)^\top \theta \tag{28}$$

where $\theta \in \mathbb{R}^n$ is the parameter vector, $\psi(\cdot) \in \mathbb{R}^n$ is the feature vector, and $z$ is the model input, which can be the labeled data in supervised learning, or the state action pairs in reinforcement learning. We let $n = 10$. Under the multiagent setup, given the eight-agent network, as shown in Fig. 4, with $\bar{\kappa} = 1$, suppose each agent $i$ has local data sample pairs $(Q_{i,k}, z_{i,k})$, $k = 1, 2, \ldots$, which are subject to measurement noises. The goal is to find the optimal $\theta^\star$ parameter vector such that

$$\min_\theta \sum_i \sum_k \|Q_\theta(z_{i,k}) - Q_{i,k}\|^2, \quad \text{s.t.} \|\theta\|_2 \le \rho \tag{29}$$

where the regularization constraint avoids overfitting [61].

During the communication process, malicious parameter information is received by the normal agents through the red arrows in Fig. 4. Thus, similar to Fig. 5, traditional multiagent learning algorithms cannot solve (29) in a resilient way. Motivated by this, we introduce the following algorithm:

$$\theta_i(t+1) = \mathcal{P}_i\left(v_i^\star(t) - \eta(t) g_{i,k}(\theta_i(t))\right) \tag{30}$$

that replace the consensus vector in traditional multiagent learning algorithms [44] with a resilient convex combination developed in this article. Particularly, in (30), $v_i^\star(t)$ is the resilient convex combination of $\theta_i(t)$ computed by 1 with $\sigma_i = n\kappa_i + 2$; $g_{i,k}(\theta_i(t))$ is the stochastic gradient corresponding to a random sample pair $(Q_{i,k}, z_{i,k})$ of agent $i$

$$\begin{aligned} g_{i,k}(\theta_i(t)) &= \nabla_{\theta_i} \|Q_{\theta_i}(z_{i,k}) - Q_{i,k}\|^2 \\ &= \left(Q_{\theta_i(t)}(z_{i,k}) - Q_{i,k}\right) \psi(z_{i,k}). \end{aligned}$$

The projection operator $\mathcal{P}_i(\theta)$ ensures the satisfactory of regularization constraints

$$\mathcal{P}_i(\theta) = \underset{\tilde{\theta}:\ \|\tilde{\theta}\| \le \rho}{\arg\min} \|\tilde{\theta} - \theta\|$$

and $\eta(t) = \frac{1}{t}$ is a diminishing learning rate that is commonly used in stochastic gradient descent [44]. By defining

$$W(t) \triangleq \sum_{i=1}^m \|\theta_i(t) - \theta^*\|_2^2$$

and running update (30), one obtains the result shown in (7). One can observe in Fig. 7 that in the presence of Byzantine attacks, the system can perform safe multiagent learning, i.e., the $\theta_i(t)$ for all normal agents converge to the desired parameter $\theta^*$.

## VI. Conclusion

This article aims to achieve resilient constrained consensus for multiagent systems in the presence of Byzantine attacks. We started by introducing the concept of $(\gamma_i, \alpha_i)$-resilient convex combination and proposed a new approach (see Algorithm 1) to compute it. We showed that the normal agents in a multiagent system can use their local information to automatically isolate the impact of their Byzantine neighbors. Given a fixed-state dimension and an upper bound for agents' Byzantine neighbors, the computational complexity of the algorithm is polynomial in the network degree of each agent. Under sufficient redundancy conditions on the network topology [see Assumption 2(a)] and constraints [see Assumption 2(b)], the proposed algorithm can be employed to achieve resilient constrained distributed consensus with exponential convergence rate. Since the network redundancy condition [see Assumption 2(a)] is difficult to directly verify, we proposed a sufficient condition (see Corollary 1), which offers an easy way to design a network satisfying the network redundancy condition. We have validated the correctness of all the proposed results by theoretical analysis. We also illustrated the effectiveness of the proposed algorithms by numerical simulations, with applications in unconstrained consensus, constrained consensus, and multiagent safe multiagent learning. Our future work includes further simplifying the computational complexity of the proposed algorithm, studying the tradeoff between resilience and computational complexity, and integrating the proposed method with other resilience approaches, such as anomaly detection, to better mitigate Byzantine attacks.

## APPENDIX

### *Proof of Lemma 1*

In order to prove Lemma 1, we first introduce the following lemmas.

*Lemma 3:* If $x_i \notin \mathcal{H}(x_{\widehat{\mathcal{S}}_{ij}})$, then

$$\bigcap_{k\in\Omega(n)} \mathcal{H}(x_{\mathcal{B}_{ijk}}) \setminus \{x_i\} \neq \emptyset \tag{31}$$

where $\Omega(n)$ is any subset of $\{1,\dots,b_i\}$ with $|\Omega(n)| = n$.

*Proof:* By definition, $\forall k \in \{1,\dots,b_i\}$, one has $x_i \in x_{\mathcal{B}_{ijk}}$. Thus, (31) holds if there exists another common state, say $x_\ell \neq x_i$, such that $x_\ell \in x_{\mathcal{B}_{ijk}}\ \forall k \in \Omega(n)$. In the following, we show that such $x_\ell$ always exists by contradiction.

Suppose $x_i$ is the only state shared by sets $x_{\mathcal{B}_{ijk}}\ \forall k \in \Omega(n)$. Since $|\Omega(n)| = n$ and $\mathcal{B}_{ijk} \subset \mathcal{S}_{ij}$, the overall number of states in sets $x_{\mathcal{B}_{ijk}}$ must satisfy

$$\sum_{k\in\Omega(n)} |x_{\mathcal{B}_{ijk}}| \leq n + (n-1)(|\mathcal{S}_{ij}| - 1). \tag{32}$$

In the right-hand side of (32), the first term indicates that $x_i$ appear $n$ times as elements in sets $x_{\mathcal{B}_{ijk}}\ \forall k \in \Omega(n)$. The second term indicates that any states other than $x_i$ in $x_{\mathcal{S}_{ij}}$ may appear at most $n-1$ times as elements in sets $x_{\mathcal{B}_{ijk}}\ \forall k \in \Omega(n)$. To continue, recall that $|\Omega(n)| = n$, $|x_{\mathcal{B}_{ijk}}| = |\mathcal{B}_{ijk}| = \sigma_i - \kappa_i$, $|\mathcal{S}_{ij}| = \sigma_i$, and $\sigma_i \geq (n\kappa_i + 2)$, then

$$\begin{aligned}
&\sum_{k\in\Omega(n)} |x_{\mathcal{B}_{ijk}}| - n - (n-1)(|\mathcal{S}_{ij}| - 1)\\
&= n(\sigma_i - \kappa_i) - (n-1)\sigma_i - 1\\
&= \sigma_i - n\kappa_i - 1\\
&\geq n\kappa_i + 2 - n\kappa_i - 1 = 1.
\end{aligned} \tag{33}$$

Clearly, (32) contradicts with (33). Thus, there are at least two common states shared by sets $x_{\mathcal{B}_{ijk}}\ \forall k \in \Omega(n)$. Therefore, (31) must hold. ■

*Lemma 4:* If $x_i \notin \mathcal{H}(x_{\hat{\mathcal{S}}_{ij}})$, then

$$\left(\bigcap_{k=1}^{b_i} \mathcal{H}(x_{\mathcal{B}_{ijk}})\right) \setminus \{x_i\} \neq \emptyset. \tag{34}$$

*Proof:* Based on Lemma 3, we prove Lemma 4 by induction. Define subsets $\Omega(q) \subset \{1,\dots,b_i\}$, where $|\Omega(q)| = q$ and $q$ is an integer such that $n \leq q \leq b_i$. By Lemma 3, when $q = n$, one has

$$\bigcap_{k\in\Omega(q)} \mathcal{H}(x_{\mathcal{B}_{ijk}}) \setminus \{x_i\} \neq \emptyset. \tag{35}$$

Now, given any $\Omega(q+1)$, define $\Omega_h(q) = \Omega(q+1) \setminus \{h\}$ with $h \in \Omega(q+1)$. Since $|\Omega_h(q)| = q$, (35) yields

$$\bigcap_{k\in\Omega_h(q)} \mathcal{H}(x_{\mathcal{B}_{ijk}}) \setminus \{x_i\} \neq \emptyset \quad \forall h \in \Omega(q+1). \tag{36}$$

Consequently, $\forall h \in \Omega(q+1)$, there must exist points $y_h$, such that

$$y_h \in \bigcap_{k\in\Omega_h(q)} \mathcal{H}(x_{\mathcal{B}_{ijk}}) \setminus \{x_i\}. \tag{37}$$

In the following, we use contradiction to prove:

$$\bigcap_{h\in\Omega(q+1)} \mathcal{H}(x_{\mathcal{B}_{ijh}}) \setminus \{x_i\} \neq \emptyset. \tag{38}$$

We first assume that (38) does not hold, i.e., the set in (38) is an empty set. Since $\Omega(q+1) = \Omega_h(q) \bigcup \{h\}$, this assumption leads to $\forall h \in \Omega(q+1)$

$$\left(\bigcap_{k\in\Omega_h(q)} \mathcal{H}(x_{\mathcal{B}_{ijk}})\right) \bigcap \mathcal{H}(x_{\mathcal{B}_{ijh}}) \setminus \{x_i\} = \emptyset. \tag{39}$$

Together with (37), one has $\forall h \in \Omega(q+1)$

$$y_h \notin \mathcal{H}(x_{\mathcal{B}_{ijh}}) \tag{40a}$$

$$y_h \in \mathcal{H}(x_{\mathcal{B}_{ij\widehat{h}}}) \quad \forall \widehat{h} \in \Omega(q+1),\ \widehat{h} \neq h. \tag{40b}$$

To continue, construct a set $\mathcal{Z} = \{y_h,\ h \in \Omega(q+1)\} \bigcup \{x_i\}$ with $|\mathcal{Z}| = q+2$. Since $q \geq n$, one has $|\mathcal{Z}| \geq n+2$. Then, from Radon's theorem [62], $\mathcal{Z}$ can always be partitioned into two sets $\mathcal{Z}_1$ and $\mathcal{Z}_2$, whose convex hulls intersect, i.e.,

$$y^* \in \mathcal{H}(\mathcal{Z}_1) \bigcap \mathcal{H}(\mathcal{Z}_2) \tag{41}$$

for any $h \in \Omega(q+1)$, $y_h$ only exists in one of them. Without loss of generality, let $\mathcal{Z}_e(h) \in \{\mathcal{Z}_1, \mathcal{Z}_2\}$ represent the subset such that $y_h \notin \mathcal{Z}_e(h)$. Then, based on the definition of $\mathcal{Z}$, all elements in $\mathcal{Z}_e(h)$ are either $y_{\widehat{h}}$ with $\widehat{h} \neq h$ or $x_i$. Furthermore, we have $y_{\widehat{h}} \in \mathcal{H}(x_{\mathcal{B}_{ijh}})$, $\widehat{h} \neq h$, and $x_i \in \mathcal{H}(x_{\mathcal{B}_{ijh}})$, where the first statement is derived from (40b) by exchanging the indices of $h$ and $\widehat{h}$; the second statement is by the definition of the set $x_{\mathcal{B}_{ijh}}$. Consequently, $\mathcal{H}(\mathcal{Z}_e(h)) \subset \mathcal{H}(x_{\mathcal{B}_{ijh}})$. This, together with $\mathcal{Z}_e(h) \in \{\mathcal{Z}_1, \mathcal{Z}_2\}$ and (41), yields

$$y^* \in \mathcal{H}(x_{\mathcal{B}_{ijh}}) \quad \forall h \in \Omega(q+1). \tag{42}$$

Finally, recall that $x_i \notin \mathcal{H}(x_{\hat{\mathcal{S}}_{ij}})$ and $\widehat{\mathcal{S}}_{ij} = \mathcal{S}_{ij} \setminus \{i\}$. Thus, $x_i \notin \mathcal{H}(\{y_h,\ h = 1,\dots,(q+1)\})$, that is, $x_i \notin \mathcal{H}(\mathcal{Z} \setminus \{x_i\})$. This further leads to $y^* \neq x_i$. We have

$$y^* \in \bigcap_{h\in\Omega(q+1)} \mathcal{H}(x_{\mathcal{B}_{ijh}}) \setminus \{x_i\} \neq \emptyset. \tag{43}$$

Equation (43) clearly contradicts with the assumption that (38) is false. We conclude that given (35), (38) must hold. Finally, by induction, when $q+1 = b_i$, one has $\Omega(b_i) = \{1,\dots,b_i\}$, that is

$$\left(\bigcap_{k=1}^{b_i} \mathcal{H}(x_{\mathcal{B}_{ijk}})\right) \setminus \{x_i\} \neq \emptyset. \tag{44}$$

This completes the proof.

***1) Proof of Lemma 1(a):*** From the definition of convex hull, we know each $\mathcal{H}(x_{\mathcal{B}_{ijk}})$ is a convex polytope in $\mathbb{R}^n$. Since $x_i \notin \mathcal{H}(x_{\widehat{\mathcal{S}}_{ij}})$ and $\mathcal{B}_{ijk} \subset \mathcal{S}_{ij} = \widehat{\mathcal{S}}_{ij} \bigcup \{i\}$, for all $\mathcal{H}(x_{\mathcal{B}_{ijk}})$, $k \in \{1, \ldots, b_i\}$, $x_i$ is a vertex of $\mathcal{H}(x_{\mathcal{B}_{ijk}})$. Now, let

$$\mathcal{Q}_{ij} = \bigcap_{k=1}^{b_i} \mathcal{H}(x_{\mathcal{B}_{ijk}}). \tag{45}$$

Obviously, $\mathcal{Q}_{ij}$ is a convex polytope, $x_i \in \mathcal{Q}_{ij}$, and $x_i$ is one of the vertices of $\mathcal{Q}_{ij}$. Furthermore, from Lemma 4, we know that $\mathcal{Q}_{ij} \setminus \{x_i\} \neq \emptyset$, and thus, $\mathcal{Q}_{ij}$ has at least two vertices. Since in (45), $\mathcal{Q}_{ij}$ is obtained by intersecting $\mathcal{H}(x_{\mathcal{B}_{ijk}})$, using supporting hyperplane theorem [63], it can be derived that all vertices of $\mathcal{Q}_{ij}$ must lie on some of the faces of $\mathcal{H}(x_{\mathcal{B}_{ijk}})$, for $k \in \{1, \ldots, b_i\}$. Followed by this, let $v_Q$ denote a vertex of $\mathcal{Q}_{ij}$, and let $\mathcal{Y}_1, \mathcal{Y}_2, \ldots,$ denote all the faces of polytope $\mathcal{H}(x_{\mathcal{B}_{ijk}})$, $k \in \{1, \ldots, b_i\}$ containing $v_Q$ such that

$$v_Q \in \bigcap_{\ell=1,2,\cdots} \mathcal{Y}_\ell. \tag{46}$$

From (46), if $v_Q \neq x_i$, there must exist an $\ell^*$ such that $x_i \notin \mathcal{Y}_{\ell^*}$. This along with the fact that $x_i \notin \mathcal{H}(x_{\widehat{\mathcal{S}}_{ij}})$ means the $\mathcal{Y}_{\ell^*}$ must be spanned by vectors in $x_{\widehat{\mathcal{S}}_{ij}}$, where $\widehat{\mathcal{S}}_{ij} = \mathcal{S}_{ij} \setminus \{i\}$ (examples for such $\mathcal{Y}_{\ell^*}$ are the faces of the green–yellow intersected areas in Fig. 1). Consequently, $v_Q \in \mathcal{H}(x_{\widehat{\mathcal{S}}_{ij}})$, and $v_Q \in \mathcal{Q}_{ij} \bigcap \mathcal{H}(x_{\widehat{\mathcal{S}}_{ij}}) \neq \emptyset$. This completes the proof. ■

***2) Proof of Lemma 1(b):*** First, consider the case of $x_i \in \mathcal{H}(x_{\widehat{\mathcal{S}}_{ij}})$, i.e., the step 8 of Algorithm 1. Since $\phi_{ij} = x_i$ and the agent $i$ itself is a normal agent, thus, $\phi_{ij}$ is a resilient convex combination and (10) holds.

Second, for the case of $x_i \notin \mathcal{H}(x_{\widehat{\mathcal{S}}_{ij}})$, i.e., the step 11 of Algorithm 1, one has

$$\phi_{ij} \in \left( \bigcap_{k=1}^{b_i} \mathcal{H}(x_{\mathcal{B}_{ijk}}) \right) \bigcap \mathcal{H}(x_{\widehat{\mathcal{S}}_{ij}}) \subset \bigcap_{k=1}^{b_i} \mathcal{H}(x_{\mathcal{B}_{ijk}}).$$

Recall that $\mathcal{B}_{ijk} \subset \mathcal{S}_{ij}$, $i \in \mathcal{B}_{ijk}$, and $|\mathcal{B}_{ijk}| = \sigma_i - \kappa_i$. Since the number of Byzantine neighbors of agent $i$ is bounded by $\kappa_i$, among all $k = 1, \ldots, b_i$, there must exist one $\mathcal{B}_{ijk^\star}$ which consists only normal neighbors. Then, given $\phi_{ij} \in \mathcal{H}(x_{\mathcal{B}_{ijk^\star}})$, one has $\phi_{ij}$ is a resilient convex combination and (10) holds. This completes the proof. ■

***3) Proof of Lemma 1(c):*** From steps 8 and 11 of Algorithm 1, obviously, $\phi_{ij} \in \mathcal{H}(x_{\widehat{\mathcal{S}}_{ij}})$. Then, based on Carathéodory's theorem [64], we know that there exists a subset $\Psi \subset \widehat{\mathcal{S}}_{ij}$ with $|\Psi| \leq (n+1)$, such that $\phi_{ij} \in \mathcal{H}(x_\Psi)$. That is

$$\phi_{ij} = \sum_{\ell \in \Psi} \bar{\beta}_{ij\ell} x_\ell, \quad \sum_{\ell \in \Psi} \bar{\beta}_{ij\ell} = 1, \quad \bar{\beta}_{ij\ell} \geq 0. \tag{47}$$

Then, since

$$\sum_{\ell \in \Psi} \bar{\beta}_{ij\ell} = 1, \quad \bar{\beta}_{ij\ell} \geq 0 \quad \text{and} \quad |\Psi| \leq (n+1)$$

there exists at least one $\ell^* \in \Psi$, such that $\bar{\beta}_{ij\ell^*} \geq \frac{1}{n+1}$. Bringing this to (11), for $l \in \widehat{\mathcal{S}}_{ij}$, if $\ell \in \Psi \subset \widehat{\mathcal{S}}_{ij}$, let $\beta_{ij\ell} = \bar{\beta}_{ij\ell}$; if $\ell \notin \Psi$, let $\beta_{ij\ell} = 0$, otherwise. Then, it follows that $\beta_{ij\ell^*} = \bar{\beta}_{ij\ell^*} \geq \frac{1}{n+1}$. This completes the proof. ■

## REFERENCES


[1] F. Bullo, J. Cortes, and S. Martinez, *Distributed Control of Robotic Networks*. Princeton, NJ, USA: Princeton Univ. Press, 2009.

[2] W. Ren and R. W. Beard, *Distributed Consensus in Multi-Vehicle Cooperative Control*. Berlin, Germany: Springer, 2008.

[3] G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 3, pp. 1245–1260, Sep. 2018.

[4] Z. Qiu, S. Liu, and L. Xie, "Distributed constrained optimal consensus of multi-agent systems," *Automatica*, vol. 68, pp. 209–215, 2016.

[5] M. Cao, A. S. Morse, and B. D. O. Anderson, "Agree asynchronously," *IEEE Trans. Autom. Control*, vol. 53, no. 8, pp. 1826–1838, Sep. 2008.

[6] M. Cao, A. S. Morse, and B. D. O. Anderson, "Reaching a consensus in a dynamically changing environment: A graphical approach," *SIAM J. Control Optim.*, vol. 47, no. 2, pp. 575–600, 2008.

[7] X. Chen, M. A. Belabbas, and T. Basar, "Distributed averaging with linear objective maps," *Automatica*, vol. 70, no. 3, pp. 179–188, 2016.

[8] Q. Liu, S. Yang, and Y. Hong, "Constrained consensus algorithms with fixed step size for distributed convex optimizations over multiagent networks," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 4259–4265, Aug. 2017.

[9] X. Wang, S. Mou, and B. D. Anderson, "Consensus-based distributed optimization enhanced by integral feedback," *IEEE Trans. Autom. Control*, vol. 68, no. 3, pp. 1894–1901, Mar. 2023.

[10] P. Wang, S. Mou, J. Lian, and W. Ren, "Solving a system of linear equations: From centralized to distributed algorithms," *Annu. Rev. Control*, vol. 47, pp. 306–322, 2019.

[11] T. Yang et al., "A survey of distributed optimization," *Annu. Rev. Control*, vol. 47, pp. 278–305, 2019.

[12] X. Zeng, P. Yi, and Y. Hong, "Distributed continuous-time algorithm for constrained convex optimizations via nonsmooth analysis approach," *IEEE Trans. Autom. Control*, vol. 62, no. 10, pp. 5227–5233, Oct. 2017.

[13] M. Mesbahi and M. Egerstedt, *Graph Theoretic Methods in Multi-Agent Networks*. Princeton, NJ, USA: Princeton Univ. Press, 2010.

[14] X. Chen, M.-A. Belabbas, and T. Başar, "Controllability of formations over directed time-varying graphs," *IEEE Trans. Control Netw. Syst.*, vol. 4, no. 3, pp. 407–416, Sep. 2017.

[15] Y. Zhou, W. Jin, and X. Wang, "D3G: Learning multi-robot coordination from demonstrations," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 8898–8904.

[16] C. Yan and H. Fang, "A new encounter between leader–follower tracking and observer-based control: Towards enhancing robustness against disturbances," *Syst. Control Lett.*, vol. 129, pp. 1–9, 2019.

[17] G. Shi, B. D. O. Anderson, and U. Helmke, "Network flows that solve linear equations," *IEEE Trans. Autom. Control*, vol. 62, no. 6, pp. 2659–2674, Jun. 2017.

[18] X. Wang, J. Zhou, S. Mou, and M. J. Corless, "A distributed algorithm for least squares solutions," *IEEE Trans. Autom. Control*, vol. 64, no. 10, pp. 4217–4222, Oct. 2019.

[19] S. Sundaram and B. Gharesifard, "Distributed optimization under adversarial nodes," *IEEE Trans. Autom. Control*, vol. 64, no. 3, pp. 1063–1076, Mar. 2019.

[20] M. E. Whitman and H. J. Mattord, *Principles of Information Security*. Boston, MA, USA: Cengage Learn., 2011.

[21] M. E. Liggins, C.-Y. Chong, I. Kadar, M. G. Alford, V. Vannicola, and S. Thomopoulos, "Distributed fusion architectures and algorithms for target tracking," *Proc. IEEE*, vol. 85, no. 1, pp. 95–107, Jan. 1997.

[22] W. Yang, Y. Zhang, G. Chen, C. Yang, and L. Shi, "Distributed filtering under false data injection attacks," *Automatica*, vol. 102, pp. 34–44, 2019.

[23] J. Zhou, W. Yang, H. Zhang, W. X. Zheng, Y. Xu, and Y. Tang, "Security analysis and defense strategy of distributed filtering under false data injection attacks," *Automatica*, vol. 138, 2022, Art. no. 110151.

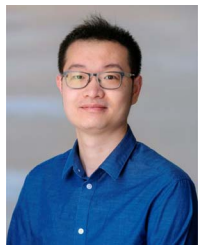
[24] M. Ben-Or, "Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols," in *Proc. 2nd Annu. ACM Symp. Princ. Distrib. Comput.*. ACM, 1983, pp. 27–30.

[25] G. Bracha, "An asynchronous $[(n-1)/3]$-resilient consensus protocol," in *Proc. 3rd Annu. ACM Symp. Princ. Distrib. Comput.*. ACM, 1984, pp. 154–162.

[26] H. J. LeBlanc, H. Zhang, X. Koutsoukos, and S. Sundaram, "Resilient asymptotic consensus in robust networks," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 4, pp. 766–781, Apr. 2013.

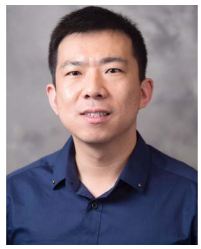
[27] S. M. Dibaji and H. Ishii, "Resilient consensus of second-order agent networks: Asynchronous update rules with delays," *Automatica*, vol. 81, pp. 123–132, 2017.

[28] L. Su and N. Vaidya, "Byzantine multi-agent optimization: Part I," 2015, *arXiv:1506.04681*.
[29] L. Su and N. H. Vaidya, "Byzantine-resilient multiagent optimization," *IEEE Trans. Autom. Control*, vol. 66, no. 5, pp. 2227–2233, May 2021.
[30] K. Kuwarancharoen, L. Xin, and S. Sundaram, "Byzantine-resilient distributed optimization of multi-dimensional functions," in *Proc. Amer. Control Conf.*, 2020, pp. 4399–4404.
[31] J. Li, W. Abbas, M. Shabbir, and X. Koutsoukos, "Resilient distributed diffusion for multi-robot systems using centerpoint," in *Proc. Robot.: Sci. Syst.*, Corvalis, OR, USA, 2020, pp. 1–9.
[32] P. Blanchard et al., "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 119–129.
[33] H. Mendes, M. Herlihy, N. Vaidya, and V. K. Garg, "Multidimensional agreement in Byzantine systems," *Distrib. Comput.*, vol. 28, no. 6, pp. 423–441, 2015.
[34] N. H. Vaidya, "Iterative Byzantine vector consensus in incomplete graphs," in *Proc. Int. Conf. Distrib. Comput. Netw.*, 2014, pp. 14–28.
[35] H. Park and S. Hutchinson, "A distributed robust convergence algorithm for multi-robot systems in the presence of faulty robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 2980–2985.
[36] H. Park and S. Hutchinson, "An efficient algorithm for fault-tolerant rendezvous of multi-robot systems with controllable sensing range," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 358–365.
[37] H. Park and S. A. Hutchinson, "Fault-tolerant rendezvous of multirobot systems," *IEEE Trans. Robot.*, vol. 33, no. 3, pp. 565–582, Jun. 2017.
[38] W. Mulzer and D. Werner, "Approximating Tverberg points in linear time for any fixed dimension," *Discrete Comput. Geometry*, vol. 50, no. 2, pp. 520–535, 2013.
[39] P. K. Agarwal, M. Sharir, and E. Welzl, "Algorithms for center and Tverberg points," *ACM Trans. Algorithms*, vol. 5, no. 1, p. 1–20, 2008.
[40] L. Tseng and N. H. Vaidya, "Asynchronous convex Hull consensus in the presence of crash faults," in *Proc. ACM Symp. Princ. Distrib. Comput.*, 2014, pp. 396–405.
[41] X. Wang, S. Mou, and S. Sundaram, "A resilient convex combination for consensus-based distributed algorithms," *Numer. Algebra, Control Optim.*, vol. 9, pp. 269–281, 2018.
[42] J. Yan, Y. Mo, X. Li, and C. Wen, "A "safe kernel" approach for resilient multi-dimensional consensus," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 2507–2512, 2020.
[43] J. Yan, X. Li, Y. Mo, and C. Wen, "Resilient multi-dimensional consensus in adversarial environment," *Automatica*, vol. 145, 2022, Art. no. 110530.
[44] A. Nedic, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Trans. Autom. Control*, vol. 55, no. 4, pp. 922–938, Apr. 2010.
[45] S. Mou, J. Liu, and A. S. Morse, "A distributed algorithm for solving a linear algebraic equation," *IEEE Trans. Autom. Control*, vol. 60, no. 11, pp. 2863–2878, Nov. 2015.
[46] X. Wang, S. Mou, and B. D. Anderson, "Scalable, distributed algorithms for solving linear equations via double-layered networks," *IEEE Trans. Autom. Control*, vol. 65, no. 3, pp. 1132–1143, Mar. 2020.
[47] Y. T. Lee and A. Sidford, "Efficient inverse maintenance and faster algorithms for linear programming," in *Proc. IEEE 56th Annu. Symp. Foundations Comput. Sci.*, 2015, pp. 230–249.
[48] W. Ren and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Trans. Autom. Control*, vol. 50, no. 5, pp. 655–661, May 2005.
[49] P. Lin and W. Ren, "Constrained consensus in unbalanced networks with communication delays," *IEEE Trans. Autom. Control*, vol. 59, no. 3, pp. 775–781, Mar. 2014.
[50] J. Wolfowitz, "Products of indecomposable, aperiodic, stochastic matrices," *Proc. Amer. Math. Soc.*, vol. 14, no. 5, pp. 733–737, 1963.
[51] J. M. Anthonisse and H. Tijms, "Exponential convergence of products of stochastic matrices," *J. Math. Anal. Appl.*, vol. 59, no. 2, pp. 360–364, 1977.
[52] K. Jetter and X. Li, "SIA matrices and non-negative subdivision," *Results Math.*, vol. 62, no. 3-4, pp. 355–375, 2012.
[53] W. Cao, J. Zhang, and W. Ren, "Leader–follower consensus of linear multi-agent systems with unknown external disturbances," *Syst. Control Lett.*, vol. 82, pp. 64–70, 2015.
[54] M. Ajtai, "The complexity of the pigeonhole principle," *Combinatorica*, vol. 14, pp. 417–433, 1994.
[55] Y. Chen, W. Xiong, and F. Li, "Convergence of infinite products of stochastic matrices: A graphical decomposition criterion," *IEEE Trans. Autom. Control*, vol. 61, no. 11, pp. 3599–3605, Nov. 2016.
[56] D. B. West, *Introduction to Graph Theory*. Upper Saddle River, NJ, USA: Prentice Hall, 2001, vol. 2.
[57] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Rev. Modern Phys.*, vol. 74, no. 1, 2002, Art. no. 47.
[58] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar, "Fully decentralized multi-agent reinforcement learning with networked agents," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5872–5881.
[59] A. Burkov, *The Hundred-Page Machine Learning Book*, vol. 1. Québec, QC, Canada: Andriy Burkov, 2019.
[60] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT press, 2018.
[61] P. Heredia, H. Ghadialy, and S. Mou, "Finite-sample analysis of distributed Q-learning for multi-agent networks," in *Proc. Amer. Control Conf.*, 2020, pp. 3511–3516.
[62] H. Tverberg, "A generalization of Radon's theorem," *J. London Math. Soc.*, vol. 41, pp. 123–128, 1966.
[63] R. Hartshorne, *Algebraic Geometry*, vol. 52. Berlin, Germany: Springer, 2013.
[64] W. Cook, J. Fonlupt, and A. Schrijver, "An integer analogue of Caratheodory's theorem," *J. Combinatorial Theory, Ser. B*, vol. 40, no. 1, pp. 63–70, 1986.



**Xuan Wang** received the Ph.D. degree in autonomy and control from the School of Aeronautics and Astronautics, Purdue University, West Lafayette, IN, USA, in 2020.

He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, George Mason University, Fairfax, VA, USA. From 2020 to 2021, he was a Postdoctoral Researcher with the Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA, USA. His research interests include multiagent control and optimization, resilient multiagent coordination, system identification, and data-driven control of network dynamical systems.



**Shaoshuai Mou** (Member, IEEE) received the Ph.D. degree in electrical engineering from Yale University, New Haven, CT, USA, in 2014.

He was a Postdoctoral Researcher with the Massachusetts Institute of Technology, Cambridge, MA, USA, for a year, and then joined Purdue University, West Lafayette, IN, USA, as a Tenure-Track Assistant Professor, in August 2015. He is the Elmer Bruhn Associate Professor of aeronautics and astronautics with Purdue University. His research interests include advancing control theories with recent progress in optimization, networks, and learning to address fundamental challenges in autonomous systems, with particular research interests in distributed control, optimization and learning, control of autonomous robots, human–robot teaming, cybersecurity, and resilience.

**Shreyas Sundaram** (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 2009.

He was a Postdoctoral Researcher with the University of Pennsylvania, Philadelphia, PA, USA, from 2009 to 2010, and an Assistant Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, from 2010 to 2014. He is currently the Marie Gordon Professor with the Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA. His research interests include network science, analysis of large-scale dynamical systems, fault-tolerant and secure control, linear system and estimation theory, and game theory.

Dr. Sundaram was the recipient of the NSF CAREER Award. At Purdue, he was the recipient of the HKN Outstanding Professor Award, Outstanding Mentor of Engineering Graduate Students Award, Hesselberth Award for Teaching Excellence, and Ruth and Joel Spira Outstanding Teacher Award.