Iterative Byzantine Vector Consensus in Incomplete Graphs

Nitin H. Vaidya

Department of Electrical and Computer Engineering,
University of Illinois at Urbana-Champaign
nhv@illinois.edu

Abstract. This work addresses *Byzantine vector consensus*, wherein the input at each process is a d-dimensional vector of reals, and each process is required to decide on a *decision vector* that is in the *convex hull* of the input vectors at the fault-free processes [9,12]. The input *vector* at each process may also be viewed as a *point* in the d-dimensional Euclidean space $\mathbf{R}^d$, where $d > 0$ is a finite integer. Recent work [9,12] has addressed Byzantine vector consensus, and presented algorithms with optimal fault tolerance in complete graphs. This paper considers Byzantine vector consensus in incomplete graphs using *a restricted class* of iterative algorithms that maintain only a small amount of memory across iterations. For such algorithms, we prove a necessary condition, and a sufficient condition, for the graphs to be able to solve the vector consensus problem iteratively. We present an iterative Byzantine vector consensus algorithm, and prove it correct under the sufficient condition. The necessary condition presented in this paper for vector consensus does not match with the sufficient condition for $d > 1$; thus, a weaker condition may potentially suffice for Byzantine vector consensus.

1 Introduction

This work addresses *Byzantine vector consensus* (BVC), wherein the input at each process is a d-dimensional vector consisting of d real numbers, and each process is required to decide on a *decision vector* that is in the *convex hull* of the input vectors at the fault-free processes [9,12]. The input *vector* at each process may also be viewed as a *point* in the d-dimensional Euclidean space $\mathbf{R}^d$, where $d > 0$ is a finite integer. Due to this correspondence, we use the terms *point* and *vector* interchangeably. Recent work [9,12] has addressed Byzantine vector consensus, and presented algorithms with optimal fault tolerance in complete graphs. The correctness conditions for Byzantine vector consensus (elaborated below) cannot be satisfied by independently performing consensus on each element of the input vectors; therefore, new algorithms are necessary [9,12].

In this paper, we consider Byzantine vector consensus in incomplete graphs using *a restricted class* of iterative algorithms that maintain only a small amount of memory across iterations. We prove a necessary condition, and a sufficient condition, for the graphs to be able to solve the vector consensus problem using

M. Chatterjee et al. (Eds.): ICDCN 2014, LNCS 8314, pp. 14–28, 2014.

such restricted algorithms. We present an iterative Byzantine vector consensus algorithm, and prove it correct under the sufficient condition; our proof of correctness follows a structure previously used in our work to prove correctness of other consensus algorithms [11,14]. The use of matrix analysis tools in our proofs is inspired by the prior work on non-fault tolerant consensus (e.g., [6]). For lack of space, the proofs of most claims in the paper are omitted here. Further details can be found in [15]. The necessary condition presented in this paper for vector consensus does not match with the sufficient condition for $d > 1$; thus, it is possible that a weaker condition may also suffice for Byzantine vector consensus.

This paper extends our past work on *scalar* consensus (i.e., consensus with scalar inputs) in incomplete graphs in presence of Byzantine faults [13], using similarly restricted iterative algorithms. The work in [13] yielded an exact characterization of graphs in which the scalar Byzantine consensus problem is solvable.

Related Work: Approximate consensus has been previously explored in synchronous as well as asynchronous systems. Dolev et al. [3] were the first to consider approximate consensus in presence of Byzantine faults in asynchronous systems. Subsequently, for complete graphs, Abraham, Amit and Dolev [1] established that approximate Byzantine consensus is possible in asynchronous systems if $n \geq 3f+1$. Other algorithms for approximate consensus in presence of Byzantine faults have also been proposed (e.g., [4]). Scalar consensus in incomplete graphs under a *malicious* fault model in which the faulty nodes are restricted to sending identical messages to their neighbors has also been explored by other researchers (e.g., [7,8]).

The paper is organized as follows. Section 2 presents our system model. The iterative algorithm structure considered in our work is presented in Section 3. Section 4 presents a necessary condition, and Section 5 presents a sufficient condition. Section 5 also presents an iterative algorithm and proves its correctness under the sufficient condition. The paper concludes with a summary in Section 6.

2 System Model

The system is assumed to be *synchronous*.[1] The communication network is modeled as a simple *directed* graph $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, n\}$ is the set of n processes, and $\mathcal{E}$ is the set of directed edges between the processes in $\mathcal{V}$. Thus, $|\mathcal{V}| = n$. We assume that $n \geq 2$, since the consensus problem for $n = 1$ is trivial. Process i can reliably transmit messages to process j, $j \neq i$, if and only if the directed edge (i, j) is in $\mathcal{E}$. Each process can send messages to itself as well, however, for convenience of presentation, we exclude self-loops from set $\mathcal{E}$. That is, $(i, i) \notin \mathcal{E}$ for $i \in \mathcal{V}$. We will use the terms *edge* and *link* interchangeably.

For each process i, let N_i^- be the set of processes from which i has incoming edges. That is, $N_i^- = \{\, j \mid (j, i) \in \mathcal{E} \,\}$. Similarly, define N_i^+ as the set of processes to which process i has outgoing edges. That is, $N_i^+ = \{\, j \mid (i, j) \in \mathcal{E} \,\}$. Since

[1] Analogous results can be similarly derived for asynchronous systems, using the asynchronous algorithm structure presented in [13] for the case of $d = 1$.

we exclude self-loops from $\mathcal{E}$, $i \notin N_i^-$ and $i \notin N_i^+$. However, we note again that each process can indeed send messages to itself.

We consider the Byzantine failure model, with up to f processes becoming faulty. A faulty process may *misbehave* arbitrarily. The faulty processes may potentially collaborate with each other. Moreover, the faulty processes are assumed to have a complete knowledge of the execution of the algorithm, including the states of all the processes, contents of messages the other processes send to each other, the algorithm specification, and the network topology.

We use the notation $|X|$ to denote the size of a set or a multiset, and the notation $\|x\|$ to denote the absolute value of a real number x.

3 Byzantine Vector Consensus and Iterative Algorithms

Byzantine vector consensus: We are interested in iterative algorithms that satisfy the following conditions in presence of up to f Byzantine faulty processes:

- *Termination*: Each fault-free process must terminate after a finite number of iterations.
- *Validity*: The state of each fault-free process at the end of *each iteration* must be in the convex hull of the d-dimensional input vectors at the fault-free processes.
- ϵ-*Agreement*: When the algorithm terminates, the l-th elements of the decision vectors at any two fault-free processes, where $1 \leq l \leq d$, must be within ϵ of each other, where $\epsilon > 0$ is a pre-defined constant.

Any information carried over by a process from iteration t to iteration $t+1$ is considered the state of process t at the end of iteration t. The above *validity* condition forces the algorithms to maintain "minimal" state, for instance, precluding the possibility of remembering messages received in several of the past iterations, or remembering the history of detected misbehavior of the neighbors. We focus on such restricted algorithms with the iterative structure below.

Iterative structure: Each process i maintains a state variable $\mathbf{v}_i$, which is a d-dimensional vector. The initial state of process i is denoted as $\mathbf{v}_i[0]$, and it equals the *input* provided to process i. For $t \geq 1$, $\mathbf{v}_i[t]$ denotes the state of process i at the *end* of the t-th iteration of the algorithm. At the *start* of the t-th iteration ($t \geq 1$), the state of process i is $\mathbf{v}_i[t-1]$. The iterative algorithms of interest will require each process i to perform the following three steps in the t-th iteration. Each "value" referred in the algorithm below is a d-dimensional vector (or, equivalently, a point in the d-dimensional Euclidean space).

1. *Transmit step:* Transmit current state, namely $\mathbf{v}_i[t-1]$, on all outgoing edges to processes in N_i^+.
2. *Receive step:* Receive values on all incoming edges from processes in N_i^-. Denote by $r_i[t]$ the multiset[2] of values received by process i from its neighbors. The size of multiset $r_i[t]$ is $|N_i^-|$.

[2] The same value may occur multiple times in a multiset.

3. *Update step:* Process i updates its state using a transition function T_i as follows. T_i is a part of the specification of the algorithm, and takes as input the multiset $r_i[t]$ and state $\mathbf{v}_i[t-1]$.

$$\mathbf{v}_i[t] = T_i \left(r_i[t] \, , \, \mathbf{v}_i[t-1] \right) \tag{1}$$

The decision (or output) of each process equals its state when the algorithm terminates.

We assume that each element of the input vector at each fault-free process is lower bounded by a constant μ and upper bounded by a constant U. The iterative algorithm may terminate after a number of rounds that is a function of μ and U. μ and U are assumed to be known *a priori*. This assumption holds in many practical systems, because the input vector elements represent quantities that are constrained. For instance, if the input vectors represent locations in 3-dimensional space occupied by mobile robots, then U and μ are determined by the boundary of the region in which the robots are allowed to operate [12].

In Section 4, we develop a necessary condition that the graph $G(\mathcal{V}, \mathcal{E})$ must satisfy in order for the Byzantine vector consensus algorithm to be solvable using the above iterative structure. In Section 5, we develop a sufficient condition, such that the Byzantine vector consensus algorithm is solvable using the above iterative structure in any graph that satisfies this condition. We present an iterative algorithm, and prove its correctness under the sufficient condition.

4 A Necessary Condition

Hereafter, when we say that an algorithm solves Byzantine vector consensus, we mean that the algorithm satisfies the termination, validity and ϵ-agreement conditions stated above. Thus, the state the algorithm can carry across iterations is restricted by the above validity condition. Also, hereafter when we refer to an iterative algorithm, we mean an algorithm with the structure specified in the previous section. In this section, we state a necessary condition on graph $G(\mathcal{V}, \mathcal{E})$ to be able to solve Byzantine vector consensus. First we present three definitions.

Definition 1

- *Define* $\mathbf{e}_0$ *to be a* d*-dimensional vector with all its elements equal to 0. Thus,* $\mathbf{e}_0$ *corresponds to the origin in the* d*-dimensional Euclidean space.*
- *Define* $\mathbf{e}_i$, $1 \leq i \leq d$, *to be a* d*-dimensional vector with the* i*-th element equal to* 2ϵ*, and the remaining elements equal to 0. Recall that* ϵ *is the parameter of the* ϵ*-agreement condition.*

Definition 2. *For non-empty disjoint sets of processes* A *and* B*, and a non-negative integer* c*,*

- $A \xrightarrow{c} B$ *if and only if there exists a process* $v \in B$ *that has at least* $c+1$ *incoming edges from processes in* A*, i.e.,* $|N_v^- \cap A| \geq c+1$.
- $A \not\xrightarrow{c} B$ *iff* $A \xrightarrow{c} B$ *is* not *true.*

Definition 3. *$\mathcal{H}(X)$ denotes the convex hull of a multiset of points X.*

Now we state the necessary condition.

Condition NC: For any partition $V_0, V_1, \cdots, V_p, C, F$ of set $\mathcal{V}$, where $1 \leq p \leq d$, $V_k \neq \emptyset$ for $0 \leq k \leq p$, and $|F| \leq f$, there exist i, j $(0 \leq i, j \leq p, i \neq j)$, such that

$$V_i \cup C \xrightarrow{f} V_j$$

That is, there are $f+1$ incoming links from processes in $V_i \cup C$ to some process in V_j.

The proof of the necessary condition below extends the proofs of necessary conditions in [9,12,13].

Lemma 1. *If the Byzantine vector consensus problem can be solved using an iterative algorithm in $G(\mathcal{V}, \mathcal{E})$, then $G(\mathcal{V}, \mathcal{E})$ satisfies Condition NC.*

Proof. The proof is by contradiction. Suppose that Condition NC is not true. Then there exists a certain partition $V_0, V_1, \cdots, V_p, C, F$ such that $V_k \neq \emptyset$ $(1 \leq k \leq p)$, $|F| \leq f$, and for $0 \leq i, k \leq p$, $V_k \cup C \not\xrightarrow{f} V_i$.

Let the initial state of each process in V_i be $\mathbf{e}_i$ $(0 \leq i \leq p)$. Suppose that all the processes in set F are faulty. For each link (j, k) such that $j \in F$ and $k \in V_i$ $(0 \leq i \leq p)$, the faulty process j sends value $\mathbf{e}_i$ to process j in each iteration.

We now prove by induction that if the iterative algorithm satisfies the validity condition then the state of each fault-free process $j \in V_i$ at the start of iteration t equals $\mathbf{e}_i$, for all $t > 0$. The claim is true for $t = 1$ by assumption on the inputs at the fault-free processes. Now suppose that the claim is true through iteration t, and prove it for iteration $t+1$. Thus, the state of each fault-free process in V_i at the start of iteration t equals $\mathbf{e}_i$, $0 \leq i \leq p$.

Consider any fault-free process $j \in V_i$, where $0 \leq i \leq p$. In iteration t, process j will receive $\mathbf{v}_g[t-1]$ from each fault-free incoming neighbor g, and receive $\mathbf{e}_i$ from each faulty incoming neighbor. These received values form the multiset $r_j[t]$. Since the condition in the lemma is assumed to be false, for any $k \neq i$, $0 \leq k \leq p$, we have

$$V_k \cup C \not\xrightarrow{f} V_i.$$

Thus, at most f incoming neighbors of j belong to $V_k \cup C$, and therefore, at most f values in $r_j[t]$ equal $\mathbf{e}_k$.

Since process j does not know which of its incoming neighbors, if any, are faulty, it must allow for the possibility that any of its f incoming neighbors are faulty. Let $A_k \subseteq V_k \cup C$, $k \neq i$, be the set containing all the incoming neighbors of process j in $V_k \cup C$. Since $V_k \cup C \not\xrightarrow{f} V_i$, $|A_k| \leq f$; therefore, all the processes in A_k are *potentially* faulty. Also, by assumption, the values received from all fault-free processes equal their input, and the values received from faulty processes in F equal $\mathbf{e}_i$. Thus, due to the validity condition, process j must choose as its new state a value that is in the convex hull of the set

$$S_k = \{\mathbf{e}_m \mid m \neq k, 0 \leq m \leq p\}.$$

where $k \neq i$. Since this observation is true for each $k \neq i$, it follows that the new state $\mathbf{v}_j[t]$ must be a point in the convex hull of

$$\cap_{1 \leq k \leq p,\ k \neq i}\ \mathcal{H}(S_k).$$

It is easy to verify that the above intersection only contains the point $\mathbf{e}_i$. Therefore, $\mathbf{v}_j[t] = \mathbf{e}_i$. Thus, the state of process j at the start of iteration $t+1$ equals $\mathbf{e}_i$. This concludes the induction.

The above result implies that the state of each fault-free process remains unchanged through the iterations. Thus, the state of any two fault-free processes differs in at least one vector element by 2ϵ, precluding ϵ-agreement. □

The above lemma demonstrates the necessity of Condition NC. Necessary condition NC implies a lower bound on the number of processes $n = |\mathcal{V}|$ in $G(\mathcal{V}, \mathcal{E})$, as stated in the next lemma.

Lemma 2. *Suppose that the Byzantine vector consensus problem can be solved using an iterative algorithm in $G(\mathcal{V}, \mathcal{E})$. Then, $n \geq (d+2)f + 1$.*

Proof. Since the Byzantine vector consensus problem can be solved using an iterative algorithm in $G(\mathcal{V}, \mathcal{E})$, by Lemma 1, graph G must satisfy Condition NC. Suppose that $2 \leq |\mathcal{V}| = n \leq (d+2)f$. Then there exists p, $1 \leq p \leq d$, such that we can partition $\mathcal{V}$ into sets $V_0, ..., V_p, F$ such that for each V_i, $0 < |V_i| \leq f$, and $|F| \leq f$. Define $C = \emptyset$. Since $|C \cup V_i| \leq f$ for each i, it is clear that this partition of $\mathcal{V}$ cannot satisfy Condition NC. This is a contradiction. □

When $d = 1$, the input at each process is a scalar. For the $d = 1$ case, our prior work [13] yielded a tight necessary and sufficient condition for Byzantine consensus to be achievable in $G(\mathcal{V}, \mathcal{E})$ using iterative algorithms. For $d = 1$, the necessary condition stated in Lemma 1 is equivalent to the necessary condition in [13]. We previously showed that, for $d = 1$, the same condition is also sufficient [13]. However, in general, for $d > 1$, Condition NC is not proved sufficient. Instead, we prove the sufficiency of another condition stated in the next section.

5 A Sufficient Condition

We now present Condition SC that is later proved to be sufficient for achieving Byzantine vector consensus in graph $G(\mathcal{V}, \mathcal{E})$ using an iterative algorithm. Condition SC is a generalization of the sufficient condition presented in [13] for $d = 1$.

Condition SC: For any partition F, L, C, R of set $\mathcal{V}$, such that L and R are both non-empty, and $|F| \leq f$, at least one of these conditions is true: $R \cup C \xrightarrow{df} L$, or $L \cup C \xrightarrow{df} R$.

Later in the paper we will present a Byzantine vector consensus algorithm named `Byz-Iter` that is proved correct in all graphs that saitsfy Condition SC. The proof makes use of Lemmas 3 and 4 presented below.

Lemma 3. *For $f > 0$, if graph $G(\mathcal{V}, \mathcal{E})$ satisfies Condition SC, then in-degree of each process in $\mathcal{V}$ must be at least $(d+1)f + 1$. That is, for each $i \in \mathcal{V}$, $|N_i^-| \geq (d+1)f + 1$.*

Proof. The proof is by contradiction. As per the assumption in the lemma, $f > 0$, and graph $G(\mathcal{V}, \mathcal{E})$ satisfies condition SC.

Suppose that some process i has in-degree at most $(d+1)f$. Define $L = \{i\}$, and $C = \emptyset$. Partition the processes in $\mathcal{V} - \{i\}$ into sets R and F such that $|F| \leq f$, $|F \cap N_i^-| \leq f$ and $|R \cap N_i^-| \leq df$. Such sets R and F exist because in-degree of process i is at most $(d+1)f$. L, R, C, F thus defined form a partition of $\mathcal{V}$.

Now, $f > 0$ and $d \geq 1$, and $|L \cup C| = 1$. Thus, there can be at most 1 link from $L \cup C$ to any process in R, and $1 \leq df$. Therefore, $L \cup C \stackrel{df}{\not\longrightarrow} R$. Also, because $C = \emptyset$, $|(R \cup C) \cap N_i^-| = |R \cap N_i^-| \leq df$. Thus, there can be at most df links from $R \cup C$ to process i, which is the only process in $L = \{i\}$. Therefore, $R \cup C \stackrel{df}{\not\longrightarrow} L$. Thus, the above partition of $\mathcal{V}$ does not satisfy Condition SC. This is a contradiction. □

Definition 4. Reduced Graph: *For a given graph $G(\mathcal{V}, \mathcal{E})$ and $\mathcal{F} \subset \mathcal{V}$ such that $|\mathcal{F}| \leq f$, a graph $H(\mathcal{V}_\mathcal{F}, \mathcal{E}_\mathcal{F})$ is said to be a* reduced graph, *if: (i) $\mathcal{V}_\mathcal{F} = \mathcal{V} - \mathcal{F}$, and (ii) $\mathcal{E}_\mathcal{F}$ is obtained by first removing from $\mathcal{E}$ all the links incident on the processes in $\mathcal{F}$, and* then *removing up to df additional incoming links at each process in $\mathcal{V}_\mathcal{F}$.*

Note that for a given $G(\mathcal{V}, \mathcal{E})$ and a given $\mathcal{F}$, multiple reduced graphs may exist (depending on the choice of the links removed at each process).

Lemma 4. *Suppose that graph $G(\mathcal{V}, \mathcal{E})$ satisfies Condition SC, and $\mathcal{F} \subset \mathcal{V}$, such that $|\mathcal{F}| \leq f$. Then, in any reduced graph $H(\mathcal{V}_\mathcal{F}, \mathcal{E}_\mathcal{F})$, there exists a process that has a directed path to all the remaining processes in $\mathcal{V}_\mathcal{F}$.*

The proof of Lemma 4 is omitted for lack of space. This proof, and the other omitted proofs in the paper, are presented in [15].

5.1 Algorithm `Byz-Iter`

We prove that, if graph $G(\mathcal{V}, \mathcal{E})$ satisfies Condition SC, then Algorithm `Byz-Iter` presented below achieves Byzantine vector consensus. Algorithm `Byz-Iter` has the three-step structure described in Section 3.

The proposed algorithm is based on the following result by Tverberg [10].

Theorem 1. (Tverberg's Theorem [10]) *For any integer $f \geq 0$, and for every multiset Y containing at least $(d+1)f + 1$ points in $\mathbf{R}^d$, there exists a partition $Y_1, \cdots, Y_{f+1}$ of Y into $f+1$ non-empty multisets such that $\cap_{l=1}^{f+1} \mathcal{H}(Y_l) \neq \emptyset$.*

The points in Y above need not be distinct [10]; thus, the same point may occur multiple times in Y, and also in each of its subsets (Y_l's) above. The partition in Theorem 1 is called a *Tverberg partition*, and the points in $\cap_{l=1}^{f+1} \mathcal{H}(Y_l)$ in Theorem 1 are called Tverberg points.

Algorithm. `Byz-Iter`

Each iteration consists of three steps: *Transmit*, *Receive*, and *Update*:

1. *Transmit step:* Transmit current state $\mathbf{v}_i[t-1]$ on all outgoing edges.
2. *Receive step:* Receive values on all incoming edges. These values form multiset $r_i[t]$ of size $|N_i^-|$. (If a message is not received from some incoming neighbor, then that neighbor must be faulty. In this case, the missing message value is assumed to be $\mathbf{e}_0$ by default. Recall that we assume a *synchronous* system.)
3. *Update step:* Form a multiset $Z_i[t]$ using the steps below:
   - Initialize $Z_i[t]$ as empty.
   - Add to $Z_i[t]$, any one *Tverberg point* corresponding to *each* multiset $C \subseteq r_i[t]$ such that $|C| = (d+1)f+1$. Since $|C| = (d+1)f+1$, by Theorem 1, such a Tverberg point exists.

   $Z_i[t]$ is a multiset; thus a single point may appear in $Z_i[t]$ more than once. Note that $|Z_i[t]| = \binom{|r_i[t]|}{(d+1)f+1} \leq \binom{n}{(d+1)f+1}$. Compute new state $\mathbf{v}_i[t]$ as:

$$\mathbf{v}_i[t] = \frac{\mathbf{v}_i[t-1] + \sum_{\mathbf{z} \in Z_i[t]} \mathbf{z}}{1 + |Z_i[t]|} \tag{2}$$

Termination: Each fault-free process terminates after completing t_{end} iterations, where t_{end} is a constant defined later in Equation (10). The value of t_{end} depends on graph $G(\mathcal{V}, \mathcal{E})$, constants U and μ defined earlier in Section 3 and parameter ϵ of ϵ-agreement.

The proof of correctness of Algorithm `Byz-Iter` makes use of a matrix representation of the algorithm's behavior. Before presenting the matrix representation, we introduce some notations and definitions related to matrices.

5.2 Matrix Preliminaries

We use boldface letters to denote matrices, rows of matrices, and their elements. For instance, $\mathbf{A}$ denotes a matrix, $\mathbf{A}_i$ denotes the i-th row of matrix $\mathbf{A}$, and $\mathbf{A}_{ij}$ denotes the element at the intersection of the i-th row and the j-th column of matrix $\mathbf{A}$.

Definition 5. *A vector is said to be stochastic if all its elements are non-negative, and the elements add up to 1. A matrix is said to be row stochastic if each row of the matrix is a stochastic vector.*

For matrix products, we adopt the "backward" product convention below, where $a \leq b$,

$$\Pi_{\tau=a}^{b} \mathbf{A}[\tau] = \mathbf{A}[b]\mathbf{A}[b-1]\cdots\mathbf{A}[a] \tag{3}$$

For a row stochastic matrix $\mathbf{A}$, coefficients of ergodicity $\delta(\mathbf{A})$ and $\lambda(\mathbf{A})$ are defined as follows [17]:

$$\delta(\mathbf{A}) = \max_{j} \max_{i_1, i_2} \|\mathbf{A}_{i_1 j} - \mathbf{A}_{i_2 j}\|$$

$$\lambda(\mathbf{A}) = 1 - \min_{i_1, i_2} \sum_{j} \min(\mathbf{A}_{i_1 j}, \mathbf{A}_{i_2 j})$$

Lemma 5. *For any p square row stochastic matrices $\mathbf{A}(1), \mathbf{A}(2), \ldots, \mathbf{A}(p)$,*

$$\delta(\Pi_{\tau=1}^{p} \mathbf{A}(\tau)) \ \leq \ \Pi_{\tau=1}^{p} \ \lambda(\mathbf{A}(\tau)).$$

The above lemma is proved in [5]. The lemma below follows directly from the definition of $\lambda(\cdot)$.

Lemma 6. *If all the elements in any one column of matrix $\mathbf{A}$ are lower bounded by a constant γ, then $\lambda(\mathbf{A}) \leq 1 - \gamma$. That is, if $\exists g$, such that $\mathbf{A}_{ig} \geq \gamma$, $\forall i$, then $\lambda(\mathbf{A}) \leq 1 - \gamma$.*

5.3 Correctness of Algorithm `Byz-Iter`

Let $\mathcal{F}$ denote the actual set of faulty processes in a given execution of Algorithm `Byz-Iter`. Let $|\mathcal{F}| = \psi$. Thus, $0 \leq \psi \leq f$. Without loss of generality, suppose that processes 1 through $(n-\psi)$ are fault-free, and if $\psi > 0$, processes $(n-\psi+1)$ through n are faulty.

In the analysis below, it is convenient to view the state of each process as a point in the d-dimensional Euclidean space. Denote by $\mathbf{v}[0]$ the column vector consisting of the initial states of the $(n-\psi)$ fault-free processes. The i-th element of $\mathbf{v}[0]$ is $\mathbf{v}_i[0]$, the initial state of process i. Thus, $\mathbf{v}[0]$ is a vector consisting of $(n-\psi)$ points in the d-dimensional Euclidean space. Denote by $\mathbf{v}[t]$, for $t \geq 1$, the column vector consisting of the states of the $(n-\psi)$ fault-free processes at the end of the t-th iteration. The i-th element of vector $\mathbf{v}[t]$ is state $\mathbf{v}_i[t]$.

Lemma 7 below states the key result that helps in proving the correctness of Algorithm `Byz-Iter`. In particular, Lemma 7 allows us to use results for non-homogeneous Markov chains to prove the correctness of Algorithm `Byz-Iter`.

Lemma 7. *Suppose that graph $G(\mathcal{V}, \mathcal{E})$ satisfies Condition SC. Then the state updates performed by the fault-free processes in the t-th iteration ($t \geq 1$) of Algorithm* `Byz-Iter` *can be expressed as*

$$\mathbf{v}[t] = \mathbf{M}[t]\,\mathbf{v}[t-1] \tag{4}$$

where $\mathbf{M}[t]$ *is a* $(n-\psi)\times(n-\psi)$ *row stochastic matrix with the following property: there exists a reduced graph* $H[t]$*, and a constant* β *(*$0 < \beta \leq 1$*) that depends only on graph* $G(\mathcal{V}, \mathcal{E})$*, such that*

$$\mathbf{M}_{ij}[t] \ \geq \ \beta$$

if $j = i$ *or edge* (j, i) *is in* $H[t]$*.*

The proof is presented in [15].

Matrix $\mathbf{M}[t]$ above is said to be a transition matrix. As the lemma states, $\mathbf{M}[t]$ is a row stochastic matrix. The proof of Lemma 7 shows how to identify a suitable row stochastic matrix $\mathbf{M}[t]$ for each iteration t. The matrix $\mathbf{M}[t]$ depends on t, as well as the behavior of the faulty processes. $\mathbf{M}_i[t]$ is the i-th row of transition matrix $\mathbf{M}[t]$. Thus, (4) implies that

$$\mathbf{v}_i[t] \ = \ \mathbf{M}_i[t]\, \mathbf{v}[t-1]$$

That is, the state of any fault-free process i at the end of iteration t can be expressed as a convex combination of the state of just the fault-free processes at the end of iteration $t-1$. Recall that vector $\mathbf{v}$ only includes the state of fault-free processes.

Theorem 2. *Algorithm* `Byz-Iter` *satisfies the termination, validity and* ϵ*-agreement conditions.*

Proof. This proof follows a structure used to prove correctness of other consensus algorithms in our prior work [14,11]. Sections 5.4, 5.5 and 5.6 provide the proof that Algorithm `Byz-Iter` satisfies the three conditions for Byzantine vector consensus, and thus prove Theorem 2.

5.4 Algorithm `Byz-Iter` Satisfies the Validity Condition

Observe that $\mathbf{M}[t+1]\,(\mathbf{M}[t]\mathbf{v}[t-1]) \ = \ (\mathbf{M}[t+1]\mathbf{M}[t])\,\mathbf{v}[t-1]$. Therefore, by repeated application of (4), we obtain for $t \geq 1$,

$$\mathbf{v}[t] = \left(\mathit{\Pi}_{\tau=1}^{t} \mathbf{M}[\tau] \right) \mathbf{v}[0] \tag{5}$$

Since each $\mathbf{M}[\tau]$ is row stochastic, the matrix product $\mathit{\Pi}_{\tau=1}^{t}\mathbf{M}[\tau]$ is also a row stochastic matrix. Recall that vector $\mathbf{v}$ only includes the state of fault-free processes. Thus, (5) implies that the state of each fault-free process i at the end of iteration t can be expressed as a convex combination of the initial state of the fault-free processes. Therefore, the validity condition is satisfied.

5.5 Algorithm `Byz-Iter` Satisfies the Termination Condition

Algorithm `Byz-Iter` stops after a finite number (t_{end}) of iterations, where t_{end} is a constant that depends only on $G(\mathcal{V}, \mathcal{E})$, U, μ and ϵ. Therefore, trivially, the algorithm satisfies the termination condition. Later, using (10) we define a suitable value for t_{end}.

5.6 Algorithm Byz-Iter Satisfies the ϵ-Agreement Condition

The proof structure below is derived from our previous work [14] wherein we proved the correctness of an iterative algorithm for *scalar* Byzantine consensus (i.e., the case of $d = 1$), and its generalization to a broader class of fault sets [11].

Let R_F denote the set of all the reduced graph of $G(\mathcal{V}, \mathcal{E})$ corresponding to fault set F. Thus, $R_{\mathcal{F}}$ is the set of all the reduced graph of $G(\mathcal{V}, \mathcal{E})$ corresponding to actual fault set $\mathcal{F}$. Let

$$r = \max_{|F| \leq f} |R_F|.$$

r depends only on $G(\mathcal{V}, \mathcal{E})$ and f, and it is finite. Note that $|R_{\mathcal{F}}| \leq r$.

For each reduced graph $H \in R_{\mathcal{F}}$, define connectivity matrix $\mathbf{H}$ as follows, where $1 \leq i, j \leq n - \psi$:

- $\mathbf{H}_{ij} = 1$ if either $j = i$, or edge (j, i) exists in reduced graph H.
- $\mathbf{H}_{ij} = 0$, otherwise.

Thus, the non-zero elements of row $\mathbf{H}_i$ correspond to the incoming links at process i in the reduced graph H, and the self-loop at process i. Observe that $\mathbf{H}$ has a non-zero diagonal.

Lemma 8. *For any $H \in R_F$, and any $k \geq n - \psi$, matrix product $\mathbf{H}^k$ has at least one non-zero column (i.e., a column with all elements non-zero).*

Proof. Each reduced graph contains $n - \psi$ processes because the fault set $\mathcal{F}$ contain ψ processes. By Lemma 4, at least one process in the reduced graph, say process p, has directed paths to all the processes in the reduced graph H. Element $\mathbf{H}^k_{jp}$ of matrix product $\mathbf{H}^k$ is 1 if and only if process p has a directed path to process j containing at most k edges; each of these directed paths must contain less than $n - \psi$ edges, because the number of processes in the reduced graph is $n - \psi$. Since p has directed paths to all the processes, it follows that, when $k \geq n - \psi$, all the elements in the p-th column of $\mathbf{H}^k$ must be non-zero.

For matrices $\mathbf{A}$ and $\mathbf{B}$ of identical dimensions, we say that $\mathbf{A} \leq \mathbf{B}$ if and only if $\mathbf{A}_{ij} \leq \mathbf{B}_{ij}, \forall i, j$. Lemma 9 relates the transition matrices with the connectivity matrices. Constant β used in the lemma below was introduced in Lemma 7.

Lemma 9. *For any $t \geq 1$, there exists a reduced graph $H[t] \in R_{\mathcal{F}}$ such that $\beta \mathbf{H}[\mathbf{t}] \leq \mathbf{M}[t]$, where $\mathbf{H}[t]$ is the connectivity matrix for $H[t]$.*

The proof is presented in [15].

Lemma 10. *At least one column in the matrix product $\Pi_{t=u}^{u+r(n-\psi)-1} \mathbf{H}[t]$ is non-zero.*

The proof is presented in [15].

Let us now define a sequence of matrices $\mathbf{Q}(i)$, $i \geq 1$, such that each of these matrices is a product of $r(n-\psi)$ of the $\mathbf{M}[t]$ matrices. Specifically,

$$\mathbf{Q}(i) = \Pi_{t=(i-1)r(n-\psi)+1}^{ir(n-\psi)} \mathbf{M}[t] \tag{6}$$

From (5) and (6) observe that

$$\mathbf{v}[kr(n-\psi)] = \left(\Pi_{i=1}^{k} \mathbf{Q}(i) \right) \mathbf{v}[0] \tag{7}$$

Lemma 11. *For $i \geq 1$, $\mathbf{Q}(i)$ is a row stochastic matrix, and*

$$\lambda(\mathbf{Q}(i)) \leq 1 - \beta^{r(n-\psi)}.$$

The proof is presented in [15].

Let us now continue with the proof of ϵ-agreement. Consider the coefficient of ergodicity $\delta(\Pi_{i=1}^{t}\mathbf{M}[i])$.

$$\begin{aligned}
\delta(\Pi_{i=1}^{t}\mathbf{M}[i]) &= \delta\left(\left(\Pi_{i=\lfloor \frac{t}{r(n-\psi)} \rfloor r(n-\psi)+1}^{t}\mathbf{M}[i]\right)\left(\Pi_{i=1}^{\lfloor \frac{t}{r(n-\psi)} \rfloor}\mathbf{Q}(i)\right)\right) && (8)\\
&\quad \text{by definition of } Q(i)\\
&\leq \lambda\left(\Pi_{i=\lfloor \frac{t}{r(n-\psi)} \rfloor r(n-\psi)+1}^{t}\mathbf{M}[i]\right)\Pi_{i=1}^{\lfloor \frac{t}{r(n-\psi)} \rfloor}\lambda(\mathbf{Q}(i)) \quad \text{by Lemma 5}\\
&\leq \Pi_{i=1}^{\lfloor \frac{t}{r(n-\psi)} \rfloor}\lambda(\mathbf{Q}(i)) \quad \text{because } \lambda(.) \leq 1\\
&\leq \left(1-\beta^{r(n-\psi)}\right)^{\lfloor \frac{t}{r(n-\psi)} \rfloor} \quad \text{by Lemma 11}\\
&\leq (1-\beta^{rn})^{\lfloor \frac{t}{rn} \rfloor} \quad \text{because } 0 < \beta \leq 1 \text{ and } 0 \leq \psi < n. && (9)
\end{aligned}$$

Observe that the upper bound on right side of (9) depends only on graph $G(\mathcal{V}, \mathcal{E})$ and t, and is independent of the input vectors, the fault set $\mathcal{F}$, and the behavior of the faulty processes. Also, the upper bound on the right side of (9) is a non-increasing function of t. Define t_{end} as the smallest positive integer t for which the right hand side of (9) is smaller than $\frac{\epsilon}{n\max(\|U\|,\|\mu\|)}$, where $\|x\|$ denotes the absolute value of real number x. Thus,

$$\delta(\Pi_{i=1}^{t_{end}}\mathbf{M}[i]) \;\leq\; (1-\beta^{rn})^{\lfloor \frac{t_{end}}{rn} \rfloor} \;<\; \frac{\epsilon}{n\max(\|U\|,\|\mu\|)} \tag{10}$$

Recall that β and r depend only on $G(\mathcal{V}, \mathcal{E})$. Thus, t_{end} depends only on graph $G(\mathcal{V}, \mathcal{E})$, and constants U, μ and ϵ.

Recall that $\Pi_{i=1}^{t}\mathbf{M}[i]$ is a $(n-\psi) \times (n-\psi)$ row stochastic matrix. Let $\mathbf{M}^* = \Pi_{i=1}^{t}\mathbf{M}[i]$. From (5) we know that state $\mathbf{v}_j[t]$ of any fault-free process j is obtained as the product of the j-th row of $\Pi_{i=1}^{t}\mathbf{M}[i]$ and $\mathbf{v}[0]$. That is, $\mathbf{v}_j[t] = \mathbf{M}_j^*\mathbf{v}[0]$.

Recall that $\mathbf{v}_j[t]$ is a d-dimensional vector. Let us denote the l-th element of $\mathbf{v}_j[t]$ as $\mathbf{v}_j[t](l)$, $1 \leq l \leq d$. Also, by $\mathbf{v}[0](l)$, let us denote a vector consisting of the l-th elements of $\mathbf{v}_i[0], \forall i$. Then by the definitions of $\delta(.)$, U and μ, for any two fault-free processes j and k, we have

$$\begin{aligned}
\|\mathbf{v}_j[t](l) - \mathbf{v}_k[t](l)\| &= \|\mathbf{M}_j^* \mathbf{v}[0](l) - \mathbf{M}_k^* \mathbf{v}[0](l)\| && (11)\\
&= \|\sum_{i=1}^{n-\psi} \mathbf{M}_{ji}^* \mathbf{v}_i[0](l) - \sum_{i=1}^{n-\psi} \mathbf{M}_{ki}^* \mathbf{v}_i[0](l)\| && (12)\\
&= \|\sum_{i=1}^{n-\psi} \left(\mathbf{M}_{ji}^* - \mathbf{M}_{ki}^*\right) \mathbf{v}_i[0](l)\| && (13)\\
&\leq \sum_{i=1}^{n-\psi} \|\mathbf{M}_{ji}^* - \mathbf{M}_{ki}^*\| \, \|\mathbf{v}_i[0](l)\| && (14)\\
&\leq \sum_{i=1}^{n-\psi} \delta(\mathbf{M}^*) \|\mathbf{v}_i[0](l)\| && (15)\\
&\leq (n-\psi)\delta(\mathbf{M}^*) \max(\|U\|, \|\mu\|) && (16)\\
&\leq (n-\psi) \max(\|U\|, \|\mu\|) \, \delta(\Pi_{i=1}^{t} \mathbf{M}[i])\\
&\leq n \max(\|U\|, \|\mu\|) \, \delta(\Pi_{i=1}^{t} \mathbf{M}[i]) \text{ because } \psi \leq n && (17)
\end{aligned}$$

Therefore, by (10) and (17),

$$\|\mathbf{v}_i[t_{end}](l) - \mathbf{v}_j[t_{end}](l)\| < \epsilon, \qquad 1 \leq l \leq d. \tag{18}$$

The output of a fault-free process equals its state at termination (after t_{end} iterations). Thus, (18) implies that Algorithm `Byz-Iter` satisfies the ϵ-agreement condition.

6 Summary

This paper addresses *Byzantine vector consensus* (BVC), wherein the input at each process is a d-dimensional vector of reals, and each process is expected to decide on a *decision vector* that is in the *convex hull* of the input vectors at the fault-free processes [9,12]. We address a particular class of *iterative* algorithms in *incomplete* graphs, and prove a necessary condition (NC), and a sufficient condition (SC), for the graphs to be able to solve the vector consensus problem iteratively. This paper extends our past work on *scalar* consensus (i.e., $d = 1$) in incomplete graphs in presence of Byzantine faults [13,14], which yielded an exact characterization of graphs in which the problem is solvable for $d = 1$. However, the necessary condition NC presented in the paper for *vector* consensus does not match with the sufficient condition SC. We hope that this paper will motivate further work on identifying the tight sufficient condition.

Acknowledgments. The results presented in the paper are generalizations of results in prior work [12,13,16] performed in collaboration with Vijay Garg, Guanfeng Liang and Lewis Tseng. The author has benefited from the discussions with his collaborators.

This research is supported in part by National Science Foundation award CNS-1059540 and Army Research Office grant W-911-NF-0710287. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies or the U.S. government.

References

1. Abraham, I., Amit, Y., Dolev, D.: Optimal resilience asynchronous approximate agreement. In: Higashino, T. (ed.) OPODIS 2004. LNCS, vol. 3544, pp. 229–239. Springer, Heidelberg (2005)
2. Dasgupta, S., Papadimitriou, C., Vazirani, U.: Algorithms. McGraw-Hill Higher Education (2006)
3. Dolev, D., Lynch, N.A., Pinter, S.S., Stark, E.W., Weihl, W.E.: Reaching approximate agreement in the presence of faults. J. ACM 33, 499–516 (1986)
4. Fekete, A.D.: Asymptotically optimal algorithms for approximate agreement. In: Proceedings of the Fifth Annual ACM Symposium on Principles of Distributed Computing, PODC 1986, pp. 73–87. ACM, New York (1986)
5. Hajnal, J.: Weak ergodicity in non-homogeneous markov chains. Proceedings of the Cambridge Philosophical Society 54, 233–246 (1958)
6. Jadbabaie, A., Lin, J., Morse, A.S.: Coordination of groups of mobile autonomous agents using nearest neighbor rules. IEEE Transactions on Automatic Control 48, 988–1001 (2003)
7. LeBlanc, H., Koutsoukos, X.: Consensus in networked multi-agent systems with adversaries. In: 14th International Conference on Hybrid Systems: Computation and Control, HSCC (2011)
8. LeBlanc, H., Koutsoukos, X.: Low complexity resilient consensus in networked multi-agent systems with adversaries. In: 15th International Conference on Hybrid Systems: Computation and Control, HSCC (2012)
9. Mendes, H., Herlihy, M.: Multidimensional approximate agreement in byzantine asynchronous systems. In: 45th ACM Symposium on the Theory of Computing, STOC (June 2013)
10. Perles, M.A., Sigron, M.: A generalization of Tverberg's theorem, CoRR (2007), `http://arxiv.org/abs/0710.4668`
11. Tseng, L., Vaidya, N.: Iterative approximate byzantine consensus under a generalized fault model. In: Frey, D., Raynal, M., Sarkar, S., Shyamasundar, R.K., Sinha, P. (eds.) ICDCN 2013. LNCS, vol. 7730, pp. 72–86. Springer, Heidelberg (2013)
12. Vaidya, N.H., Garg, V.K.: Byzantine vector consensus in complete graphs. In: ACM Symposium on Principles of Distributed Computing (PODC) (July 2013)
13. Vaidya, N.H., Tseng, L., Liang, G.: Iterative approximate byzantine consensus in arbitrary directed graphs. In: ACM Symposium on Principles of Distributed Computing (PODC) (July 2012)

14. Vaidya, N.H.: Matrix representation of iterative approximate byzantine consensus in directed graphs. CoRR (March 2012), http://arxiv.org/abs/1203.1888
15. Vaidya, N.H.: Iterative Byzantine vector consensus in incomplete graphs. CoRR (July 2013), http://arxiv.org/abs/1307.2483
16. Vaidya, N.H., Garg, V.: Byzantine vector consensus in complete graphs. CoRR (July 2013), http://arxiv.org/abs/1302.2543
17. Wolfowitz, J.: Products of indecomposable, aperiodic, stochastic matrices. In: Proceedings of the American Mathematical Society, pp. 733–737 (1963)