Asynchronous Convex Consensus in the Presence of Crash Faults *

Lewis Tseng[1], and Nitin Vaidya[2]

[1] Department of Computer Science,
[2] Department of Electrical and Computer Engineering,
University of Illinois at Urbana-Champaign

Email: {ltseng3, nhv}@illinois.edu
Phone: +1 217-244-6024, +1 217-265-5414
Mailing address: Coordinated Science Lab., 1308 West Main St., Urbana, IL 61801, U.S.A.

February 13, 2014

Abstract

This paper defines a new consensus problem, *convex consensus*. Similar to *vector consensus* [13, 20, 19], the input at each process is a d-dimensional vector of reals (or, equivalently, a point in the d-dimensional Euclidean space). However, for convex consensus, the output at each process is a *convex polytope* contained within the convex hull of the inputs at the fault-free processes. We explore the *convex consensus* problem under crash faults with *incorrect* inputs, and present an *asynchronous* approximate convex consensus algorithm with optimal fault tolerance that reaches consensus on an *optimal* output polytope. *Convex consensus* can be used to solve other related problems. For instance, a solution for convex consensus trivially yields a solution for *vector consensus*. More importantly, convex consensus can potentially be used to solve other more interesting problems, such as *convex function optimization* [5, 4].

*This research is supported in part by National Science Foundation awards 1059540 and 1329681. 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.

1 Introduction

The distributed *consensus* problem has received significant attention over the past three decades [3]. The traditional consensus problem formulation assumes that each process has a scalar input. As a generalization of this problem, recent work [13, 20, 19] has addressed *vector* consensus (also called *multidimensional* consensus) in the presence of Byzantine faults, wherein each process has a d-dimensional vector of reals as input, and the processes reach consensus on a d-dimensional vector within the convex hull of the inputs at fault-free processes ($d \geq 1$). In the discussion below, it will be more convenient to view a d-dimensional vector as a *point* in the d-dimensional Euclidean space.

This paper defines the problem of *convex consensus*. Similar to *vector consensus*, the input at each process is a point in the d-dimensional Euclidean space. However, for convex consensus, the output at each process is a *convex polytope* contained within the convex hull of the inputs at the fault-free processes. Intuitively, the goal is to reach consensus on the "largest possible" polytope within the convex hull of the inputs at fault-free processes, allowing the processes to estimate the domain of inputs at the fault-free processes. In some cases, the output convex polytope may consist of just a single point, but in general, it may contain an infinite number of points.

Convex consensus may be used to solve other related problems. For instance, a solution for convex consensus trivially yields a solution for *vector consensus* [13, 20]. More importantly, convex consensus can potentially be used to solve other more interesting problems, such as *convex function optimization* [5, 4, 15] with the convex hull of the inputs at fault-free processes as the domain. We will discuss the application of convex consensus to function optimization in Section 4.

Fault model: With the exception of Section 3.3, rest of the paper assumes the *crash faults with incorrect inputs* [6, 3] fault model. In this model, each faulty process may crash, and may also have an *incorrect input*. A faulty process performs the algorithm faithfully, using possibly incorrect input, until it (possibly) crashes. The implication of an *incorrect input* will be clearer when we formally define convex consensus below. At most f processes may be faulty. All fault-free processes have *correct inputs*. Since this model allows incorrect inputs at faulty processes, the simulation techniques in [6, 3] can be used to transform an algorithm designed for this fault model to an algorithm for tolerating Byzantine faults. For brevity, we do not discuss this transformation. (A Byzantine convex consnesus algorithm is also presented in our technical report [17].) Section 3.3 briefly discusses how our results extend naturally to the more commonly used *crash fault* model wherein faulty processes have correct inputs (we will refer to the latter model as *crash faults with correct inputs*).[1]

System model: The system under consideration is *asynchronous*, and consists of n processes. Let the set of processes be denoted as $V = \{1, 2, \cdots, n\}$. All processes can communicate with each other. Thus, the underlying communication network is modeled as a *complete graph*. Communication channels are reliable and FIFO [7, 6]. Each message is delivered exactly once on each channel. The input at process i, denoted as x_i, is a point in the d-dimensional Euclidean space (equivalently, a d-dimensional vector of real numbers).

[1] Our results also easily extend to the case when up to f processes may crash, and up to ψ processes may have incorrect inputs, with the set of crashed processes not necessarily being identical to the processes with incorrect inputs. For brevity, we omit this generalization.

Convex consensus: The FLP impossibility of reaching *exact* consensus in asynchronous systems with crash faults [9] extends to the problem of convex consensus as well. Therefore, we consider *approximate* convex consensus in our work. An *approximate convex consensus* algorithm must satisfy the following properties:

- **Validity**: The *output* (or *decision*) at each fault-free process must be a convex polytope in the convex hull of *correct inputs*. Under the *crash fault with incorrect inputs* model, the input at any faulty process may possibly be *incorrect*.
- ϵ**-Agreement**: For a given constant $\epsilon > 0$, the *Hausdorff distance* (defined below) between the output polytopes at any two fault-free processes must be at most ϵ.
- **Termination:** Each fault-free process must terminate within a finite amount of time.

Distance metrics:

- $\mathbf{d}_E(p, q)$ denotes the Euclidean distance between points p and q. All *points* and *polytopes* in our discussion belong to a d-dimensional Euclidean space, for some $d \geq 1$, even if this is not always stated explicitly.
- For two convex polytopes h_1, h_2, the *Hausdorff distance* $\mathbf{d}_H(h_1, h_2)$ is defined as follows [11].

$$\mathbf{d}_H(h_1, h_2) \quad = \quad \max \ \left\{ \ \max_{p_1 \in h_1} \ \min_{p_2 \in h_2} \mathbf{d}_E(p_1, p_2), \qquad \max_{p_2 \in h_2} \ \min_{p_1 \in h_1} \mathbf{d}_E(p_1, p_2) \ \right\} \tag{1}$$

Optimality of approximate convex consensus: The algorithm proposed in this paper is optimal in two ways. It requires an optimal number of processes to tolerate f faults, and it decides on a convex polytope that is optimal in a "worst-case sense", as elaborated below:

- Prior work on approximate *vector* consensus mentioned earlier [13, 20] showed that $n \geq (d+2)f + 1$ is necessary to solve that problem in an asynchronous system consisting of n processes with at most f Byzantine faults. Although these prior papers dealt with Byzantine faults, it turns out that their proof of lower bound on n (i.e., lower bound of $(d+2)f+1$) is also directly applicable to approximate *vector* consensus under the *crash fault with incorrect inputs* model used in our present work. Thus, $n \geq (d+2)f + 1$ is a necessary condition for *vector* consensus under this fault model. Secondly, it is easy to show that an algorithm for approximate *convex* consensus can be transformed into an algorithm for approximate *vector* consensus. Therefore, $n \geq (d+2)f+1$ is a necessary condition for approximate *convex* consensus as well. For brevity, we omit a formal proof of the lower bound, and our subsequent discussion under the *crash faults with incorrect inputs* model assumes that

$$n \quad \geq \quad (d+2)f + 1 \tag{2}$$

  Our algorithm is correct under this condition, and thus achieves optimal fault resilience. For crash faults with *correct inputs*, a smaller n suffices, as discussed later in Section 3.3.

- In this paper, we only consider deterministic algorithms. A convex consensus algorithm A is said to be optimal if the following condition is true:

Let F denote a set of up to f faulty processes. For a **given execution** of algorithm A with F being the set of faulty processes, let $y_i(A)$ denote the output polytope at process i at the end of the given execution. For any other convex consensus algorithm B, **there exists** an execution with F being the set of faulty processes, such that $y_i(B)$ is the output at fault-free process i, and $y_j(B) \subseteq y_j(A)$ for **each** fault-free process j.

The goal here is to decide on an output polytope that includes as much of the convex hull of *all* correct inputs as possible. However, since any process may be potentially faulty (with incorrect input), the output polytope can be smaller than the convex hull of *all* correct inputs. Intuitively speaking, the optimality condition says that an optimal algorithm should decide on a convex region that is *no smaller than that decided in a worst-case execution* of algorithm B. In Section 3.2, we will show that our proposed algorithm is optimal in the above sense.

Summary of main contributions of the paper:

- The paper introduces the problem of *convex consensus*. We believe that feasibility of convex consensus can be used to infer feasibility of solving other interesting problems as well.
- We present an approximate convex consensus algorithm in *asynchronous* systems, and show that it achieves optimality in terms of its resilience, and also in terms of the convex polytope that it decides on.
- We show that the convex consensus algorithm can be used to solve a version of the *convex function optimization* problem. We also prove an impossibility result pertaining to convex function optimization with crash faults in asynchronous systems.

Related Work: For brevity, we only discuss the most relevant prior work here. Many researchers in the decentralized control area, including Bertsekas and Tsitsiklis [4] and Jadbabaei, Lin and Moss [12], have explored approximate consensus *in the absence of process faults*, using only near-neighbor communication in systems wherein the communication graph may be partially connected and time-varying. The structure of the proof of correctness of the algorithm presented in this paper, and our use of well-known matrix analysis results [21], is inspired by the above prior work. We have also used similar proof structures in our prior work on other (Byzantine) consensus algorithms [18, 20]. With regards to the proof technique, this paper's contribution is to show how the above proof structure can be extended to the case when the process state consists of convex polytopes.

Dolev et al. addressed approximate Byzantine consensus in both synchronous and asynchronous systems [7] (with scalar input). Subsequently, Coan proposed a simulation technique to transform consensus algorithms that are resilient to crash faults into algorithms tolerating Byzantine faults [6, 3]. Independently, Abraham, Amit and Dolev proposed an algorithm for approximate Byzantine consensus [1]. As noted earlier, the recent work of Mendes and Herlihy [13] and Vaidya and Garg [20] has addressed approximate *vector* consensus in the presence of Byzantine faults. This work has yielded lower bounds on the number of processes, and algorithms with optimal resilience for asynchronous [13, 20] as well as synchronous systems [20] modeled as complete graphs. Subsequent work [19] has explored the vector consensus problem in incomplete graphs.

Mendes, Tasson and Herlihy [14] study the problem of *Barycentric* agreement. *Barycentric* agreement has some similarity to convex consensus, in that the output of Barycentric agreement is not limited to a single value (or a single point). However, the correctness conditions for Barycentric agreement are different from those of our convex consensus problem.

2 Preliminaries

Some notations introduced in the paper are summarized in Appendix A. In this section, we introduce functions $\mathcal{H}$, $\mathbf{L}$, and a communication primitive used in our algorithm.

Definition 1 *For a multiset of points X, $\mathcal{H}(X)$ is the convex hull of the points in X.*

A multiset may contain the same element more than once.

Definition 2 *Function $\mathbf{L}$: Suppose that ν non-empty convex polytopes $h_1, h_2, \cdots, h_\nu$, and ν weights $c_1, c_2, \cdots, c_\nu$ are given such that $0 \le c_i \le 1$ and $\sum_{i=1}^{\nu} c_i = 1$, Linear combination of these convex polytopes, $\mathbf{L}([h_1, h_2, \cdots, h_\nu]\ ;\ [c_1, c_2, \cdots, c_\nu])$, is defined as follows:*

- *$p \in \mathbf{L}([h_1, h_2, \cdots, h_\nu]; [\ c_1, c_2, \cdots, c_\nu])$ if and only if*

$$\text{for } 1 \le i \le \nu, \text{ there exists } p_i \in h_i, \quad \text{such that } \ p = \sum_{1 \le i \le \nu} c_i p_i \tag{3}$$

Because h_i's above are all convex and non-empty, $\mathbf{L}([h_1, h_2, \cdots, h_\nu]\ ;\ [c_1, c_2, \cdots, c_\nu])$ is also a convex non-empty polytope. (The proof is straightforward.) The parameters for $\mathbf{L}$ consist of two vectors, with the elements of the first vector being polytopes, and the elements of the second vector being the corresponding weights in the linear combination. With a slight abuse of notation, we will also specify the vector of polytopes as a multiset – in such cases, we will always assign an identical weight to all the polytopes in the multiset, and hence their ordering is not important.

***Stable vector* communication primitive:** As seen later, our algorithm proceeds in asynchronous rounds. In round 0 of the algorithm, the processes use a communication primitive called *stable vector* [2, 14], to try to learn each other's inputs. *Stable vector* was originally developed in the context of Byzantine faults [2, 14]. To achieve its desirable properties (listed below), *stable vector* requires at least $3f + 1$ processes, with at most f being Byzantine faulty. Since the *crash fault with incorrect inputs* model is weaker than the Byzantine fault model, the properties of *stable vector* listed below will hold in our context, provided that $n \ge 3f + 1$. As noted earlier in Section 1, $n \ge (d + 2)f + 1$ is a necessary condition for approximate convex consensus in the presence of crash faults with incorrect inputs. Then, with $d \ge 1$, we have $n \ge 3f + 1$, and the properties of stable vector below will hold.

In round 0 of our algorithm, each process i first broadcasts a message consisting of the tuple $(x_i, i, 0)$, where x_i is process i's input. In this tuple, 0 indicates the (asynchronous) round index. Process i then waits for the *stable vector* primitive to return a set R_i containing round 0 messages. We will rely on the following properties of the *stable vector* primitive, which are implied by results proved in prior work [2, 14].

- **Liveness**: At each process i that does not crash before the end of round 0, *stable vector* returns a set R_i containing at least $n - f$ distinct tuples of the form $(x, k, 0)$.
- **Containment**: For processes i, j that do not crash before the end of round 0, let R_i, R_j be the set of messages returned to processes i, j by *stable vector* in round 0, respectively. Then, either $R_i \subseteq R_j$ or $R_j \subseteq R_i$. (Also, by the previous property, $|R_i| \ge n - f$ and $|R_j| \ge n - f$.)

A description of the implementation of the *stable vector* primitive is omitted for lack of space. Please refer to [2, 14] for more details.

3 Proposed Algorithm and its Correctness

The proposed algorithm, named *Algorithm CC*, proceeds in asynchronous rounds. The input at each process i is named x_i. The initial round of the algorithm is called round 0. Subsequent rounds are named round 1, 2, 3, etc. In each round $t \geq 0$, each process i computes a state variable h_i, which represents a convex polytope in the d-dimensional Euclidean space. We will refer to the value of h_i at the *end* of the t-th round performed by process i as $h_i[t]$, $t \geq 0$. Thus, for $t \geq 1$, $h_i[t-1]$ is the value of h_i at the *start* of the t-th round at process i. The algorithm terminates after t_{end} rounds, where t_{end} is a constant defined later in equation (13). The state $h_i[t_{end}]$ of each fault-free process i at the end of t_{end} rounds is its output (or decision) for the consensus algorithm.

X_i and $Y_i[t]$ defined on lines 4 and 13 of the algorithm are both *multisets*. A given value may occur multiple times in a multiset. Also, the intersection in line 5 is over the convex hulls of the subsets of multiset X_i of size $|X_i|-f$ (note that each of these subsets is also a multiset). Elements of X_i are points in the d-dimensional Euclidean space, whereas elements of $Y_i[t]$ are convex polytopes. In line 14, $Y_i[t]$ specifies the multiset of polytopes whose linear combination is obtained using $\mathbf{L}$; all the weights specified as parameters to $\mathbf{L}$ here are equal to $\frac{1}{|Y_i[t]|}$.

Algorithm CC: Steps performed at process i shown below.

Initialization: All sets used below are initialized to $\emptyset$.

Round 0 **at process** i**:**

1 • On entering round 0:

2 Send message $(x_i, i, 0)$ to all the processes

3 • When *stable vector* returns a set R_i:

4 Multiset $X_i := \{\, x \mid (x, k, 0) \in R_i\}$ // Note: $|X_i| = |R_i|$

5 $h_i[0] \ := \ \cap_{C \subseteq X_i,\, |C| = |X_i| - f} \ \ \mathcal{H}(C)$

6 Proceed to Round 1

Round $t \geq 1$ **at process** i**:**

7 • On entering round $t \geq 1$:

8 $\texttt{MSG}_i[t] := \texttt{MSG}_i[t] \cup (h_i[t-1], i, t)$

9 Send message $(h_i[t-1], i, t)$ to all the processes

10 • When message (h, j, t) is received from process $j \neq i$

11 $\texttt{MSG}_i[t] := \texttt{MSG}_i[t] \cup \{(h, j, t)\}$

12 • When $|\texttt{MSG}_i[t]| \geq n - f$ for the first time:

13 Multiset $Y_i[t] := \{h \mid (h, j, t) \in \texttt{MSG}_i[t]\}$ // Note: $|Y_i[t]| = |\texttt{MSG}_i[t]|$

14 $h_i[t] \ := \ \mathbf{L}(\ Y_i[t] \ ; \ [\frac{1}{|Y_i[t]|}, \cdots, \frac{1}{|Y_i[t]|}])$

15 If $t < t_{end}$, then proceed to Round $t+1$

3.1 Proof of Correctness

The use of matrix representation in our correctness proof below is inspired by the prior work on non-fault-tolerant consensus (e.g., [12, 4]). We have also used such a proof structure in our work on Byzantine consensus [18, 20]. We now introduce more notations (some of the notations are summarized in Appendix A):

- For a *given* execution of the proposed algorithm, let F denote the *actual* set of faulty processes in that execution. Processes in F may have incorrect inputs, and they may potentially crash.
- For round $r \geq 0$, let $\mathcal{F}[r]$ denote the set of faulty processes that have crashed before sending any round r messages. Note that $\mathcal{F}[r] \subseteq \mathcal{F}[r+1] \subseteq F$.

Proofs of Lemmas 1 and 2 below are presented in Appendices B, and C, respectively.

Lemma 1 *Algorithm CC ensures* progress*: (i) all the fault-free processes will eventually progress to round 1; and, (ii) if all the fault-free processes progress to the start of round t, $t \geq 1$, then all the fault-free processes will eventually progress to the start of round $t+1$.*

Lemma 2 *For each process $i \in V - \mathcal{F}[1]$, the polytope $h_i[0]$ is non-empty and convex.*

We now introduce some matrix notation and terminology to be used in our proof. Boldface upper case letters are used below 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 $\mathbf{A}$. 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 [12]. For matrix products, we adopt the "backward" product convention below, where $a \leq b$,

$$\Pi_{\tau=a}^{b}\mathbf{A}[\tau] = \mathbf{A}[b]\mathbf{A}[b-1]\cdots\mathbf{A}[a] \tag{4}$$

Let $\mathbf{v}$ be a column vector of size n whose elements are convex polytopes. The i-th element of $\mathbf{v}$ is $\mathbf{v}_i$. Let $\mathbf{A}$ be a $n \times n$ row stochastic square matrix. We define the product of $\mathbf{A}_i$ (the i-th row of $\mathbf{A}$) and $\mathbf{v}$ as follows using function $\mathbf{L}$ defined in Section 2.

$$\mathbf{A}_i\mathbf{v} \quad = \quad \mathbf{L}(\mathbf{v}^T\,;\ \mathbf{A}_i) \tag{5}$$

where ^{T} denotes the transpose operation. The above product is a polytope in the d-dimensional Euclidean space. Product of matrix $\mathbf{A}$ and $\mathbf{v}$ is then defined as follows:

$$\mathbf{A}\mathbf{v} = [\ \mathbf{A}_1\mathbf{v} \quad \mathbf{A}_2\mathbf{v} \quad \cdots \quad \mathbf{A}_n\mathbf{v}\ ]^T \tag{6}$$

Due to the transpose operation above, the product $\mathbf{A}\mathbf{v}$ is a column vector consisting of n polytopes.

Now, we describe how to represent Algorithm CC using a matrix form. Let $\mathbf{v}[t]$, $t \geq 0$, denote a column vector of length n. In the remaining discussion, we will refer to $\mathbf{v}[t]$ as the state of the system at the end of round t. In particular, $\mathbf{v}_i[t]$ for $i \in V$ is viewed as the state of process i at the end of round t. We define $\mathbf{v}[0]$ as follows as *initialization* of the state vector:

(I1) For each process $i \in V - \mathcal{F}[1]$, $\mathbf{v}_i[0] := h_i[0]$.

(I2) Pick any one fault-free process $m \in V - F \subseteq V - \mathcal{F}[1]$. For each process $k \in \mathcal{F}[1]$, $\mathbf{v}_k[0]$ is *arbitrarily* defined to be equal to $h_m[0]$. Such an arbitrary choice suffices because the state $\mathbf{v}_k[0]$ for $k \in \mathcal{F}[1]$ does not impact future state of any other process (by definition, processes in $\mathcal{F}[1]$ do not send any messages in round 1 and beyond).

We will show that the state evolution can be expressed using matrix form as in (7) below, where $\mathbf{M}[t]$ is an $n \times n$ matrix with certain desirable properties. The state $\mathbf{v}_k[t]$ of process $k \in \mathcal{F}[t]$ is not meaningful, since process k has crashed. However, (7) assigns it a value for convenience of analysis. $\mathbf{M}[t]$ is said to be the *transition matrix* for round t.

$$\mathbf{v}[t] = \mathbf{M}[t]\ \mathbf{v}[t-1], \qquad t \geq 1 \tag{7}$$

In particular, given an execution of the algorithm, we construct the transition matrix $\mathbf{M}[t]$ for round $t \geq 1$ of that execution using the two rules below (*Rule 1* and *Rule 2*). Elements of row $\mathbf{M}_i[t]$ will determine the state $\mathbf{v}_i[t]$ of process i (specifically, $\mathbf{v}_i[t] = \mathbf{M}_i[t]\mathbf{v}[t-1]$). Note that *Rule 1* applies to processes in $V - \mathcal{F}[t+1]$. Each process $i \in V - \mathcal{F}[t+1]$ survives at least until the start of round $t+1$, and sends at least one message in round $t+1$. Therefore, its state $\mathbf{v}_i[t]$ at the end of round t is of consequence. On the other hand, processes in $\mathcal{F}[t+1]$ crash sometime before sending any messages in round $t+1$ (possibly crashing in previous rounds). Thus, their states at the end of round t are not relevant to the fault-free processes anymore, and hence *Rule 2* defines the entries of the corresponding rows of $\mathbf{M}[t]$ somewhat arbitrarily.

In the matrix specification below, $\mathtt{MSG}_i[t]$ is the message set at the point where $Y_i[t]$ is defined on line 13 of the algorithm. Thus, $Y_i[t] := \{h \mid (h, j, t) \in \mathtt{MSG}_i[t]\}$, and $|\mathtt{MSG}_i[t]| = |Y_i[t]|$.

- *Rule 1*: For each process $i \in V - \mathcal{F}[t+1]$, and each $k \in V$:

  If a round t message from process k (of the form $(*, k, t)$) is in $\mathtt{MSG}_i[t]$, then

  $$\mathbf{M}_{ik}[t] := \frac{1}{|\mathtt{MSG}_i[t]|} \tag{8}$$

  Otherwise,

  $$\mathbf{M}_{ik}[t] := 0 \tag{9}$$

- *Rule 2*: For each process $j \in \mathcal{F}[t+1]$, and each $k \in V$,

  $$\mathbf{M}_{jk}[t] \ := \ \frac{1}{n} \tag{10}$$

Theorem 1 *For $t \geq 1$, define $\mathbf{v}[t] = \mathbf{M}[t]\mathbf{v}[t-1]$, with $\mathbf{M}[t]$ as specified above. Then, for $\tau \geq 0$, and for all $i \in V - \mathcal{F}[\tau+1]$, $\mathbf{v}_i[\tau]$ equals $h_i[\tau]$.*

The proof is presented in Appendix D. The above theorem states that, for $t \geq 1$, equation (7), that is, $\mathbf{v}[t] = \mathbf{M}[t]\mathbf{v}[t-1]$, correctly characterizes the state of the processes that have not crashed before the end of round t. For processes that have crashed, their states are not relevant, and could be assigned any arbitrary value for analytical purposes (this is what *Rule 2* above effectively does). Given the matrix product definition in (6), and by repeated application of the state evolution equation (7), we obtain

$$\mathbf{v}[t] \ = \ \left(\Pi_{\tau=1}^{t}\mathbf{M}[\tau]\right)\ \mathbf{v}[0], \qquad t \geq 1 \tag{11}$$

Recall that we adopt the "backward" matrix product convention presented in (4).

Definition 3 *A polytope is* valid *if it is contained in the convex hull of the inputs of fault-free processes.*

Theorem 2 *Algorithm CC satisfies the* validity*,* ϵ-agreement *and* termination *properties.*

Proof Sketch: Appendix F presents the complete proof. Repeated application of Lemma 1 ensures that the fault-free processes will progress to the end of round t, for $t \geq 1$. By repeated application of Theorem 1, $h_i[t]$ equals the i-th element of $(\Pi_{\tau=1}^{t}\mathbf{M}[\tau])\mathbf{v}[0]$, for $i \in V - \mathcal{F}[t+1]$.

Validity: By design, $\mathbf{M}[\tau]$ is a row stochastic matrix for each τ, therefore, $\Pi_{\tau=1}^{t}\mathbf{M}[\tau]$ is also row stochastic. As shown in Lemma 6 in Appendix E, $\mathbf{v}_i[0] = h_i[0]$ is valid for each fault-free process $i \in V - \mathcal{F}[1]$. Also, for each $k \in \mathcal{F}[1]$, in initialization step (I2), we defined $\mathbf{v}_k[0] = h_m[0]$, where m is a fault-free process. Hence, $\mathbf{v}_k[0]$ is valid for process $k \in \mathcal{F}[1]$. Therefore, all the elements of $\mathbf{v}[0]$ are valid. This observation, in conjunction with the previous observation that $\Pi_{\tau=1}^{t}\mathbf{M}[\tau]$ is row stochastic, and the product definition in (6), implies that each element of $\mathbf{v}[t] = (\Pi_{\tau=1}^{\tau}\mathbf{M}[\tau])\mathbf{v}[0]$ is also valid. Then, Theorem 1 implies that the state of each fault-free process is always valid, and hence its output (i.e., its state after t_{end} rounds) meets the validity condition.

ϵ-Agreement and Termination: To simplify the termination of the algorithm, we assume that the input at each process belongs to a bounded space; in particular, each coordinate of x_i is lower bounded by μ and upper bounded by U, where μ and U are known constants. Let $\mathbf{P}[t] = \Pi_{\tau=1}^{t}\,\mathbf{M}[\tau]$. Then, as shown in Lemma 5 (Appendix E), for $i, j \in V - F$ and $k \in V$,

$$\| \mathbf{P}_{ik}[t] - \mathbf{P}_{jk}[t] \| \quad \leq \quad \left(1 - \frac{1}{n}\right)^t \tag{12}$$

where $\|x\|$ denotes absolute value of a real number x. Recall from previous discussion that, due to Theorem 1, for each fault-free process i, $h_i[t]$ equals the i-th element of $\mathbf{P}[t]\mathbf{v}[0]$. This in conjunction with (12) can be used to prove that for $i, j \in V - F$, the Hausdorff distance between $h_i[t]$ and $h_j[t]$ is bounded. In particular, for $i, j \in V - F$,

$$\mathbf{d}_H(h_i[t], h_j[t]) < \left(1 - \frac{1}{n}\right)^t \sqrt{dn^2 \max(U^2, \mu^2)}$$

By defining t_{end} to be the smallest integer satisfying the inequality below, ϵ-agreement and termination conditions both follow.

$$\left(1 - \frac{1}{n}\right)^t \sqrt{dn^2 \max(U^2, \mu^2)} < \epsilon \tag{13}$$

□

3.2 Optimality of Algorithm CC

Due to the *Containment* property of *stable vector* mentioned in Section 2, the set Z defined below contains at least $n - f$ messages. Recall that set R_i is defined on line 3 of Algorithm CC.

$$Z \quad := \quad \cap_{i \in V - F}\ R_i \tag{14}$$

Define multiset $X_Z := \{x \mid (x, k, 0) \in Z\}$. Then, define a convex polytope I_Z as follows.

$$I_Z \quad := \quad \cap_{D \subset X_Z, |D| = |X_Z| - f}\ \mathcal{H}(D) \tag{15}$$

Now we establish a "lower bound" on output at the fault-free processes.

Lemma 3 *For all* $i \in V - \mathcal{F}[t+1]$ *and* $t \geq 0$, $I_Z \subseteq h_i[t]$.

Lemma 3 is proved in Appendix G. The following theorem is proved in Appendix H.

Theorem 3 Algorithm CC *is optimal under the notion of optimality in Section 1.*

Degenerate Cases: In some cases, the output polytope at fault-free processes may be a single point, making the output equivalent to that obtained from *vector consensus* [13, 20]. As a trivial example, this occurs when all the fault-free processes have identical input. It is possible to identify scenarios when the number of processes is exactly equal to the lower bound, i.e., $n = (d+2)f + 1$ processes, when the output polytope consists of just a single point. However, in general, particularly when n is larger than the lower bound, the output polytopes will contain infinite number of points. In any event, as shown in Theorem 3, our algorithm achieves optimality in all cases. Thus, any other algorithm can also produce such degenerate outputs for the same inputs.

3.3 Convex Consensus under Crash Faults with Correct Inputs

With some simple changes, our algorithm and results can be extended to achieve convex consensus under the *crash faults with correct inputs* model. Under this model, we still need to satisfy the ϵ-agreement and termination properties stated in Section 1. The validity property remains unchanged as well, however, in this model, inputs at all processes are always correct. Thus, validity implies that the ouput will be contained in the convex hull of the inputs at *all* the processes.

To obtain the algorithm for convex consensus under the *crash faults with correct inputs* model, three key changes required. First, the lower bound on the number of processes becomes $n \geq 2f+1$, which is independent of the dimension d. Second, we need a version of the *stable vector* primitive that satisfies the properties stated previously with just $2f+1$ processes (this is feasible). Finally, instead of the computation in line 5 of Algorithm CC, the computation of $h_i[0]$ needs to be modified as $h_i[0] := \mathcal{H}(X_i)$, where $X_i := \{\, x \,|\, (x, k, 0) \in R_i\}$. With these changes, the modified algorithm achieves convex consensus under the crash faults with *correct* input model, with the rest of the proof being similar to the proof for the crash faults with *incorrect* inputs model. The modified algorithm exhibits optimal resilence as well.

4 Convex Function Optimization

A motivation behind our work on *convex consensus* was to develop an algorithm that may be used to solve a broader range of problems. For instance, *vector consensus* can be achieved by first solving *convex consensus*, and then using the centroid of the output polytope of convex consensus as the output of *vector consensus*. Similarly, *convex consensus* can be used to solve a *convex function optimization* problem [5, 4, 15] under the crash faults with *incorrect inputs* model. We present an algorithm for this, and then discuss some of its properties, followed by an impossibility result of more general interest. The desired outcome of the function optimization problem is to minimize a cost function, say function c, over a domain consisting of the convex hull of the correct inputs. The proposed algorithm has two simple steps:

- Step 1: First solve convex consensus with parameter ϵ. Let h_i be the output polytope of convex consensus at process i.

- Step 2: The output of function optimization is the tuple $(y_i, c(y_i))$, where $y_i = \arg\min_{x \in h_i} c(x)$.

We assume the following property for some constant B: for any inputs x, y, $\|c(x) - c(y)\| \leq B\, \mathbf{d}_E(x, y)$ (B-Lipschitz continuity). Then, it follows that, for fault-free processes i, j, $\|c(y_i) - c(y_j)\| \leq \epsilon B$. Thus, the fault-free processes find approximately equal minimum value for the function. However, $c(y_i)$ at process i may not be minimum over the *entire* convex hull of the inputs of fault-free processes. For instance, even when all the processes are fault-free, each subset of f processes is viewed as *possibly* faulty with incorrect inputs. The natural question then is "Is it possible to find an algorithm that always find a smaller minimum value than the above algorithms?" We can extend the notion of optimality from Section 1 to function optimization in a natural way, and show that no other algorithm can obtain a lower minimum value (in the worst-case) than the above algorithm. Appendix I elaborates on this.

The above discussion implies that for any $\beta > 0$, we can achieve $\|c(y_i) - c(y_j)\| < \beta$ by choosing $\epsilon = \beta/B$ for convex consensus in Step 1. However, we are not able to similarly show that $\mathbf{d}_E(y_i, y_j)$ is small. In particular, if there are multiple points on the *boundary* of h_i that minimize the cost function, then one of the points is chose arbitrarily as $\arg\min_{x \in h_i} c(x)$, and consensus on the point is not guaranteed. It turns out it is not feasible to simultaneously reach (approximate) consensus on a point, and to also ensure that the cost function at that point is "small enough". We briefly present an impossibility result that makes a more precise statement of this infeasibility. It can be shown (Appendix I) that the following four properties cannot be satisfied simultaneously in the presence of up to f crash faults with incorrect inputs.[2] The intuition behind part (ii) of the weak optimality condition below is as follows. When $2f + 1$ processes have an identical input, say x^*, even if f of them are slow (or crash), each fault-free process must be able to learn that $f + 1$ processes have input x^*, and at least one of these $f + 1$ processes must be fault-free. Therefore, it would know that the minimum value of the cost function over the convex hull of the correct inputs is at most $c(x^*)$. Note that our algorithm above achieves weak β-optimality but not ϵ-agreement.

- **Validity**: output y_i at fault-free process i is a point in the convex hull of the correct inputs.
- **ϵ-Agreement**: for any constant $\epsilon > 0$, for any fault-free processes i, j, $\mathbf{d}_E(y_i, y_j) < \epsilon$.
- **Weak β-Optimality**: (i) for any constant $\beta > 0$, for any fault-free processes i, j, $\|c(y_i) - c(y_j)\| < \beta$, and (ii) if at least $2f + 1$ processes (faulty or fault-free) have an identical input, say x, then for any fault-free process i, $c(y_i) \leq c(x)$.
- **Termination:** each fault-free process must terminate within a finite amount of time.

The proof of impossibility for $n \geq 4f + 1$ and $d \geq 1$ is presented in Appendix I. We know that even without the weak β-optimality, we need $n \geq (d+2)f+1$. Thus, the impossibility result is complete for $d \geq 2$. Whether the impossibility extends to $3f + 1 \leq n \leq 4f$ and $d = 1$ is presently unknown.

5 Summary

We introduce the *convex consensus* problem under crash faults with incorrect inputs, and present an asynchronous approximate convex consensus algorithm with optimal fault tolerance that reaches consensus on an *optimal* output polytope. We briefly extend the results to the *crash faults with correct inputs* model, and also use the convex consensus algorithm to solve convex function optimization. An impossibility result for asynchronous function optimization is also presented.

[2] Similar impossibility result can be shown for the crash fault with *correct* inputs model too.

References

[1] I. Abraham, Y. Amit, and D. Dolev. Optimal resilience asynchronous approximate agreement. In *OPODIS*, pages 229–239, 2004.

[2] H. Attiya, A. Bar-Noy, D. Dolev, D. Peleg, and R. Reischuk. Renaming in an asynchronous environment. *Journal of the ACM*, July 1990.

[3] H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*. Wiley Series on Parallel and Distributed Computing, 2004.

[4] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Optimization and Neural Computation Series. Athena Scientific, 1997.

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1122, 2011.

[6] B. A. Coan. A compiler that increases the fault tolerance of asynchronous protocols. *IEEE Trans. Comput.*, 37(12):1541–1553, Dec. 1988.

[7] D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, and W. E. Weihl. Reaching approximate agreement in the presence of faults. *J. ACM*, 33:499–516, May 1986.

[8] M. J. Fischer, N. A. Lynch, and M. Merritt. Easy impossibility proofs for distributed consensus problems. In *Proceedings of the fourth annual ACM symposium on Principles of distributed computing*, PODC '85, pages 59–70, New York, NY, USA, 1985. ACM.

[9] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32:374–382, April 1985.

[10] J. Hajnal. Weak ergodicity in non-homogeneous markov chains. In *Proceedings of the Cambridge Philosophical Society*, volume 54, pages 233–246, 1958.

[11] D. Huttenlocher, G. Klanderman, and W. Rucklidge. Comparing images using the hausdorff distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(9):850–863, 1993.

[12] A. Jadbabaie, J. Lin, and A. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *Automatic Control, IEEE Transactions on*, 48(6):988 – 1001, june 2003.

[13] H. Mendes and M. Herlihy. Multidimensional approximate agreement in byzantine asynchronous systems. In *STOC '13*, 2013.

[14] H. Mendes, C. Tasson, and M. Herlihy. Brief announcement: The topology of asynchronous byzantine colorless tasks. In *The 27th International Symposium on Distributed Computing (DISC)*, 2013.

[15] A. Nedic and A. Ozdaglar. Cooperative distributed multi-agent optimization. *Convex Optimization in Signal Processing and Communications*, 2010.

[16] M. A. Perles and M. Sigorn. A generalization of tverberg's theorem. *CoRR*, abs/0710.4668, 2007.

[17] L. Tseng and N. H. Vaidya. Byzantine convex consensus: An optimal algorithm. *CoRR*, abs/1307.1332, 2013.

[18] L. Tseng and N. H. Vaidya. Iterative approximate byzantine consensus under a generalized fault model. In *In International Conference on Distributed Computing and Networking (ICDCN)*, January 2013.

[19] N. H. Vaidya. Iterative byzantine vector consensus in incomplete graphs. In *In International Conference on Distributed Computing and Networking (ICDCN)*, January 2014.

[20] 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 '13, pages 65–73, New York, NY, USA, 2013. ACM.

[21] J. Wolfowitz. Products of indecomposable, aperiodic, stochastic matrices. In *Proceedings of the American Mathematical Society*, volume 14, pages 733–737, 1963.

A Notations

This appendix summarizes some of the notations and terminology introduced throughout the paper.

- d = dimension of the input vector at each process.
- n = number of processes. We assume that $n \geq (d+2)f + 1$.
- f = maximum number of faulty processes.
- $V = \{1, 2, \cdots, n\}$ is the set of all processes.
- $\mathbf{d}_E(p, q)$ = Euclidean distance between points p and q.
- $\mathbf{d}_H(h_1, h_2)$ = the Hausdorff distance between convex polytopes h_1, h_2.
- $\mathcal{H}(C)$ = the convex hull of a multiset C.
- $\mathbf{L}([h_1, h_2, \cdots, h_k];\ [c_1, c_2, \cdots, c_k])$, defined in Section 2, is a linear combination of convex polytopes $h_1, h_2, ..., h_k$ with weights $c_1, c_2, \cdots, c_k$, respectively.
- $|X|$ = the size of a *multiset* or *set* X.
- $\|a\|$ = the absolute value of a real number a.
- F denotes the *actual* set of faulty processes in an execution of the algorithm.
- $\mathcal{F}[t]$ $(t \geq 0)$, defined in Section 3, denotes the set of (faulty) processes that do not send any messages in round t. Thus, each process in $\mathcal{F}[t]$ must have crashed before sending any message in round t (it may have possibly crashed in an earlier round). Note that $\mathcal{F}[r] \subseteq \mathcal{F}[r+1] \subseteq F$ for $r \geq 1$.
- We use boldface upper case 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 $\mathbf{A}$.

B Proof of Lemma 1

Lemma 1: *Algorithm CC ensures* progress*: (i) all the fault-free processes will eventually progress to round 1; and, (ii) if all the fault-free processes progress to the start of round t, $t \geq 1$, then all the fault-free processes will eventually progress to the start of round $t+1$.*

Proof:

Part (i): By assumption, all fault-free processes begin the round 0 eventually, and perform a broadcast of their input (line 1). There at least $3f + 1$ processes as argued in Section 2, and at most f may crash. The Liveness property of *stable vector* ensures that it will eventually return (on line 3). Therefore, each process that does not crash in round 0 will eventually proceed to round 1 (line 6).

Part (ii): The proof is by induction. Suppose that the fault-free processes begin round $t \geq 1$. (We already proved that the fault-free processes begin round 1.) Thus, each fault-free process i will perform a broadcast of $(h_i[t-1], i, t)$ on line 9. By the assumption of reliable channels, process i will eventually receive message $(h_j[t-1], j, t)$ from each fault-free process j. Thus, it will receive messages from at least $n - f - 1$ *other* processes, and include these received messages in $\texttt{MSG}_i[t]$ (line 10-11). Also, it includes (on line 8) its own message into $\texttt{MSG}_i[t]$. Thus, $\texttt{MSG}_i[t]$ is sure to reach size $n - f$ eventually, and process i will be able to progress to round $t + 1$ (line 12-15). □

C Proof of Lemma 2

The proof of Lemma 2 uses the following theorem by Tverberg [16]:

Theorem 4 *(Tverberg's Theorem [16]) For any integer $f \geq 0$, for every multiset T containing at least $(d+1)f+1$ points in a d-dimensional space, there exists a partition $T_1, .., T_{f+1}$ of T into $f+1$ non-empty multisets such that $\cap_{l=1}^{f+1} \mathcal{H}(T_l) \neq \emptyset$.*

Now, we prove Lemma 2.

Lemma 2: *For each process $i \in V - \mathcal{F}[1]$, the polytope $h_i[0]$ is non-empty and convex.*

Proof:

Consider any $i \in V - \mathcal{F}[1]$. Consider the computation of polytope $h_i[0]$ on line 5 of the algorithm as

$$h_i[0] \; := \; \cap_{C \subseteq X_i, \, |C|=|X_i|-f} \quad \mathcal{H}(C), \tag{16}$$

where $X_i := \{ x \,|\, (x, k, 0) \in R_i \}$ (lines 4-5). Convexity of $h_i[0]$ follows directly from (16), because $h_i[0]$ is an intersection of convex hulls.

Recall that, due to the lower bound on n discussed in Section 1, we assume that $n \geq (d+2)f+1$. Thus, $|X_i| \geq n - f \geq (d+1)f + 1$. By Theorem 4 above, there exists a partition $T_1, T_2, \cdots, T_{f+1}$ of X_i into multisets (T_j's) such that $\cap_{j=1}^{f+1} \mathcal{H}(T_j) \neq \emptyset$. Let us define

$$J = \cap_{j=1}^{f+1} \mathcal{H}(T_j) \tag{17}$$

Thus, by Tverberg's theorem above, J is non-empty. Now, each multiset C used in (16) to compute $h_i[0]$ excludes only f elements of X_i, whereas there are $f + 1$ multisets in the partition $T_1, T_2, \cdots, T_{f+1}$ of multiset X_i. Therefore, each multiset C will fully contain at least one multiset from the partition. It follows that $\mathcal{H}(C)$ will contain J defined above. Since this property holds true for each multiset C used to compute $h_i[0]$, J is contained in the convex polytope $h_i[0]$ computed as per (16). Since J is non-empty, $h_i[0]$ is non-empty.

□

D Proof of Theorem 1

Theorem 1: *For $t \geq 1$, define $\mathbf{v}[t] = \mathbf{M}[t]\mathbf{v}[t-1]$, with $\mathbf{M}[t]$ as specified above. Then, for $\tau \geq 0$, and for all $i \in V - \mathcal{F}[\tau+1]$, $\mathbf{v}_i[\tau]$ equals $h_i[\tau]$.*

Proof: The proof of the above theorem is by induction on τ. Recall that we defined $\mathbf{v}_i[0]$ to be equal to $h_i[0]$ for all $i \in V - \mathcal{F}[1]$ in the initialization step (I1) in Section 3. Thus, the theorem trivially holds for $\tau = 0$.

Now, for some $\tau \geq 0$, and for all $i \in V - \mathcal{F}[\tau+1]$, suppose that $\mathbf{v}_i[\tau] = h_i[\tau]$. Recall that processes in $V - \mathcal{F}[\tau+2]$ surely survive at least till the end of round $\tau+1$ (by definition of $\mathcal{F}[\tau+2]$). Therefore, in round $\tau+1 \geq 1$, each process in $i \in V - \mathcal{F}[\tau+2]$ computes its new state $h_i[\tau+1]$ at line 14 of Algorithm CC, using function $\mathbf{L}(\ Y_i[\tau+1]\ ;\ [\frac{1}{|Y_i[\tau+1]|}, \cdots, \frac{1}{|Y_i[\tau+1]|}])$, where $Y_i[\tau+1] := \{h \mid (h, j, \tau+1) \in \texttt{MSG}_i[\tau+1]\}$. Also, if $(h, j, \tau+1) \in \texttt{MSG}_i[\tau+1]$, then process j must have sent round $\tau+1$ message $(h_j[\tau], j, \tau+1)$ to process i – in other words, h above (in $(h, j, \tau+1) \in \texttt{MSG}_i[\tau+1]$) must be equal to $h_j[\tau]$. Also, since j did send a round $\tau+1$ message, $j \in V - \mathcal{F}[\tau+1]$. Thus, by induction hypothesis, $\mathbf{v}_j[\tau] = h_j[\tau]$.

Now observe that, by definition of $Y_i[\tau+1]$ at line 13 of the algorithm, $|Y_i[\tau+1]| = |\texttt{MSG}_i[\tau+1]|$. Thus, the definition of the matrix elements in (8) and (9) ensures that $\mathbf{M}_i[\tau+1]\mathbf{v}[\tau]$ equals $\mathbf{L}(\ Y_i[\tau+1]\ ;\ [\frac{1}{|Y_i[\tau+1]|}, \cdots, \frac{1}{|Y_i[\tau+1]|}])$, i.e., $h_i[\tau+1]$. Thus, $\mathbf{v}_i[\tau+1]$ defined as $\mathbf{M}_i[\tau+1]\mathbf{v}[\tau]$ also equals $h_i[\tau+1]$. This holds for all $i \in V - \mathcal{F}[\tau+2]$, completing the induction. □

E Useful Lemmas

In this section, we prove four lemmas used later in Appendix F.

The procedure for constructing $\mathbf{M}[t]$ that the lemma below refers to is presented in Section 3.1.

Lemma 4 *For $t \geq 1$, transition matrix $\mathbf{M}[t]$ constructed using the above procedure satisfies the following conditions:*

- *$\mathbf{M}[t]$ is a row stochastic matrix.*
- *For $i, j \in V - \mathcal{F}[t+1]$, there exists a fault-free process $g(i,j)$ such that $\mathbf{M}_{ig(i,j)}[t] \geq \frac{1}{n}$ and $\mathbf{M}_{jg(i,j)}[t] \geq \frac{1}{n}$*

Proof:

- Observe that, by construction, for each $i \in V$, the row vector $\mathbf{M}_i[t]$ contains only non-negative elements, which add up to 1. Thus, each row $\mathbf{M}_i[t]$ is a stochastic vector, and hence the matrix $\mathbf{M}[t]$ is row stochastic.
- To prove the second claim in the lemma, consider any pair of processes $i, j \in V - \mathcal{F}[t+1]$. Recall that the set $\texttt{MSG}_i[t]$ used in the construction of $\mathbf{M}[t]$ is such that $|\texttt{MSG}_i[t]| = |Y_i[t]|$ (i.e., $\texttt{MSG}_i[t]$ is the message set at the point where $Y_i[t]$ is created). Thus, $|\texttt{MSG}_i[t]| \geq n - f$ and $|\texttt{MSG}_j[t]| \geq n - f$, and there must be at least $n - 2f$ messages in $\texttt{MSG}_i[t] \cap \texttt{MSG}_j[t]$. By assumption, $n \geq (d+2)f + 1$. Hence, $n - 2f \geq df + 1 \geq f + 1$, since $d \geq 1$. Therefore, there exists a fault-free process $g(i,j)$ such that $(h_{g(i,j)}[t-1], g(i,j), t) \in \texttt{MSG}_i[t] \cap \texttt{MSG}_j[t]$. By (8) in the procedure to construct $\mathbf{M}[t]$, $\mathbf{M}_{ig(i,j)}[t] = \frac{1}{|\texttt{MSG}_i[t]|} \geq \frac{1}{n}$ and $\mathbf{M}_{jg(i,j)}[t] = \frac{1}{|\texttt{MSG}_j[t]|} \geq \frac{1}{n}$.

□

To facilitate the proof of next lemma below, we first introduce some terminology and results related to matrices.

For a row stochastic matrix $\mathbf{A}$, coefficients of ergodicity $\delta(\mathbf{A})$ and $\lambda(\mathbf{A})$ are defined as follows [21]:

$$\begin{aligned}\delta(\mathbf{A}) &= \max_j \max_{i_1,i_2} \|\mathbf{A}_{i_1\,j} - \mathbf{A}_{i_2\,j}\| \\ \lambda(\mathbf{A}) &= 1 - \min_{i_1,i_2} \sum_j \min(\mathbf{A}_{i_1\,j}\ , \mathbf{A}_{i_2\,j})\end{aligned}$$

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

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

Claim 1 is proved in [10].

Claim 2 *If there exists a constant γ, where $0 < \gamma \le 1$, such that, for any pair of rows i, j of matrix $\mathbf{A}$, there exists a column g (that may depend on i, j) such that, $\min(\mathbf{A}_{ig}, \mathbf{A}_{jg}) \ge \gamma$, then $\lambda(\mathbf{A}) \le 1 - \gamma < 1$.*

Claim 2 follows directly from the definition of $\lambda(\cdot)$.

Lemma 5 *For $t \ge 1$, let $\mathbf{P}[t] = \Pi_{\tau=1}^{t}\mathbf{M}[\tau]$. Then,*

- *$\mathbf{P}[t]$ is a row stochastic matrix.*
- *For $i, j \in V - F$, and $k \in V$,*

$$\|\,\mathbf{P}_{ik}[t] - \mathbf{P}_{jk}[t]\,\| \le \left(1 - \frac{1}{n}\right)^t \tag{18}$$

  where $\|a\|$ denotes absolute value of real number a.

Proof: By the first claim of Lemma 4, $\mathbf{M}[\tau]$ for $1 \le \tau \le t$ is row stochastic. Thus, $\mathbf{P}[t]$ is a product of row stochastic matrices, and hence, it is itself also row stochastic.

Now, observe that by the second claim in Lemma 4 and Claim 2, $\lambda(\mathbf{M}[t]) \le 1 - \frac{1}{n} < 1$. Then by Claim 1 above,

$$\delta(\mathbf{P}[t]) = \delta(\Pi_{\tau=1}^{t}\mathbf{M}[\tau]) \ \le\ \Pi_{\tau=1}^{t}\lambda(\mathbf{M}[\tau]) \ \le\ \left(1 - \frac{1}{n}\right)^t \tag{19}$$

Consider any two fault-free processes $i, j \in V - F$. By (19), $\delta(\mathbf{P}[t]) \le \left(1 - \frac{1}{n}\right)^t$. Therefore, by the definition of $\delta(\cdot)$, for $1 \le k \le n$, we have

$$\|\mathbf{P}_{ik}[t] - \mathbf{P}_{jk}[t]\| \le \left(1 - \frac{1}{n}\right)^t \tag{20}$$

□

We now prove two lemmas related to validity of convex hulls computed in Algorithm CC. Recall that a valid convex hull is defined in Definition 3.

Lemma 6 *$h_i[0]$ for each process $i \in V - \mathcal{F}[1]$ is valid.*

Proof: Recall that $h_i[0]$ is obtained on line 5 of Algorithm CC as

$$h_i[0] \ := \ \cap_{C \subseteq X_i,\, |C|=|X_i|-f} \ \ \mathcal{H}(C),$$

where $X_i = \{\, x \,|\, (x,k,0) \in R_i\}$. Under the *crash faults with incorrect inputs* model, except for up to f values in X_i (which may correspond to inputs at faulty processes), all the other values in X_i must correspond to inputs at fault-free processes (and hence they are correct). Therefore, at least one set C used in the computation of $h_i[0]$ must contain only the inputs at fault-free processes. Therefore, $h_i[0]$ is in the convex hull of the inputs at fault-free processes. That is, $h_i[0]$ is valid. □

Lemma 7 *Suppose non-empty convex polytopes $h_1, h_2, \cdots, h_\nu$ are all valid. Consider ν constants $c_1, c_2, \cdots, c_\nu$ such that $0 \le c_i \le 1$ and $\sum_{i=1}^{\nu} c_i = 1$. Then the linear combination of these convex polytopes, $\boldsymbol{L}([h_1, h_2, \cdots, h_\nu]\,;\,[c_1, c_2, \cdots, c_\nu])$, is* **convex, non-empty, and valid***.*

Proof: Polytopes $h_1, \cdots, h_\nu$ are given as non-empty, convex, and valid. Let

$$L := \mathbf{L}([h_1, h_2, \cdots, h_\nu]\,;\ [c_1, c_2, \cdots, c_\nu]) \tag{21}$$

We will show that L is convex, non-empty, and valid.

L ***is convex:*** Given any two points x, y in L, by Definition 2, we have

$$x = \sum_{1 \le i \le \nu} c_i p_{(i,x)} \ \ \text{for some } p_{(i,x)} \in h_i, 1 \le i \le \nu \tag{22}$$

and

$$y = \sum_{1 \le i \le \nu} c_i p_{(i,y)} \ \ \text{for some } p_{(i,y)} \in h_i, \ \ 1 \le i \le \nu \tag{23}$$

Now, we show that any convex combination of x and y is also in L defined in (21). Consider a point z such that

$$z = \theta x + (1-\theta) y \quad \text{where } 0 \le \theta \le 1 \tag{24}$$

Substituting (22) and (23) into (24), we have

$$\begin{aligned} z &= \theta \sum_{1 \le i \le \nu} c_i p_{(i,x)} + (1-\theta) \sum_{1 \le i \le \nu} c_i p_{(i,y)} \\ &= \sum_{1 \le i \le \nu} c_i \left(\theta p_{(i,x)} + (1-\theta) p_{(i,y)}\right) \end{aligned} \tag{25}$$

Define $p_{(i,z)} = \theta p_{(i,x)} + (1-\theta) p_{(i,y)}$ for $1 \leq i \leq \nu$. Since h_i is convex, and $p_{(i,z)}$ is a convex combination of $p_{(i,x)}$ and $p_{(i,y)}$, $p_{(i,z)}$ is also in h_i. Substituting the definition of $p_{(i,z)}$ in (25), we have

$$z = \sum_{1 \leq i \leq \nu} c_i \, p_{(i,z)} \quad \text{where } p_{(i,z)} \in h_i, \quad 1 \leq i \leq \nu$$

Hence, by Definition 2, z is also in L. Therefore, L is convex.

L ***is non-empty:*** The proof that L is non-empty is trivial. Since each of the h_i's is non-empty, there exists at least one point $z_i \in h_i$ for $1 \leq i \leq \nu$. Then $\sum_{1 \leq i \leq \nu} c_i z_i$ is in L, and hence L is non-empty.

L ***is valid:*** The proof that L is valid is also straightforward. Since each of the h_i's is valid, each point in each h_i is a convex combination of the inputs at the fault-free processes. Since each point in L is a convex combination of points in h_i's, it then follows that each point in L is in the convex hull of the inputs at fault-free processes. □

F Proof of Theorem 2

Theorem 2: *Algorithm CC satisfies the* validity*,* ϵ-agreement *and* termination *properties.*

Proof: We prove that Algorithm CC satisfies the *validity*, *ϵ-agreement* and *termination* properties after a large enough number of asynchronous rounds.

Repeated applications of Lemma 1 ensures that the fault-free processes will progress from round 0 through round r, for any $r \geq 0$, allowing us to use (11). Consider round $t \geq 1$. Let

$$\mathbf{P}[t] \;=\; \Pi_{\tau=1}^{t} \mathbf{M}[\tau] \tag{26}$$

Validity: We prove validity using the series of observations below:

- *Observation 1*: By Lemma 2 (in Appendix C) and Lemma 6 (in Appendix E), $h_i[0]$ for each $i \in V - \mathcal{F}[1]$ is non-empty and valid. Also, each such $h_i[0]$ is convex by construction (line 5 of Algorithm CC).
- *Observation 2*: As per the initialization step (I1) (in Section 3.1), for each $i \in V - \mathcal{F}[1]$, $\mathbf{v}_i[0] := h_i[0]$; thus, by Observation 1 above, for each such process i, $\mathbf{v}_i[0]$ is convex, valid and non-empty. Also, in initialization step (I2) (in Section 3.1), for each process $k \in \mathcal{F}[1]$, we set $\mathbf{v}_k[0] := h_m[0]$, where m is a fault-free process; thus, by Observation 1, for each such process k, $\mathbf{v}_k[0]$ is convex, valid and non-empty. Therefore, each element of $\mathbf{v}[0]$ is a non-empty, convex and valid polytope.
- *Observation 3*: By Lemma 5 in Appendix E, $\mathbf{P}[t]$ is a *row stochastic* matrix. Thus, elements of each row of $\mathbf{P}[t]$ are non-negative and add up to 1. Therefore, by Observation 2 above, and Lemma 7 in Appendix E, $\mathbf{P}_i[t]\mathbf{v}[0]$ for each $i \in V - F$ is valid, convex and non-empty.

Also, by Theorem 1, and equation (11), $h_i[t] = \mathbf{P}[t]\mathbf{v}[0]$ for $i \in V - F$. Thus, $h_i[t]$ is valid, convex and non-empty for $t \geq 1$.

Therefore, *Algorithm CC* satisfies the validity property.

ϵ-Agreement and Termination: Recall that by Lemma 5 in Appendix E, for any two fault-free processes $i, j \in V - F$, and for $1 \leq k \leq n$, we have

$$\|\mathbf{P}_{ik}[t] - \mathbf{P}_{jk}[t]\| \leq \left(1 - \frac{1}{n}\right)^t$$

Processes in $\mathcal{F}[1]$ do not send any messages to any other process in round 1 and beyond. Thus, by the construction of $\mathbf{M}[t]$, for each $a \in V - \mathcal{F}[1]$ and $b \in \mathcal{F}[1]$, $\mathbf{M}_{ab}[t] = 0$ for all $t \geq 1$; it then follows that $\mathbf{P}_{ab}[t] = 0$ as well.[3]

Consider fault-free processes $i, j \in V - F$. (In the following discussion, we will denote a point in the d-dimensional Euclidean space by a list of its d coordinates.) The previous paragraph implies that, for any point q_i in $h_i[t] = \mathbf{v}_i[t] = \mathbf{P}_i[t]\mathbf{v}[0]$, there must exist, for all $k \in V - \mathcal{F}[1]$, $p_k \in h_k[0]$, such that

$$q_i = \sum_{k \in V - \mathcal{F}[1]} \mathbf{P}_{ik}[t] p_k = \left(\sum_{k \in V - \mathcal{F}[1]} \mathbf{P}_{ik}[t] p_k(1), \quad \sum_{k \in V - \mathcal{F}[1]} \mathbf{P}_{ik}[t] p_k(2), \cdots, \sum_{k \in V - \mathcal{F}[1]} \mathbf{P}_{ik}[t] p_k(d) \right) \tag{27}$$

where $p_k(l)$ denotes the value of p_k's l-th coordinate. The list on the right-hand-side of the above equation represents the d coordinates of point p_i.

Using points p_k in the above equation, now choose point q_j in $h_j[t]$ defined as follows.

$$q_j = \sum_{k \in V - \mathcal{F}[1]} \mathbf{P}_{jk}[t] p_k = \left(\sum_{k \in V - \mathcal{F}[1]} \mathbf{P}_{jk}[t] p_k(1), \quad \sum_{k \in V - \mathcal{F}[1]} \mathbf{P}_{jk}[t] p_k(2), \cdots, \sum_{k \in V - \mathcal{F}[1]} \mathbf{P}_{jk}[t] p_k(d) \right) \tag{28}$$

Recall that the Euclidean distance between q_i and q_j is $\mathbf{d}_E(q_i, q_j)$. From Lemma 5 (in Appendix E), (27) and (28), we have the following:

[3] Claim 3 in Appendix G below provides a more detailed proof of this statement.

$$
\begin{aligned}
\mathbf{d}_E(q_i, q_j) &= \sqrt{\sum_{l=1}^{d} (q_i(l) - q_j(l))^2} \\
&= \sqrt{\sum_{l=1}^{d} \left(\sum_{k \in V - \mathcal{F}[1]} \mathbf{P}_{ik}[t] p_k(l) - \sum_{k \in V - \mathcal{F}[1]} \mathbf{P}_{jk} p_k(l) \right)^2} \qquad \text{by (27) and (28)} \\
&= \sqrt{\sum_{l=1}^{d} \left(\sum_{k \in V - \mathcal{F}[1]} (\mathbf{P}_{ik}[t] - \mathbf{P}_{jk}[t]) p_k(l) \right)^2} \\
&\leq \sqrt{\sum_{l=1}^{d} \left[\left(1 - \frac{1}{n}\right)^{2t} \left(\sum_{k \in V - \mathcal{F}[1]} \|p_k(l)\| \right)^2 \right]} \qquad \text{by Lemma 5} \\
&= \left(1 - \frac{1}{n}\right)^t \sqrt{\sum_{l=1}^{d} \left(\sum_{k \in V - \mathcal{F}[1]} \|p_k(l)\| \right)^2}
\end{aligned}
$$

Define

$$
\Omega = \max_{p_k \in h_k[0], k \in V - \mathcal{F}[1]} \sqrt{\sum_{l=1}^{d} (\sum_{k \in V - \mathcal{F}[1]} \|p_k(l)\|)^2}
$$

Then, we have

$$
\mathbf{d}_E(q_i, q_j) \leq \left(1 - \frac{1}{n}\right)^t \sqrt{\sum_{l=1}^{d} \left(\sum_{k \in V - \mathcal{F}[1]} \|p_k(l)\| \right)^2} \;\leq\; (1 - \frac{1}{n})^t \, \Omega \tag{29}
$$

Because the $h_k[0]$'s in the definition of Ω are all valid (by Lemma 6 in Appendix E), Ω can itself be upper bounded by a function of the input vectors at the fault-free processes. In particular, under the assumption that each element of fault-free processes' input vectors is upper bounded by U and lower bounded by μ, Ω is upper bounded by $\sqrt{dn^2 \max(U^2, \mu^2)}$. Observe that the upper bound on the right-hand-side of (29) monotonically decreases with t, because $1 - \frac{1}{n} < 1$. Define t_{end} as the smallest positive integer t for which

$$
\left(1 - \frac{1}{n}\right)^t \sqrt{dn^2 \max(U^2, \mu^2)} < \epsilon \tag{30}
$$

Recall that the algorithm terminates after t_{end} rounds. Since t_{end} is finite, the algorithms satisfies the *termination* condition.

(29) and (30) together imply that, for fault-free processes i, j and for each point $q_i \in h_i[t_{end}]$, there exists a point $q_j[t] \in h_j[t_{end}]$, such that $\mathbf{d}_E(q_i, q_j) < \epsilon$ (and, similarly, vice-versa). Thus, by Definition of Hausdorff distance, $\mathbf{d}_H(h_i[t_{end}], h_j[t_{end}]) < \epsilon$. Since this holds true for any pair of fault-free processes i, j, the ϵ-agreement property is satisfied at termination. □

G Proof of Lemma 3

We first prove a claim that will be used in the proof of Lemma 3.

Claim 3 *For $t \geq 1$, let $\mathbf{P}[t] = \Pi_{\tau=1}^{t}\mathbf{M}[\tau]$. Then, for all processes $j \in V - \mathcal{F}[t+1]$, and $k \in \mathcal{F}[1]$, $\mathbf{P}_{jk}[t] = 0$.*

Proof: The claim is intuitively straightforward. For completeness, we present a formal proof here. The proof is by induction on t.

Induction Basis: Consider the case when $t = 1$, $j \in V - \mathcal{F}[2]$, and $k \in \mathcal{F}[1]$. Then by definition of $\mathcal{F}[1]$, $(*, k, 0) \notin \texttt{MSG}_j[1]$. Then, due to (9), $\mathbf{M}_{jk}[1] = 0$, and hence $\mathbf{P}_{jk}[1] = \mathbf{M}_{jk}[1] = 0$.

Induction: Consider $t \geq 2$. Assume that the claim holds true through round $t-1$. Then, $\mathbf{P}_{jk}[t-1] = 0$ for all $j \in V - \mathcal{F}[t]$ and $k \in \mathcal{F}[1]$. Recall that $\mathbf{P}[t-1] = \Pi_{\tau=1}^{t-1}\mathbf{M}[\tau]$.

Now, we will prove that the claim holds true for round t. Consider $j \in V - \mathcal{F}[t+1]$ and $k \in \mathcal{F}[1]$. Note that $\mathbf{P}[t] = \Pi_{\tau=1}^{t}\mathbf{M}[\tau] = \mathbf{M}[t]\Pi_{\tau=1}^{t-1}\mathbf{M}[\tau] = \mathbf{M}[t]\mathbf{P}[t-1]$. Thus, $\mathbf{P}_{jk}[t]$ can be non-zero only if there exists a $q \in V$ such that $\mathbf{M}_{jq}[t]$ and $\mathbf{P}_{qk}[t-1]$ are both non-zero.

For any $q \in \mathcal{F}[t-1]$, $(*, q, t-1) \notin \texttt{MSG}_j[t]$. Then, due to (9), $\mathbf{M}_{jq}[t] = 0$ for all $q \in \mathcal{F}[t-1]$, and hence all $q \in \mathcal{F}[1]$ (note that $\mathcal{F}[r-1] \subseteq \mathcal{F}[r]$ for $r \geq 2$). Additionally, by the induction hypothesis, for all $q \in V - \mathcal{F}[t]$ and $k \in \mathcal{F}[1]$, $\mathbf{P}_{qk}[t-1] = 0$. Thus, these two observations together imply that there does not exist any $q \in V$ such that $\mathbf{M}_{jq}[t]$ and $\mathbf{P}_{qk}[t-1]$ are both non-zero. Hence, $\mathbf{P}_{jk}[t] = 0$. □

Now, we are ready to prove Lemma 3.

Lemma 3: *For all $i \in V - \mathcal{F}[t+1]$ and $t \geq 0$, $I_Z \subseteq h_i[t]$.*

Proof: Recall that Z and I_Z are defined in (14) and (15), respectively. We first prove that for all $j \in V - \mathcal{F}[1]$, $I_Z \subseteq h_j[0]$. We make the following observations for each process $i \in V - \mathcal{F}[1]$:

- *Observation 1*: By the definition of multiset X_i at line 4 of round 0 at process i, and the definition of X_Z in Section 3.2, we have $X_Z \subseteq X_i$.

- *Observation 2*: Let A and B be sets of points in the d-dimensional space, where $|A| \geq n - f$, $|B| \geq n - f$ and $A \subseteq B$. Define $h_A := \cap_{C_A \subseteq A, |C_A| = |A| - f} \ \mathcal{H}(C_A)$ and $h_B := \cap_{C_B \subseteq B, |C_B| = |B| - f} \ \mathcal{H}(C_B)$. Then $h_A \subseteq h_B$. This observation follows directly from the fact that every multiset C_A in the computation of h_A is contained in some multiset C_B used in the computation of h_B, and the property of $\mathcal{H}$.

Now, consider the computation of $h_i[0]$ at line 5. By Observations 1 and 2, and the definitions of $h_i[0]$ and I_Z, we have that $I_Z \subseteq h_i[0] = \mathbf{v}_i[0]$, where $i \in V - \mathcal{F}[1]$. Also, by initialization step (I2) (in Section 3.1), for $k \in \mathcal{F}[1]$, $\mathbf{v}_k[0] = h_m[0]$, for some fault-free process m. Thus, all the elements of $\mathbf{v}[0]$ contain I_Z. Then, due to row stochasticity of $\Pi_{\tau=1}^{t}\mathbf{M}[\tau]$, it follows that each element of $\mathbf{v}[t] = \left(\Pi_{\tau=1}^{t}\mathbf{M}[\tau]\right) \mathbf{v}[0]$ also contain I_Z. Recalling that $h_i[t] = \mathbf{v}_i[t]$ for each fault-free process, proves the claim of the lemma.

□

H Proof of Theorem 3

Theorem 3: Algorithm CC *is optimal under the notion of optimality in Section 1.*

Proof: Consider multiset X_Z defined in Section 3.2. Recall that $|X_Z| = |Z|$, and that Z contains at least $n - f$ tuples. Thus, X_Z contains at least $n - f$ points, and of these at least $n - 2f$ points must be the inputs at fault-free processes. Let V_Z denote the set of fault-free processes whose inputs appear in X_Z. Let $S = V - F - V_Z$. Since $|X_Z| \geq n - f$, $|S| \leq f$.

Now consider the following execution of any algorithm ALGO that correctly solves approximate convex consensus. Suppose that the faulty processes in F do not crash, but have an incorrect input. Consider the case when processes in S are so slow that the other fault-free processes must terminate before receiving any messages from the processes in S. The fault-free processes in V_Z cannot determine whether the processes in S are just slow, or they have crashed.

Processes in V_Z must be able to terminate without receiving any messages from the processes in S. Thus, their output must be in the convex hull of inputs at the fault-free processes whose inputs are included in X_Z. However, any f of the processes whose inputs are in X_Z may potentially be faulty and have incorrect inputs. Therefore, the output obtained by ALGO must be contained in I_Z as defined in Section 3.2. On the other hand, by Lemma 3 in Appendix G, the output obtained using Algorithm CC contains I_Z. This proves the theorem. □

I Convex Function Optimization

I.1 Notion of Optimality

We can extend the notion of optimality (of *convex consensus* algorithms) in Section 1 to *convex function optimization* as follows. An algorithm A for convex function optimization is said to be optimal if the following condition is true.

> Let F denote a set of up to f faulty processes. For a **given execution** of algorithm A with F being the set of faulty processes, let $y_i(A)$ denote the output at process i at the end of the given execution. For any other algorithm B, **there exists** an execution with F being the set of faulty processes, such that $y_i(B)$ is the output at fault-free process i, and $c(y_j(A)) \leq c(y_j(B))$ for **each** fault-free process j.

The intuition behind the above formulation is as follows. A goal of function optimization here is to allow the processes to "learn" the smallest value of the cost function over the convex hull of the inputs at the fault-free processes. The above condition implies that an optimal algorithm will learn a function value that is no larger than that learned in a worst-case execution of any other algorithm.

The 2-step convex function optimization algorithm, with the first step being convex consensus, as described in Section 4, is optimal in the above sense. This is a direct consequence of Theorem 3.

I.2 Impossibility Result

The four properties for convex function optimization problem introduced in Section 4 are:

- **Validity**: output y_i at fault-free process i is a point in the convex hull of the correct inputs.

- **ϵ-Agreement**: for a given constant $\epsilon > 0$, for any fault-free processes i, j, $\mathbf{d}_E(y_i, y_j) < \epsilon$.
- **Weak β-Optimality**: (i) for any constant $\beta > 0$, for any fault-free processes i, j, $\|c(y_i) - c(y_j)\| < \beta$, and (ii) if at least $2f + 1$ processes (faulty or fault-free) have an identical input, say x, then for any fault-free process i, $c(y_i) \leq c(x)$.
- **Termination:** each fault-free process must terminate within a finite amount of time.

The theorem below proves the impossibility of satisfying the above properties for $n \geq 4f + 1$ and $d \geq 1$. From our prior discussion, we know that we need $n \geq (d + 2)f + 1$ even without the weak β-optimality requirement. Thus, for $d \geq 2$, the theorem implies that for $d \geq 2$ and any n, the above properties cannot be satisfied. For the specific case of $d = 1$, we do not presently know whether the above properties can be satisfied when $3f + 1 \leq n \leq 4f$.

Theorem 5 *All the four properties above cannot be satisfied simultaneously in an asynchronous system in the presence of crash faults with incorrect inputs for $n \geq 4f + 1$ and $d \geq 1$.*

Proof: We will prove the result for $d = 1$. It should be obvious that impossibility with $d = 1$ implies impossibility for larger d (since we can always choose inputs that have 0 coordinates in all dimensions except one).

The proof is by contradiction. Suppose that there exists an algorithm, say Algorithm $\mathcal{A}$, that achieves the above four properties for $n \geq 4f + 1$ and $d = 1$.

Let the cost function be given by $c(x) = 4 - (2x - 1)^2$ for $x \in [0, 1]$ and $c(x) = 3$ for $x \notin [0, 1]$. For future reference note that within the interval $[0, 1]$, function $c(x)$ has the smallest value at $x = 0, 1$ both.

Now suppose that all the inputs (correct and incorrect) are restricted to be binary, and must be 0 or 1. (We will prove impossibility under this restriction on the inputs at faulty and fault-free processes both, which suffices to prove that the four properties cannot *always* be satisfied.) Suppose that the output of Algorithm $\mathcal{A}$ at fault-free process i is y_i. Due to the validity property, and because the inputs are restricted to be 0 or 1, we know that $y_i \in [0, 1]$.

Since $\lceil \frac{n}{2} \rceil \geq \lceil \frac{4f+1}{2} \rceil = 2f + 1$, at least $2f + 1$ processes will have either input 0, or input 1. Without loss of generality, suppose that at least $2f + 1$ processes have input 0.

Consider a fault-free process i. By weak β-Optimality, $c(y_i) \leq c(0)$, that is, $c(y_i) \leq 3$. However, the minimum value of the cost function is 3 over all possible inputs. Thus, $c(y_i) = 3$. Similarly, for any other fault-free process j as well, $c(y_j)$ must equal 3. Now, due to validity, $y_j \in [0, 1]$, and the cost function is 3 in interval $[0, 1]$ only at $x = 0, 1$. Therefore, we must have y_i equal to 0 or 1, and y_j also equal to 0 or 1. However, because algorithm $\mathcal{A}$ satisfies the ϵ-agreement condition, $\mathbf{d}_E(y_i, y_j) = \|y_i - y_j\| < \epsilon$ (recall that dimension $d = 1$). If $\epsilon < 1$, then y_i and y_j must be identical (because we already know that they are either 0 or 1). Since this condition holds for any pair of fault-free processes, it implies *exact* consensus. Also, y_i and y_j will be equal to the input at a fault-free process due to the validity property above, and because the inputs are restricted to be 0 or 1. In other words, Algorithm $\mathcal{A}$ can be used to solve exact consensus in the presence of crash faults with incorrect inputs when $n \geq 4f + 1$ in an asynchronous system. This contradicts the well-known impossibility result by Fischer, Lynch, and Paterson [8]. $\square$