Resilient Vector Consensus in Multi-Agent Networks Using Centerpoints

Mudassir Shabbir, Jiani Li, Waseem Abbas, and Xenofon Koutsoukos

***Abstract*— In this paper, we study the resilient vector consensus problem in multi-agent networks and improve resilience guarantees of existing algorithms. In resilient vector consensus, agents update their states, which are vectors in $\mathbb{R}^d$, by locally interacting with other agents some of which might be adversarial. The main objective is to ensure that normal (non-adversarial) agents converge at a common state that lies in the convex hull of their initial states. Currently, resilient vector consensus algorithms, such as approximate distributed robust convergence (ADRC) are based on the idea that to update states in each time step, every normal node needs to compute a point that lies in the convex hull of its normal neighbors' states. To compute such a point, the idea of Tverberg partition is typically used, which is computationally hard. Approximation algorithms for Tverberg partition negatively impact the resilience guarantees of consensus algorithm. To deal with this issue, we propose to use the idea of centerpoint, which is an extension of median in higher dimensions, instead of Tverberg partition. We show that the resilience of such algorithms to adversarial nodes is improved if we use the notion of centerpoint. Furthermore, using centerpoint provides a better characterization of the necessary and sufficient conditions guaranteeing resilient vector consensus. We analyze these conditions in two, three, and higher dimensions separately. We also numerically evaluate the performance of our approach.**

***Index Terms*— Resilient consensus, computational geometry, centerpoint, fault tolerant networks.**

I. Introduction

Resilient consensus in a network of agents, some of which might be adversarial or faulty, has several applications in multirobot networks, distributed computing, estimation, learning and optimization (for instance, see [1], [2], [3], [4], [5]). The main goal of resilient consensus is to ensure that all normal agents in a network agree on a common state despite the presence of some adversarial agents, which aim to prevent normal nodes from consensus and whose identities are unknown to normal agents. Resilient consensus is achieved if appropriate state update laws are designed for normal agents and the underlying network topology satisfies certain connectivity and robustness conditions. For instance, when agents' states are scalars, [6] presents a resilient distributed algorithm guaranteeing convergence of normal nodes to a common state.

If agents' states are vectors or points in $\mathbb{R}^d$, $d \geq 2$, then the resilient consensus objective is to ensure that normal agents converge at some point in the convex hull of their initial states. A simple approach could be to run d instances of scalar resilient consensus, one for each dimension. However, as a result of this approach, normal agents might converge at a point outside of the convex hull of their initial states, as discussed in [7]. Thus, we cannot rely on resilient scalar consensus algorithms to achieve resilient vector consensus. Various solutions have been proposed to achieve resilient vector consensus, which has been an active research topic, for instance see [2], [7], [8], [9], [10].

In this paper, we study the resilient vector consensus problem and a recently proposed solution referred to as the *Approximate Distributed Robust Convergence (ADRC)* algorithm in [2]. We show that the resilience of the algorithm, in terms of the number of adversarial agents whose presence does not prevent normal agents from converging to a common state in the desired convex hull, is improved with some simple modification. In particular, if normal agents implement ADRC as in [2], then consensus is guaranteed if the number of adversarial agents in the neighborhood of a normal agent i is $n_{f_i} \leq \left\lceil \frac{|\mathcal{N}_i|}{2^d} \right\rceil - 1$, where $|\mathcal{N}_i|$ is the number of nodes in the neighborhood of i, and d is the dimension of state vector. We show that in the case of $d = 2, 3$, consensus is guaranteed if $n_{f_i} \leq \left\lceil \frac{|\mathcal{N}_i|}{d+1} \right\rceil - 1$, and for $d > 3$ if $n_{f_i} \leq \left\lceil \frac{|\mathcal{N}_i|}{\frac{r}{d^{r-1}}} \right\rceil - 1$ where r can be any integer.

ADRC is an iterative algorithm, and in each iteration, a normal node needs to compute some point in the convex hull of points corresponding to its normal neighbors' states. To compute such a point, which is referred to as the *safe point*, authors in [2] utilize the idea of *Tverberg partition* of points in $\mathbb{R}^d$ (discussed in Section III). We argue that instead of computing Tverberg partition, it is much better to use the notion of *centerpoint* in $\mathbb{R}^d$ to compute safe points. The notion of centerpoint and its properties have been an active research topic in discrete geometry [11], [12]. A centerpoint essentially extends the notion of median in higher dimensions. We show that safe points, as used in ADRC algorithm, are essentially the interior centerpoints. This perspective provides a complete characterization of safe points, and hence allows us to improve the resilience bound of the algorithm.

- We show that the resilience of ADRC algorithm can be improved by using the notion of centerpoint instead of Tverberg partition. We discuss these improvements in two, three and higher dimensions separately.
- Using centerpoints, we show that $|\mathcal{N}_i| \geq (n_{f_i}+1)(d+1)$ is not only sufficient but also necessary to compute a safe point, which is a key step in the ADRC algorithm. Here n_{f_i} is the number of adversaries in the neighborhood of a normal node i. We also provide an

M. Shabbir is with the Computer Science Department at the Information Technology University, Lahore, Pakistan (Email:mudassir@rutgers.edu).

J. Li, W. Abbas, and X. Koutsoukos are with the Electrical Engineering and Computer Science Department at Vanderbilt University, Nashville, TN, USA (Emails: {jiani.li, waseem.abbas, xenofon.koutsoukos}@vanderbilt.edu).

overview of various algorithms reported in the literature to compute centerpoints in different dimensions.
- We compare and numerically evaluate our results with the existing algorithm by simulating resilient vector consensus in multirobot networks.

The rest of the paper is organized as follows: Section II introduces notations and preliminaries. Section III provides an overview of the ADRC algorithm. Section IV discusses the notion of centerpoint for ADRC and presents main results in the paper. Section V gives a numerical evaluation of our results, and Section VI concludes the paper.

II. Notations and Preliminaries

We consider a network of agents modeled by a *directed graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with self-loops allowed, where $\mathcal{V}$ represents agents and $\mathcal{E}$ represents interactions between agents. Each agent $i \in \mathcal{V}$ has a d-dimensional state vector whose value is updated over time. The state of each agent i at time t is represented by a point $x_i(t) \in \mathbb{R}^d$. An edge (j, i) means that i can observe the state value of j. The *neighborhood* of i is the set of nodes $\mathcal{N}_i = \{j \in \mathcal{V} | (j, i) \in \mathcal{E}\}$. For a given set of points $X \subset \mathbb{R}^d$, we denote its *convex hull* by conv(X). A set of points in $\mathbb{R}^d$ is said to be in *general positions* if no hyperplane of dimension $d-1$ or less contains more than d points. A point $x \in \mathbb{R}^d$ is an *interior point* of a set $X \subset \mathbb{R}^d$ if there exists an open ball centered at x which is completely contained in X. We use terms agents and nodes interchangeably, and similarly use terms points and states interchangeably.

Normal and Adversarial Agents: There are two types of agents in the network, normal and adversarial. *Normal* agents are the ones that interact with their neighbors synchronously and always update their states according to a pre-defined state update rule, that is the consensus algorithm. *Adversarial* agents are the ones that can change their states arbitrarily and do not follow the pre-defined state update rule. Moreover, an adversarial node can transmit different values to its different neighbors, which is referred to as the *Byzantine* model. The number of adversarial nodes in the neighborhood of a normal node i is denoted by n_{f_i}. For a normal node i, all nodes in its neighborhood are indistinguishable, that is, i cannot identify which of its neighbors are adversarial.

Resilient Vector Consensus: The goal of the resilient vector consensus is to ensure the following two conditions:

- *Safety* – Let $X(0) = \{x_1(0), x_2(0), \cdots, x_n(0)\} \subset \mathbb{R}^d$ be the set of initial states of normal nodes, then at each time step t, and for any normal node i, the state value of i, denoted by $x_i(t)$ should be in the conv$(X(0))$.
- *Agreement* – For every $\epsilon > 0$, there exists some t_ϵ, such that for any normal node pair i, j, $||x_i(t) - x_j(t)|| < \epsilon$, $\forall t > t_\epsilon$.

III. Background and Approximate Distributed Robust Convergence (ADRC) Algorithm

In this section, first we provide an overview of a resilient vector consensus algorithm known as the approximate distributed robust convergence, recently proposed in [2]. Then, we discuss improvement in resilience guarantees of the algorithm by reconsidering its computational aspects.

The ADRC is an iterative algorithm, in which a normal node i gathers the state values of its neighbors in each iteration t, and then computes a point that lies in the interior of the convex hull of its normal neighbors' states. After computing this point, which is referred to as the *safe point* $s_i(t)$, node i updates its state as follows:

$$x_i(t+1) = \alpha_i(t) s_i(t) \; + \; (1 - \alpha_i(t)) x_i(t), \qquad (1)$$

where, $\alpha_i(t)$ is a dynamically chosen parameter in the range $(0 \quad 1]$.[1] It is shown in [2] that if all normal nodes follow this procedure, they converge at a common point and achieve resilient consensus (satisfying safety and agreement conditions stated in the previous section).

Computation of safe point in each iteration is the key step in the algorithm. For this, [2] utilizes results from discrete geometry, in particular the idea of *Tverberg partitions* [13] and related results. We first state the main result regarding the partitioning of points in $\mathbb{R}^d$, and then discuss the application of this result, as adapted in [2], for computing safe points.

Proposition 3.1: ([14], [15], [16]) If we have a set X of n points in general positions in $\mathbb{R}^d$, where $n \geq r(d+1)$ and $d \leq 8$, then it is possible to partition X into r pairwise disjoint subsets $X_1, X_2, \cdots, X_r$ such that the intersection of convex hulls of these r subsets is non-empty and is at least d-dimensional.

Such a partition is a *Tverberg partition*. Now, consider a normal node i in our network having n neighbors in its neighborhood out of which at most n_f are adversarial.[2] Each node corresponds to a point in $\mathbb{R}^d$. The goal for a node i is to compute an interior point in the convex hull of $n - n_f$ normal points. If $n \geq (n_f + 1)(d+1)$, then by Proposition 3.1, we will have a partition of n points into $n_f + 1$ subsets such that the intersection of convex hulls of these subsets is non-empty and is d-dimensional. We call this intersection region as *Tverberg region*. Since there are at most n_f adversarial nodes and we have $n_f + 1$ subsets in the partition, one of these subsets consists of points corresponding to normal nodes only. Let us denote this subset by X^*. Note that the Tverberg region lies in the convex hull of X^*, and conv(X^*) itself lies in the convex hull of all normal nodes points. Consequently, every interior point in the Tverberg region is a safe point. Thus, to compute a safe point, a normal node i computes a Tverberg partition, which is possible if $n \geq (n_f + 1)(d+1)$. In other words, a normal node can compute a safe point in the presence of n_f adversarial neighbors if $n_f \leq \frac{n}{d+1} - 1$. Figure 1 gives an illustration of these ideas.

However, computing a Tverberg partition in general is an NP-hard problem. The best known algorithm that computes

[1] The choice of $\alpha_i(t)$ depends on applications, for instance, in multirobot systems, it is selected such that the physical constraints including maximum allowable displacement by a robot is not violated.

[2] For the ease of notation, we drop the subscript i from n_i and n_{f_i} denoting the total number of neighbors and the number of adversarial neighbors of node i respectively from here onward. Note that, in general these values can be different for different nodes.

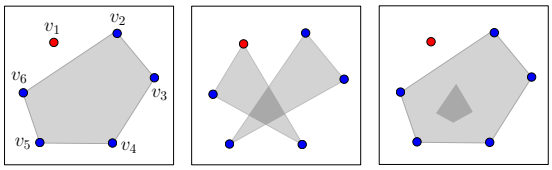


Fig. 1. (a) Five normal (blue) and a single adversarial node. Shaded area is the convex hull of normal nodes. (b) Tverberg partition consisting of two subsets, out of which one contains only normal nodes. Convex hulls of both subsets have a non-empty intersection, corresponding to a Tverberg region. (c) Intersection of Tverberg region and the convex hull of normal nodes.

it in a reasonable run time is an approximate algorithm [17], which has a time complexity of $d^{O(1)}n$. The algorithm is approximate in a sense that to have a partition of n points into r subsets, $n \geq 2^d r$ (as compared to $n \geq r(d+1)$ in Proposition 3.1). Consequently, to compute a safe point in the presence of n_f adversarial neighbors, a normal node needs to have at least $n \geq (n_f+1)2^d$ nodes in its neighborhood. In other words, with a total of n neighbors, a node i can compute a safe point, and hence achieve resilient consensus (using ADRC) if there are n_f adversarial nodes in its neighborhood, where

$$n_f \leq \left\lceil \frac{n}{2^d} \right\rceil - 1. \tag{2}$$

Note that (2) indicates *resilience* of the ADRC algorithm that relies on approximate Tverberg partitions to compute safe points. For instance, the algorithm guarantees resilient consensus in $\mathbb{R}^2$ if for every normal node, less than 25% of its neighbors are adversarial.

A. How Can We Improve the Resilience of ADRC?

Here, we ask if it is possible to improve the resilience of the ADRC algorithm? What modifications will allow us to guarantee consensus even if the number of adversarial nodes in the neighborhood of a normal node is greater than $\lceil \frac{n}{2^d} \rceil - 1$? Next, we show that it is possible to achieve a better resilience bound if we use a slightly different way of computing safe points, that is by using the notion of *centerpoint* instead of Tverberg partition. Moreover, centerpoint provides a better characterization of necessary and sufficient conditions for computing safe points.

IV. ADRC Using Centerpoints

In this section, we explain the notion of a *centerpoint* and its relation to safe point. Then, we discuss that computing a safe point through centerpoint is more desirable as it results in improving the resilience of the ADRC algorithm.

A. Safe point and the Interior Centerpoint

The notion of safe point is pivotal in the ADRC algorithm, so we define n_f-safe point as in [2] below.

Definition 4.1: *(n_f-Safe point)* Given a set of n points in d dimensions, of which at most n_f correspond to adversarial nodes, an n_f-safe point is a point that lies in the relative interior of the convex hull of $(n - n_f)$ normal points. We refer to a $(\frac{n}{d+1} - 1)$-safe point in $\mathbb{R}^d$ as an optimal safe point or just safe point.

As we discussed in the last section, a point that lies in the interior of the Tverberg region of (n_f+1) subsets is always an n_f-safe point Here, we provide a better characterization of n_f-safe point using centerpoint, which is defined below.

Definition 4.2: *(Centerpoint)* Given a set S of n points in $\mathbb{R}^d$ in general positions, a centerpoint p is a point, not necessarily from S, such that any closed half-space[3] of $\mathbb{R}^d$ that contains p also contains at least $\frac{n}{d+1}$ points from S.
Intuitively, a centerpoint lies in the "center region" of the point cloud, in the sense that there are enough points of S on each side of a centerpoint. A centerpoint, essentially, extends the notion median to higher dimensions. A related notion of centerpoint depth is defined as follows:

Definition 4.3: For a given pointset, *centerpoint depth* or simply *depth* of a point q is the maximum number α such that every closed halfspace containing q contains at least α points.
Thus, a centerpoint has depth at least $\frac{n}{d+1}$. The existence of such a point for any given set S is guaranteed by the famous Centerpoint Theorem (see [18], [19]).

Theorem 4.4: *(Centerpoint Theorem)* For any given point set in general positions in an arbitrary dimension, a centerpoint always exists.
A centerpoint doesn't need to be unique, in fact, there can be infinitely many centerpoints. The set of all centerpoints constitutes the *centerpoint region* or simply the *center region*. It is known that center region is closed and convex. We observe that the safe point from [2] is actually an interior centerpoint.

Theorem 4.5: For a given set of points S in $\mathbb{R}^d$, an n_f-safe point is equivalent to an interior centerpoint for $n_f = \frac{n}{d+1} - 1$.

Proof: See [20].

Theorem 4.5 provides a complete characterization of a safe point in the presence of n_f adversarial nodes. Here, we would also like to note that $n_f = \frac{1}{d+1} - 1$ is the best possible fraction, that is, there exist general node positions where allowing more adversary nodes would mean that there is no safe point at all.

Proposition 4.6: For a set of n nodes in general positions, if $n_f \geq \frac{n}{d+1}$, then there exist general examples in which an n_f-safe point does not exist.

Proof: Imagine $d+1$ copies of $\frac{n}{d+1}$ points at the vertices of a non-degenerate d-simplex. If there are $\frac{n}{d+1}$ adversarial nodes whose identities are unknown, then there is no point that lies in the convex hull of remaining points. Note that points in this examples can be arbitrarily perturbed to ensure that general positions condition is not violated. ■

Figure 2 demonstrates Proposition 4.6 for the planar case. Further, we note that every point in the intersection of an appropriate Tverberg partition is a centerpoint and thus, also a safe point. However, the converse is not true; a centerpoint or a safe point need not be a Tverberg point in general.

[3]Recall that closed half-space in $\mathbb{R}^d$ is a set of the form $\{x \in \mathbb{R}^d : a^T x \geq b\}$ for some $a \in \mathbb{R}^d \setminus \{0\}$.

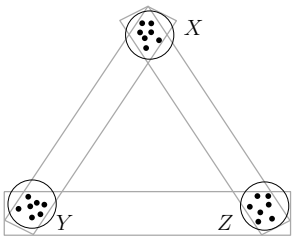


Fig. 2. S is partitioned into X, Y, Z each of which contains $n/3$ points. If there are $n/3$ adversarial nodes then points in either of these three sets can all be adversarial. We require that an n_f-safe point must lie in the convex hull of normal nodes for all three possibilities. This is not possible because intersection of three possible sets of normal nodes $X \cup Y$, $Y \cup Z$, $Z \cup X$ is empty. Therefore, there is no n_f-safe point in this case.

In [2], normal nodes compute safe points using approximate Tverberg partitions [17], which deteriorate the resilience of ADRC algorithm from $n_f \leq \frac{n}{d+1} - 1$ to $n_f \leq \frac{n}{2^d} - 1$. However, with this new characterization, we can use centerpoints to compute safe points. Thus, if we are able to compute centerpoint exactly (in a reasonable run time), then we are able to improve the resilience of ADRC algorithm, that is,

$$n_f \leq \frac{n}{d+1} - 1 \tag{3}$$

as compared to (2).

Next, we discuss the existence and computation of interior centerpoints in two, three and higher dimensions separately.

B. Centerpoint-based Resilient Consensus in 2-D

In [2], authors show that an n_f-safe point can be found in $\mathbb{R}^2$ when the number of adversarial nodes n_f is at most $\min\left(\lceil \frac{n}{4} \rceil, \lfloor \frac{n}{3} \rfloor\right) - 1$ where n is total number of nodes in the neighborhood of a normal node. As is evident, this is a loose bound on the resilience of such a consensus algorithm. Unfortunately, that is the best that can be hoped for if one is to seek a safe-point using an approximate Tverberg partition. Tverberg partitions are, in general, thought to be computationally expensive. However, we have showed in Theorem 4.5 that safe points and the interior centerpoints always coincide, and in the following subsection, we summarize a well-known linear time algorithm to find a centerpoint in the plane. It should be pointed out that complexity of finding a centerpoint in general dimensions is unknown, although computing centerpoint depth of a given point is coNP-Complete. In this section, we propose the following result to compute a safe point in the plane:

Theorem 4.7: Given a set of n points in two dimensions in general positions, an n_f-safe point exists whenever the number of adversarial nodes is $n_f \leq \frac{n}{3} - 1$. Moreover, such a safe point can be computed in linear time.

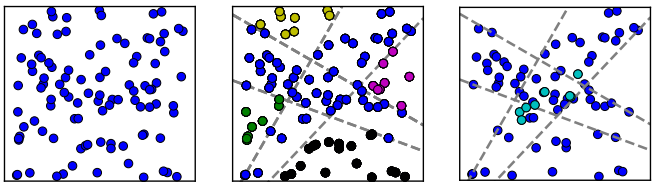
Proof: See [20].

Remark 4.8: The set of Tverberg points is a subset of the centerpoint region. Thus, the existence of an interior centerpoint in the plane, as shown above, is also implied by the existence of an interior Tverberg point in dimensions two to eight. We hope that the alternative proof above may help extend this result to dimensions greater than 8 for which the existence of an interior centerpoint and interior Tverberg point are unknown.

C. Computing Centerpoint in 2-D

Here we address the computational aspects of the centerpoint in two dimensions. Due to a seminal result in [21], it is possible to find a centerpoint for a non-degenerate pointset in the optimal $O(n)$ time. We remark that this result is also significant because it makes finding a centerpoint in linear time possible even when checking whether a point is a centerpoint can not be done in better than $\Omega(n \log n)$ time. Here, we briefly outline this method to compute a centerpoint of a set of points; the details can be found in [21].

The algorithm is based on the idea that by pruning or replacing some of the "marginal points", a centerpoint of the remaining points is still a centerpoint of the original pointset. In each iteration one can compute the points that are to be discarded or replaced, which will reduce the size of the set by a fraction. We continue the pruning procedure until the size of the set becomes smaller than a fixed constant, one can then compute a centerpoint by any straightforward brute-force method. Pruning of points is a pivotal step in the algorithm. Given a set of n points $\mathcal{P}$, we start by defining four half-planes, named L, U, R, and D (representing Left, Up, Right, and Down, respectively), such that each of them contains less than $\lceil \frac{n}{3} \rceil - 1$ points (this ensures that they don't contain any centerpoint) and their closures contain at least $\lceil \frac{n}{3} \rceil$ points. And the closure of each of the sets $L \cap U$, $L \cap D$, $R \cap U$, and $R \cap D$ contains at least $\left(\lceil \frac{n}{3} \rceil - \lceil \frac{n}{4} \rceil\right)$ points. It is, then, argued that either one can discard the points of a triangle on three points from three of the four intersections or substitute four points from the four intersection sets by their Radon point[4]. This reduces the size of $\mathcal{P}$ by a significant fraction and a centerpoint of the remaining point set is also a centerpoint of the original point set. The pruning process, as illustrated in Figure 3, is repeated until the number of points is less than a small constant, and then one can compute the centerpoint by a brute-force method. The construction of halfplanes with the prescribed number of points in their intersection is achieved by the famous ham-sandwich cut algorithm [22].



(a) (b) (c)

Fig. 3. One iteration illustration of replacing points in $L \cap U$, $L \cap D$, $R \cap U$, and $R \cap D$ by their Radon points: (a) point set of 100 points, (b) intersections of the four half-planes, and (c) replacement of points in the intersections by their Radon point.

[4]Any set of 4 points in $\mathbb{R}^2$ can be partitioned into two disjoint sets whose convex hulls intersect. A point in the intersection of these convex hulls is called a Radon point of the set.

D. Centerpoint-based Resilient Consensus in 3-D

The resilience bound that we get from the results in [2] guarantee an n_f-safe point in three dimensions whenever $n_f \leq \frac{n}{8}$-1 adversarial nodes. From the centerpoint theorem, we know that a safe point exists in the interior of centerpoint region in 3-D even in the presence of $(n/4) - 1$ adversarial nodes. In context of Theorem 4.5, this property can be leveraged to present a better resilience guarantee.

Theorem 4.9: An $\left(\frac{n}{4} - 1\right)$-safe point exists for every pointset in general positions in $\mathbb{R}^3$. Such a point can be computed in $O(n^2)$ expected time.

Proof: We know that an $\left(\frac{n}{4} - 1\right)$-safe point is an interior centerpoint from Theorem 4.5, and [15] implies that an interior centerpoint must exist in three dimensions. A randomized algorithm by Chan can be used to compute a centerpoint in three dimensions in $O(n^2)$ expected time [23]. We proceed by running Chan's algorithm four times, and compute the centroid of the four centerpoints returned to get an interior centerpoint. ∎

Next, we provide a brief overview of Chan's algorithm in which he computes a centerpoint of a non-degenerate pointset in $O(n^{d-1})$ time [23]. He first solves the decision version of the problem: does there exist a point of depth k for a given k? If the answer is yes, then a point of given depth is reported as well. This decision version is solved using a randomized Linear Program solver by dualizing the pointset: given points S are dualized to a set $S*$ of hyperplane and a point of given depth dualizes to special hyperplane that has at least k hyperplanes from $S*$ above or below it. The problem of finding this hyperplane is solved by partitioning the space and solving sub-problems in each smaller region. The partitioning is done by the famous Cutting Lemma [24]. For further details, we refer to the paper [23].

E. Centerpoint-based Resilient Consensus in d-dimensions for $d > 3$

In higher dimensions, current methods to compute either a desirable Tverberg partition or a centerpoint for a given pointset become computationally impractical. It is known deciding whether a point lies in the intersection of a Tverberg partition is NP-Complete and deciding whether a point is centerpoint is coNP-Complete. Various approximations are employed to compute these points in practice. In [2], authors use a "lifting-based" approximation that finds an n_f-safe point in presence of $n_f \leq \frac{n}{2^d} - 1$ adversarial nodes. In the following, we outline an algorithm by Miller and Sheehy to compute an approximate centerpoint [25]. The point returned by this algorithm has a centerpoint depth of $\frac{n}{d^{r/r-1}}$ for any integer $r \geq 2$. For $r = 3$, this gives an n_f-safe point when the number of adversarial nodes is at most $\frac{n}{d^{3/2}}$. By increasing r, the quality of approximation, and hence the bound on the number of adversarial nodes improves. However, it comes at the cost of increasing time complexity as the runtime of the algorithm is $O(rd^d)$ for an integer $r > 1$.

Miller and Sheehy centerpoint-approximation algorithm is based on the technique of Radon's theorem which states that for any given set of at least $d + 2$ points, there exists a partition into two sets with intersecting convex hulls; a point in the intersection of the two said sets is called a Radon point. They improve upon a classic algorithm that starts by partitioning a given pointset S into groups of $d+2$ points and computing Radon point for each group. The set of $\lceil \frac{|S|}{d+2} \rceil$ Radon points returned in the previous iteration are assumed to be the new pointset and centerpoint for these points is recursively computed. An approximate centerpoint is, thus, found in at most $\log_{d+2} |S|$ iterations. Miller and Sheehy showed that they can create groups of larger sizes (of multiples of $d + 2$ points) and reduce the number of iterations. Details of their algorithm, anyalsis and proof of correctness is available in [25]. As a consequence, we have the following result:

Theorem 4.10: For a given pointset in $\mathbb{R}^d$ in general positions, a $\left(\frac{n}{d^{r/r-1}}\right)$-safe point exists and can be computed in time $O(rd^d)$ for any integer $r > 1$.

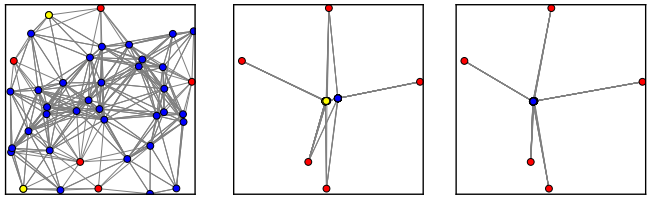
Thus, using approximate centerpoint in dimension d, consensus is guaranteed if the number of adversarial nodes in the neighborhood of every normal node is $n_f \leq \frac{n}{d^{r/r-1}}$ for an integer $r > 1$, which is better than the resilience achieved by using approximate Tverberg partition, where $n_f \leq \frac{n}{2^d} - 1$.

V. Numerical Evaluation

We perform simulations[5] to compare resilient consensus in multirobot systems in two dimensions using centerpoint and approximate Tverberg partition [2]. At each iteration t of the multi-robot consensus algorithm, a normal robot i computes a safe point $s_i(t)$ of its neighbors' positions (using centerpoint or approximate Tverberg partition), and calculates its new position using (1). In our experiments, we set $\alpha_i(t) = 0.8$. We consider stationary adversarial nodes, and assume that the network graph is undirected and fixed.[6] A group of 45 robots of which 5 are adversarial is distributed in a planar region $\mathcal{W} = [-1, 1] \times [-1, 1] \in \mathbb{R}^2$ as shown in Figure 4(a). All normal robots have at most $\left(\lceil \frac{|\mathcal{N}_i|}{3} \rceil - 1\right)$ adversaries in their neighborhood, which means resilient consensus is guaranteed by the centerpoint based algorithm. However, a couple of normal robots (depicted in yellow color) have n_{f_i} adversaries in their neighborhood, where $\left(\lceil \frac{|\mathcal{N}_i|}{4} \rceil - 1\right) < n_{f_i} \leq \left(\lceil \frac{|\mathcal{N}_i|}{3} \rceil - 1\right)$. For the two robots in yellow, they have 7 and 8 neighbors, and both of them have 2 adversarial neighbors in their respective neighborhoods. Consequently, the resilient consensus condition for the approximate Tverberg partition based algorithm is not satisfied, and consensus is not guaranteed. Figures 4(b) and (c) show final positions of robots for both algorithms. It is clear that consensus is achieved with the centerpoint based algorithm, whereas robots fail to converge at a common point using the approximate Tverberg partition based algorithm. Figure 5 illustrates positions of robots as a function of iterations and demonstrates the same results.

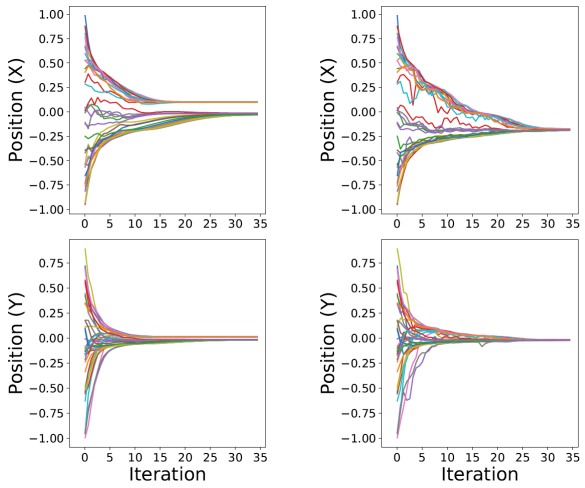
[5]Our code is available at `https://github.com/JianiLi/MultiRobotsRendezvous`

[6]For additional experiments with disk graphs and moving adversaries, see [20].



(a) (b) (c)

Fig. 4. (a) Initial positions of robots. (b) Final positions of robots using approximate Tverberg partition based algorithm. (c) Final positions using centerpoint based algorithm.



(a) (b)

Fig. 5. Positions of normal robots as a function of iterations using (a) approximate Tverberg partition based, and (b) centerpoint based algorithms.

VI. Conclusion

The task of ensuring consensus in the convex hull of vector states for a set of nodes, a fraction of which may be under the influence of an adversary, is a challenging problem. In this work, we presented a geometric characterization of an optimal point, in the form of a centerpoint, in the convex hull of normal nodes when the number of adversarial nodes is limited to a $1/(d+1)$ fraction of the size of the neighborhood of a node. It also followed that the upper bound on the number of adversarial nodes is best possible in the worst case. We proposed to use well-known efficient algorithms to compute exact centerpoints in two and three dimensions. For higher dimensions, we used an approximate centerpoint algorithm proposed in [25], and improved previous bound on the number of adversarial nodes.

It follows from our results that if the fraction of adversarial nodes in the neighborhood of every normal node is at least $d/(d+1)$, then adversarial nodes can make normal nodes converge to any point they desire. At the same time, if their fraction is less than $1/(d+1)$, then the resilient consensus is guaranteed. In future, we aim to study and characterize the effect of adversarial nodes if their fraction in the neighborhood of a normal node is between the above two numbers.

Acknowledgements

This work was supported in part by the National Institute of Standards and Technology under Grant 70NANB18H198, and by the National Science Foundation under award CNS-1739328.

References

[1] L. Tseng and N. Vaidya, "Iterative approximate byzantine consensus under a generalized fault model," in *International Conference on Distributed Computing and Networking*. Springer, 2013, pp. 72–86.

[2] H. Park and S. A. Hutchinson, "Fault-tolerant rendezvous of multirobot systems," *IEEE Transactions on Robotics*, vol. 33, pp. 565–582, 2017.

[3] J. Li and X. Koutsoukos, "Resilient distributed diffusion for multi-task estimation," in *Proceedings of the 14th International Conference on Distributed Computing in Sensor Systems*, 2018, pp. 93–102.

[4] W. Abbas, A. Laszka, and X. Koutsoukos, "Improving network connectivity and robustness using trusted nodes with application to resilient consensus," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 4, pp. 2036–2048, 2017.

[5] L. Su and N. Vaidya, "Multi-agent optimization in the presence of byzantine adversaries: Fundamental limits," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 7183–7188.

[6] H. J. LeBlanc, H. Zhang, X. Koutsoukos, and S. Sundaram, "Resilient asymptotic consensus in robust networks," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 4, pp. 766–781, 2013.

[7] N. H. Vaidya and V. K. Garg, "Byzantine vector consensus in complete graphs," in *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing (PODC)*. ACM, pp. 65–73.

[8] N. H. Vaidya, "Iterative byzantine vector consensus in incomplete graphs," in *International Conference on Distributed Computing and Networking*. Springer, 2014, pp. 14–28.

[9] H. Mendes and M. Herlihy, "Multidimensional approximate agreement in byzantine asynchronous systems," in *45th Annual ACM Symposium on Theory of Computing*, 2013, pp. 391–400.

[10] L. Su and N. H. Vaidya, "Fault-tolerant multi-agent optimization: optimal iterative distributed algorithms," in *ACM Symposium on Principles of Distributed Computing*, 2016, pp. 425–434.

[11] M. Shabbir, "Some results in computational and combinatorial geometry," Ph.D. dissertation, Rutgers University-New Brunswick, 2014.

[12] N. H. Mustafa, S. Ray, and M. Shabbir, "k-centerpoints conjectures for pointsets in d," *International Journal of Computational Geometry & Applications*, vol. 25, no. 03, pp. 163–185, 2015.

[13] H. Tverberg, "A generalization of Radon's theorem," *Journal of the London Mathematical Society*, vol. 1, no. 1, pp. 123–128, 1966.

[14] J. R. Reay, "An extension of Radon's theorem," *Illinois J. Math.*, vol. 12, no. 2, pp. 184–189, 06 1968.

[15] J.-P. Roudneff, "New cases of Reays conjecture on partitions of points into simplices with k-dimensional intersection," *European Journal of Combinatorics*, vol. 30, no. 8, pp. 1919–1943, 2009.

[16] ——, "Partitions of points into intersecting tetrahedra," *Discrete Mathematics*, vol. 81, no. 1, pp. 81–86, 1990.

[17] W. Mulzer and D. Werner, "Approximating tverberg points in linear time for any fixed dimension," *Discrete & Computational Geometry*, vol. 50, no. 2, pp. 520–535, 2013.

[18] R. Rado, "A theorem on general measure," *Journal of the London Mathematical Society*, vol. 1, no. 4, pp. 291–300, 1946.

[19] J. Matoušek, *Lectures on Discrete Geometry*. Springer, 2002.

[20] M. Shabbir, J. Li, W. Abbas, and X. Koutsoukos, "Resilient vector consensus in multi-agent networks using centerpoints," 2020. [Online]. Available: https://arxiv.org/abs/2003.05497

[21] S. Jadhav and A. Mukhopadhyay, "Computing a centerpoint of a finite planar set of points in linear time," *Discrete & Computational Geometry*, vol. 12, no. 3, pp. 291–312, 1994.

[22] N. Megiddo, "Partitioning with two lines in the plane," *Journal of Algorithms*, vol. 6, no. 3, pp. 430 – 433, 1985.

[23] T. M. Chan, "An optimal randomized algorithm for maximum tukey depth," in *Proceedings of the 15th annual ACM-SIAM Symposium on Discrete Slgorithms (SODA)*. SIAM, 2004, pp. 430–436.

[24] B. Chazelle, "Cutting hyperplanes for divide-and-conquer," *Discrete & Computational Geometry*, vol. 9, no. 2, pp. 145–158, 1993.

[25] G. L. Miller and D. R. Sheehy, "Approximate centerpoints with proofs," *Computational Geometry*, vol. 43, no. 8, pp. 647–654, 2010.