Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 230 (2009) 23–38
www.elsevier.com/locate/entcs

Stability of Multi-Valued Continuous Consensus
(Preliminary Version) ⋆

Lior Davidovitch [1]

Department of Computer Science
Ben-Gurion University, Beer-Sheva, 84105, Israel

Shlomi Dolev[2]

Department of Computer Science
Ben-Gurion University, Beer-Sheva, 84105, Israel

Sergio Rajsbaum [3]

Instituto de Matemáticas, UNAM
Ciudad Universitaria, D.F. 04510, México

Abstract

Multi-valued consensus functions defined from a vector of inputs (and possibly the previous output) to a single output are investigated. The consensus functions are designed to tolerate t faulty inputs. Two classes of multi-valued consensus functions are defined, the *exact value* and the *range value*, which require the output to be one of the non-faulty inputs or in the range of the non-faulty inputs, respectively. The *instability* of consensus functions is examined, counting the maximal number of output changes along a geodesic path of input changes, a path in which each input is changed at most once. Lower and upper bounds for the instability of multi-valued consensus functions are presented. A new technique for obtaining such lower bounds, using edgewise simplex subdivision is presented.

Keywords: Fault-tolerance, sensors, boolean functions, consensus, stability.

1 Introduction

The interest in sensor networks and the way they may control the behavior of a system, being a vehicle, airplane, satellite, or other devices is rapidly growing. The

⋆ Some proofs are omitted from this version.

[1] Email: liord@cs.bgu.ac.il Partly supported by the Frankel center for computer science.
[2] Email: dolev@cs.bgu.ac.il Partly supported by Rita Altura trust chair in computer sciences.
[3] Email: rajsbaum@math.unam.mx. Partly supported by a PAPIIT-UNAM grant.

doi:10.1016/j.entcs.2009.02.015

agreement functions used to ensure smooth and stable control while reflecting the changes in the environment are of great interest. An abstraction of many agreement functions is the *consensus* problem, where a set of n processors get input values from some set V and must agree on a value. There is always a non-triviality *validity* requirement that specifies restrictions on the decided value as a function of the input values and the failure pattern of the execution. Consensus is a fundamental problem in distributed computing that has been widely studied for more than two decades due to its theoretical and practical interest (e.g., [2,7,13]).

Research on consensus concentrated on the above, *one-shot* setting where processors start with their input values, and have to solve consensus once. Real distributed systems often need to solve consensus repeatedly, on inputs received one after the other. Thus, researchers have also investigated *continuous* versions of consensus [4] where processors have to adapt their consensus decisions continuously (e.g. [3,8,12]). A typical situation where continuous consensus problems arise is systems that read values from replicated sensors [15]. A fault-tolerant consensus algorithm is needed to decide on a single reading because sensors usually do not give the exact same reading of a physical parameter, or because some sensors can fail. Although in the simplest (and most often considered in theory) version of consensus the validity requirement is that a decided value must have been the reading of at least one sensor, in many real settings it is desired that the decided value is a value that has been produced by a majority of the sensors. These and other non-trivial validity requirements are possible, but they all imply that as the readings of the sensors change because the physical parameters that are sampled change, the consensus value will have to change: in the extreme case, all sensors can change their readings from one single value to another, forcing the consensus decision to change accordingly.

Although processors sometimes have to change their outputs during the repeated executions of a consensus algorithm, we prefer continuous consensus algorithms that are *stable*, i.e., in which the number of times the decision value is changed is as small as possible. Usually averaging functions are used in an independent way from sample to sample, sometimes combined with agreement protocols (e.g. [14]), and hence, there is no attempt to maximize stability. There are several reasons for preferring a stable consensus system (more are described in [8]). Some sensors are discrete and are used to control actuators, which may also be discrete. There is the possible operational amplification of decision changes, say turning an engine on and off. The energy or other resources consumed are sometimes proportional to the number of transitions; e.g. turning an engine on and off takes energy, time, and reduces its lifetime; some related work in VLSI is [5,16]. Independently of the distributed computing applications, our results have a mathematical interest, they are about discrete functions over vectors; they may be useful to study problems about the number of influencing variables in boolean functions (e.g. [11]).

In [8] we initiated a study of the stability of continuous consensus systems for the case of a set V of binary inputs, $|V| = 2$. We defined an abstract formalization

[4] Sometimes called *long-lived* consensus.

of a continuous consensus system and the stability measures. This formalization, presented in Section 2, is not tied to any specific model of computation, in order to understand the basic stability issues. We considered *memoryless* systems where consecutive one-shot consensus executions are independent, versus the stability of systems that can keep memory of previous executions. We also studied the stability of *symmetric* systems where decisions are taken solely on the basis of the distribution of the different input values, but not on what specific sensor or processor produced as a particular input value. We characterized the stability of systems according to their memory and symmetry properties, proving tight upper and lower bounds for the various cases.

Results: In this paper we extend the results of [8] to the case of multivalued inputs, $|V| \geq 2$. It turns out that this generalization makes all the above variants of the problem much harder. Some of them are much more interesting than those of the binary case, in particular one of the results here uses topological techniques for high dimensional complexes.

Let t be the number of sensors that may give a wrong value. The validity requirement of [8] is that the decision value is a value of a correct sensor; that is, if less than $t+1$ inputs are equal to a value b, then the consensus value is not b. Another natural validity requirement for a multi-valued consensus system, is that the decision should be between to correct sensor values. Thus we study two systems:

Exact value system (EV): the decision is the value of a correct sensor.

Range value system (RV): the decision is in the range of the correct sensor values.

First we identify the combinations of values of $n, |V|, t$ for which the above two systems exist. For EV we prove that it is necessary and sufficient that $n \geq |V|t+1$. For RV we prove that it is necessary and sufficient that $n \geq 2t+1$.

The stability of a consensus system with memory is analyzed proving that it is n in the cases where n is the smallest possible value, namely $n = vt+1$, and that the instability goes down until it becomes 1 when $n \geq v^2t+1$. The investigation of the rest of the cases for EV systems results in tight bounds for a range of stability values as a function of $n, |V|$ and t.

Lower bounds for the case of memoryless system are obtained for symmetric functions. The lower bounds are achieved using a technique to subdivide a simplex from [9] and Sperner's Lemma. This can be seen as a generalization of Lemma 2 in [10]: from having a change in the decision values in one dimension (ordering the input values in a line from one extreme to the other where two consecutive input vectors differ in exactly one input) to the case of several dimensions where the border between the different extreme values is a simplex. We also present an upper bound for a memoryless symmetric system [6], which is about a factor of 2 away from our lower bound. An interesting open question is to close this gap.

Organization: The rest of the paper is organized as follows. In the next section, we define the system settings and the problem definitions. The case of system with memory is addressed in Section 3. Memoryless systems are considered in Section 4.

2 The Model

We start with a definition of a multi-valued continuous consensus system which extends the definitions for the binary case of [8]. See [8] for a detailed discussion of the motivation of this system, and for applications to specific distributed computing models. Briefly, the system captures a situation where some distributed system gets inputs from sensors (each processor gets an input value from its corresponding sensor), runs some consensus algorithm to agree on one of the sensors' values, gets another vector of sensor values, runs a consensus algorithm on them, and so on. We consider only the state of the distributed system at the beginning of each such round, i.e., when it gets inputs from the sensors. Also, we consider the value decided by the consensus algorithm at the end of each one of these phases. We are not interested in the details of how the consensus algorithm works. Our goal is to study what is the minimum number of decision value changes the algorithm will make over the repeated invocations of the consensus algorithm. Thus, we assume the consensus algorithm decides the same value in every execution starting in the same state and with the same vector of sensor values.

We will use the following notation. Consider a set $V = \{0, 1, \ldots, v-1\}$ of possible sensor values, and an integer $n > 0$ representing the number of processors. Then $\vec{x}$ is a vector of n values from V, i.e. in V^n. Also, $\vec{x}\,[i]$ is the $i-$th element of $\vec{x}$. We denote the $i-$th vector in a sequence of vectors by $\vec{x}_i$. Thus, $\vec{x}_i\,[j]$ stands for the $j-$th element of the $i-$th vector in a series of vectors. For a vector $\vec{x}$, $\#b(\vec{x})$ is the number of entries of $\vec{x}$ that are equal to b.

2.1 Continuous Consensus Systems and Instability

A multi-valued *continuous consensus system* D, called for short a *system*, is defined by a 6-tuple, $\mathrm{D} = \langle n, v, t, S, \tau, f \rangle$, where

- $n > 1$ is an integer representing the number of processors;
- $v > 1$ is an integer specifying the size of the set V of valid processors' inputs, $V = \{0, 1, \ldots, v-1\}$;
- $t \geq 0$ is a fault tolerance integer parameter;
- S is a set of *states,* that includes a special *initial* state is s_0;
- τ is a transition function, $\tau : V^n \times S \to S$.
- f is a *decision* function $f : V^n \times S \to V$ such that $f(\vec{x}, s)$ is equal to one of the entries of $\vec{x}$, for every $s \in S$.

An *input vector* is any vector $\vec{x} \in V^n$. A sequence of input vectors, $\vec{x}_0, \vec{x}_1, \vec{x}_2, \ldots$, induces a unique *execution* of the system, $(\vec{x}_0, s_0) \to (\vec{x}_1, s_1) \to (\vec{x}_2, s_2) \to \ldots$, where for every i $s_{i+1} = \tau(\vec{x}_i, s_i)$. The *$i-$th output* for the execution is $f(\vec{x}_i, s_i)$. Note that the sequence of input vectors determines uniquely a sequence of output values, $f(\vec{x}_0, s_0), f(\vec{x}_1, s_1), \ldots$.

We use the term *path* for a sequence of input vectors $\vec{x}_0, \vec{x}_1, \ldots$. A value k *changes in the path* if the path contains two vectors $\vec{x}_i, \vec{x}_j$ which differ in their $k-$th

components; i.e., such that $\vec{x}_i[k] \neq \vec{x}_j[k]$. A *geodesic path* is a path in which each component k changes at most once. Notice that a geodesic path can be of length at most n, since the vectors of the geodesic paths we consider in this paper are of dimension n.

Definition 2.1 The *instability* of the system D is the largest number of decision changes for any *geodesic* input path.

The fault-tolerant parameter t represents the largest number of input values in a vector $\vec{x}$ that may be wrong. It is desirable that the decision function f chooses an input value that is not wrong. Therefore, if we require that the value d decided by f in $\vec{x}$ appears at least $t+1$ in $\vec{x}$, then d is certainly not wrong. A system satisfying this requirement is denoted EV, formally:

Definition 2.2 [Exact value system] A system D is *exact value* if for any input vector $\vec{x}$ and any system state s, if $f(\vec{x}, s) = b$ then $\#b(\vec{x}) \geq t+1$.

This is the natural generalization to the requirement of the binary case [8]. In the case of a multi-valued consensus system, we may consider also a different requirement: that the decided value is between to correct values; i.e, that if b is the consensus value, then there are at least *two* non-faulty processors, p_i and p_j, such that their inputs, $\vec{x}[i]$ and $\vec{x}[j]$, satisfy $\vec{x}[i] \leq b \leq \vec{x}[j]$. Such a system is denoted RV:

Definition 2.3 [Range value system] A system D is *range value* if for any input vector $\vec{x}$ and any system state s, if $f(\vec{x}, s) = b$ then $\sum_{b' \geq b} \#b'(\vec{x}) \geq t+1$, and $\sum_{b' \leq b} \#b'(\vec{x}) \geq t+1$.

If *less than* $t+1$ inputs are equal to or smaller than b, or *less than* $t+1$ inputs are equal to or greater than b, then the decision is *not* b. This constraints assures that if b is the consensus value, then there are at least two non-faulty processors, p_i and p_j, such that their inputs, $\vec{x}[i]$ and $\vec{x}[j]$, satisfy $\vec{x}[i] \leq b \leq \vec{x}[j]$. We use RV to denote a range value consensus system.

An *execution* is a sequence $(\vec{x}_0, s_0) \rightarrow (\vec{x}_1, s_1) \rightarrow (\vec{x}_2, s_2) \rightarrow \ldots$, where $s_0 = 0$, and for every i $\vec{x}_i \in V^n$, $s_{i+1} = \tau(\vec{x}_i, s_i)$. The *$i-th$ output* for the execution is $f(\vec{x}_i, s_i)$. Note that the sequence of input vectors determines uniquely a sequence of output values. We use the term *path* for a sequence of vectors $\vec{x}_0, \vec{x}_1, \ldots, \vec{x}_m$. A value k *changes in the path* if the path contains two vectors $\vec{x}_i, \vec{x}_j$ which differ in their $k-$th components; i.e., such that $\vec{x}_i[k] \neq \vec{x}_j[k]$. A *geodesic path* is a path in which each value k changes at most once. The *instability* of a system D is the largest number of decision changes for any *geodesic* input path.

2.2 The Scope for Exact and Range Value Systems

Before we continue describing our results we need to describe for what combination of the parameters n, v and t an exact or a range value system exists. First we show that for any n, v, t such that $n \geq vt+1$ an exact value system can be defined, and

if EV is an exact value system then $n \geq vt + 1$. This result generalizes the binary case of [8], which states that $n \geq 2t + 1$.

Theorem 2.4 *An exact value system* EV *can be defined if and only if* $n \geq vt + 1$.

For range value systems, we have that any such system RV must satisfy $n \geq 2t+1$, and if RV is a range value system then $n \geq 2t + 1$:

Theorem 2.5 *A range value system* RV *can be defined if and only if* $n \geq 2t + 1$, *with* $v > 1$.

3 Stability of Systems with Memory

In Section 3.1 we study the instability of a system with the least possible number of processors n, and show that when n is big enough, the instability is 1. The case of a general n is considered in Section 3.2. Then, in Section 4, we study a system that cannot keep any memory from previous input vectors, and from its previous decisions.

It is convenient to visualize the input vector $\vec{x}$ as a bar chart, where every bar relates to a possible input value b, and the height of that bar is $\#b\,(\vec{x})$. Notice that an input value may be the consensus only if the height of the bar is at least $t + 1$. This is exemplified in Figure 1.

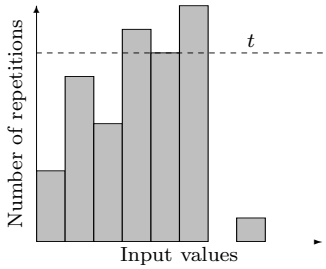

Fig. 1. A visualization of the input vector as a bar chart – every bar represents an input value, and its height is the number of repetitions of that value within the input vector. The dashed line represents height of t. Notice that only two input values are a valid consensus value.

3.1 The extreme cases of n

We will first consider the extreme case in which n is the minimal possible number of processors: $vt+1$ for EV systems as shown in Lemma 2.4, and $2t+1$ for RV systems as shown in Lemma 2.5. We prove that the instability of the system in this case is n; that is, it is possible to force any system to change its decision with each input change.

We have just seen that when n is as small as it can be, i.e., $n = vt + 1$, then we cannot reduce the number of decision changes: it is possible to force any EV system D to change its decision with each input change. In contrast, when n is large, we can build a system that changes its decision only once. This will be proved in Lemma 3.4.

Lemma 3.1 *For every* EV *system* D *with* $n = vt + 1$, $instability(\text{D}) = vt + 1$.

Proof. Obviously $instability(\text{D}) \leq n = vt + 1$, since any geodesic path is of length at most n, and hence at most n decision value changes can occur.

To prove that $instability(\text{D}) = n = vt + 1$, we construct a geodesic path of length $vt+1$ that produces $vt+1$ changes in system D. We will start with the input vector $\vec{x}_0 = (01 \dots (v-1))^t 0$, and to get $\vec{x}_{i+1}$, switch the input value of the $i+1$-th processor from i mod v to $i+1$ mod v. The geodesic path can be illustrated as follows:

$$
\begin{aligned}
&(012\dots(v-2)(v-1))^1(012\dots(v-1))^{t-1}0 \rightarrow\\
&(112\dots(v-2)(v-1))^1(012\dots(v-1))^{t-1}0 \rightarrow\\
&(122\dots(v-2)(v-1))^1(012\dots(v-1))^{t-1}0 \rightarrow\\
&(123\dots(v-2)(v-1))^1(012\dots(v-1))^{t-1}0 \rightarrow\\
&\dots \rightarrow\\
&(123\dots(v-1)(v-1))^1(012\dots(v-1))^{t-1}0 \rightarrow\\
&(123\dots(v-1)0)^1(01\dots(v-1))^{t-1}0 \rightarrow\\
&\dots \rightarrow\\
&(123\dots(v-1)0)^2(01\dots(v-1))^{t-2}0 \rightarrow\\
&\dots \rightarrow\\
&(123\dots(v-1)0)^t0 \rightarrow\\
&(123\dots(v-1)0)^t1
\end{aligned}
$$

Notice that $\#b(\vec{x}_0) = t$ for $b \neq 0$ and $\#0(\vec{x}_0) = t+1$. Since the only input value that appears more than t times is 0, the consensus value must be 0. In general, a simple induction shows that $\#b(\vec{x}_i) = t$ for $b \neq i$ mod v and $\#b(\vec{x}_i) = t+1$ for $b = i$ mod v. Therefore, the consensus value in $\vec{x}_i$ is i mod v, and there is a consensus value change with each new input vector; i.e., $instability(\text{D}) = n$. □

The previous result generalizes the result in [8] where it was proved that when $n = 2t + 1$ and $v = 2$ the instability of the system is n.

Lemma 3.2 *For every* RV *system* D *with* $n = 2t+1$ *and any* $v \geq 2$, $instability(\text{D}) = 2t + 1$.

Proof. Obviously $instability(\text{D}) \leq n = 2t + 1$. To prove that $instability(\text{D}) = n = 2t+1$, we consider the path in the proof of Lemma 3.1, for the case of only two input values, $0, 1$ (the same as the path in Theorem 3.2 of [8]). Since this path includes only input values from the set $\{0, 1\}$, and $\{0, 1\} \subseteq V$, it is a valid input path for system D with v. The path is of length $2t + 1$, and it is easy to see that it produces a consensus value change with each new input vector: for i even, $\vec{x}_i$ has $\#1(\vec{x}_i) = t$ and $\#0(\vec{x}_i) = t + 1$, while for i odd, $\vec{x}_i$ has $\#0(\vec{x}_i) = t$ and $\#1(\vec{x}_i) = t + 1$. □

It is easy to see that, for any EV system D with $v > 1$, there is a geodesic path that forces it to make at least one decision change:

Lemma 3.3 *For every* EV *system* D *with* $v > 1$, *it holds that instability*(D) ≥ 1.

In Lemma 3.1 we considered the instability of an exact value system for the smallest possible value of n, namely $n = vt + 1$ (that this is the smallest value was proved in Lemma 2.4), and showed that the instability is the largest possible, namely n. There is no upper bound for the number of processors, since for every $n \geq vt + 1$ there exists an exact value system. However, we prove next that for $n \geq v^2t + 1$ the instability is the smallest possible, namely 1, matching the trivial lower bound of Lemma 3.3. To prove it we describe a simple system, MS, with $\log(2vn + v + 1)$ bits of memory. Also, we show that for $n < v^2t + 1$ no system can achieve an instability of 1.

For its initial state, the system MS decides on the most frequent value in $\vec{x}_0$. The system MS remembers in its state its last consensus value, c, and a vector $\vec{h}$ that records in $\vec{h}[b]$, for each $b \in V$, how many times an input value has switched to b. Then, MS will keep its previous decision, c, unless forced to change, that is, will change decision only if $\#c(\vec{x}) = t$ in the current input vector $\vec{x}$. In this case, if there is a value b such that $\vec{h}[b] \geq t + 1$ it will decide b since then no more changes will occur. Otherwise, it will decide on the most frequent value in $\vec{x}$.

To compute $\vec{h}$ the system must remember also the previous input vector; actually it must remember only the multiplicity of each input value in the previous input vector, and not the position in the vector of each value. Thus, for a vector $\vec{x}$, let $bar(\vec{x})$ be the vector of dimension v such that $bar(\vec{x})[b]$ is equal to the number of times $\vec{x}[i] = b$, over $1 \leq i \leq n$. This is illustrated in Figure 1. Therefore, the current state is of the form $\tilde{s} = \left\{\vec{x}_{prev}, c, \vec{h}\right\}$. For example, if the current state is $\tilde{s} = \{(4, 5, 1), 1, (2, 0, 1)\}$, then: the input vector contains 4 inputs with value 0, 5 inputs with value 1, and 1 input with value 2; the consensus value is 1; and 2 inputs values where changed to 0 and 1 input to 2 in the input path $\vec{x}_0, \vec{x}_1, \vec{x}_2, \vec{x}_3$ (since there have been 3 value changes, the path is of length 3). Formally, the system is defined as follows, where $\tilde{s} = \left\{\vec{x}_{prev}, c, \vec{h}\right\}$:

$$f(\vec{x}, \tilde{s}) = \begin{cases} \text{maxarg}_{b \in V} \{\#b(\vec{x})\} & \text{if } \tilde{s} = s_0 \\ \begin{cases} \text{maxarg}_{b \in V} \left\{\vec{h}[b+1]\right\} & \text{if } \exists b, \vec{h}[b] \geq t+1 \\ \text{maxarg}_{b \in V} \{\#b(\vec{x})\} & \text{otherwise} \end{cases} & \\ & \text{if } \#c(\vec{x}) \leq t \\ c & \text{if } \#c(\vec{x}) \geq t+1 \end{cases}$$

$$\tau(\vec{x},\tilde{s}) = \begin{cases} \{\vec{x}, f(\vec{x},\tilde{s}), (0,\ldots,0)\} & \text{if } \tilde{s} = s_0 \\ \left\{\vec{x}, f(\vec{x},\tilde{s}), \vec{h} + \sum_{i=1}^{v} \vec{\delta}_i \max\{0, \vec{x}[i] - \vec{x}_{prev}[i]\}\right\} & \text{otherwise} \end{cases}$$

where:

$$\vec{\delta}_m[k] = \begin{cases} 1 & , m = k \\ 0 & , m \neq k \end{cases}$$

This system is a generalization of the system BP, described in [8]. We will prove that for a large enough number of processors the instability of this system is one.

Lemma 3.4 *When $n \geq v^2 t + 1$, instability$(MS) = 1$.*

3.2 The General Case

A vector $\vec{h}$ as in system MS is called an *accumulator* vector. Notice that for any geodesic input path P we can define an associated accumulator vector associated to each of the vectors of P, that records the number of times each value has been changed into. That is, initially $\vec{h} = 0^v$, and if one input bit changes from some value b to b', than $\vec{h}[b']$ is incremented by one. Thus, an accumulator vector is independent of the EV system been used, and is defined even if the system does not actually use it; it is just a devise that we use to reason about the system. The following lemma captures the importance of $\vec{h}$, and it will be useful to prove lower bounds. Here by a *optimal* we mean that the system is optimal in terms of instability.

Lemma 3.5 *Let* D *be an optimal* EV *system. Consider an input vector $\vec{x}$ at the end of a geodesic input path P, and the corresponding vector $\vec{h}$. Assume the consensus value in $\vec{x}$ is b. There is a geodesic path extending P that causes at least one consensus change to* D *if and only if $\vec{h}[b] \leq t$.*

Lemma 3.6 *For any* EV *system* D, *instability*(D) ≥ 2, *if $n \leq v^2 t$,*

We now prove our main upper bound result for EV systems with memory. For any vector $\vec{y}$, define $|\vec{y}|$ to be the $\sum_i \vec{y}[i]$, and $\max \vec{y}$ to be $\max_i \vec{y}[i]$. We will use the fact that $|\vec{h}| \geq vt + 1$ implies $\max \vec{h} \geq t + 1$, and then apply Lemma 3.5.

Lemma 3.7 *Let $n = vt + x(v-1)$, for $x \geq 1$. Then instability$(MS) \leq \lfloor \binom{vt-y}{x} \rfloor + 2$, where $y = \lceil \binom{x(v-1)}{v} \rceil$.*

Consider the extreme cases for the previous lemma. First, $x = vt$, so we have $n = v^2 t$, $y = t(v-1)$. We get *instability*$(MS) \leq \lfloor \binom{vt-t(v-1)}{vt} \rfloor + 2$, and hence *instability*$(MS) \leq \lfloor \frac{1}{v} \rfloor + 2 = 2$. Which matches the lower bound of Lemma 3.6. Second, let $x = 1$, so we have $n = vt + v - 1$, $y = \lceil \binom{(v-1)}{v} \rceil = 1$. We get *instability*$(MS) \leq \lfloor \binom{vt-1}{1} \rfloor + 2 = vt + 1$. Which matches the lower bound of Lemma 3.1.

A similar argument can be used to prove a corresponding upper bound:

Lemma 3.8 *Let $n = vt + x(v-1)$, for $x \geq 1$, and $y = \lceil \frac{x(v-1)}{v} \rceil$. For every* EV *system* D *instability*(D) $\geq \lfloor \frac{vt-y}{x} \rfloor + 2$.

Summarizing the stability results for EV

The next theorem summarizes the results obtained for EV systems with memory. Let *instability*(EV) denote the smallest instability over all EV systems.

Theorem 3.9

$$\textit{instability}\,(\text{EV}) = \begin{cases} vt+1 & \text{if } n = vt+1 \\ \lfloor \frac{vt-y}{x} \rfloor + 2 & \text{if } n = vt + x(v-1), \text{ where } y = \lceil \frac{x(v-1)}{v} \rceil, \\ 1 & \text{if } n \geq v^2 t + 1 \end{cases}$$

4 Stability of Symmetric Memoryless Systems

In this section we turn to investigate the case in which no memory is kept by the system from previous input vectors and from previous decisions, that is, we assume a *memoryless* system $\text{D} = \langle n, v, t, S, \tau, f \rangle$, where $S = \{s_0\}$. Hence instead of $f(\vec{x}, s_0)$ we simply write $f(\vec{x})$ for any input vector $\vec{x}$. Furthermore, we assume the system is *symmetric*, meaning the decision function is oblivious to the order of the input values; i.e., for any two input vectors $\vec{x}, \vec{y}$, $bar(\vec{x}) = bar(\vec{y})$ implies $f(\vec{x}) = f(\vec{y})$. Thus, f is a function of just $\#i(\vec{x})$ for every $i \in V$, and we study in this section the corresponding function $\tilde{f}$ defined on the set A,

$$A = \{(\alpha_0, \alpha_1, \ldots, \alpha_{v-1}) \mid \alpha_i \in \mathbb{N} \cup \{0\}, \alpha_0 + \alpha_1 + \cdots + \alpha_{v-1} = n\}.$$

And we define $\tilde{f} : A \to V$ in terms of f as follows:

$$\tilde{f}(\alpha_0, \alpha_1, \ldots, \alpha_{v-1}) = f(0^{\alpha_0} 1^{\alpha_1} \cdots (v-1)^{\alpha_{v-1}}).$$

4.1 Overview of the lower bound

The lower bound proof is a generalization of the one in [10], from dimension 1 to higher dimensions. The proof uses Sperner's Lemma, a well known result that generalizes the graph connectivity notion to higher dimensions. See e.g. [4] for a proof and see e.g. [1] for a recent application to distributed computing.

In dimension 1, Sperner's Lemma says that if the vertices of a path are colored with $0, 1$, with 0 coloring one end-vertex and 1 coloring the other, then at least one edge will have its two vertices colored with different colors. More generally, if we try to map a connected graph into two isolated vertices, $0, 1$ (which form a disconnected graph), then at least one edge of the graph will have to cross from 0 to 1.

In general, a *k-simplex* is a set of $k+1$ vertices; the *dimension* k of the simplex is equal to its number of vertexes minus one. A *k-face* of a simplex is a subset of $k+1$

of its vertices. A 0-face is just a vertex. A *complex* is a set of simplexes closed under containment. It is often convenient to assume that these vertices are points in space and that the simplex is the convex hull of these points. A *subdivision* of a simplex is a partition of the simplex into simplexes, such that the union of the dividing simplexes equals the original simplex, and any two dividing simplexes intersect at most only in their boundaries. A *Sperner Coloring* of a subdivided k-simplex $\mathcal{S}$ is an assignment of colors $\{0, ..., k\}$ to its vertices such that the $k+1$ corners are assigned the $k+1$ distinct colors, and the vertices in the subdivision of a face of $\mathcal{S}$ are assigned colors from the corners of that face.

In our proof we put the input vectors of A in a space of dimension v. The input vectors in which one component is n and all the rest are zero form a simplex, in fact a $(v-1)-$simplex. The rest of the vertices are all convex combination of the above vertices, and therefore reside within the simplex. We use Sperner's Lemma to conclude that there is a dividing simplex such that its v vertices have v distinct decision values.

There are several ways to define such a subdivision, fortunately we found in [9] a subdivision that serves us well, in finding a geodesic path of length $vt+1$ in every dividing simplex. In particular, we have such a path in the dividing simplex that has different function values for each of its vertices.

We illustrate the instability lower bound for the case of $v = 3$. This should give the reader a more intuitive view of the general proof.

When $v = 3$, the set A can be used as vertices of a subdivision of a 2-simplex — or in other words, a solid flat triangle. This triangle can be divided into subsimplexes (subtriangles), such that the vertices of a subtriangle represent input vectors that are adjacent, in the sense that they can result from each other by a change of a single input value. The vertices will be colored by the consensus value of a given input vector, and using Sperner's Lemma we will find a sub-triangle which its vertices' coloring is pairwise different. This sub-triangle will imply a geodesic path with $3t+1$ changes to the consensus value.

Lemma 4.1 *Let* D *be a symmetric memoryless* EV *system with* $v = 3$. *Then* $instability(\text{D}) \geq 3t+1$.

4.2 Lower Bound for Memoryless Systems

We will now prove that for any EV system D with a symmetric function f, $instability(\text{D}) \geq vt+1$. To do so, we will use the *edgewise subdivision of a simplex*, defined in [9] (See Appendix A for details).

Let $\mathcal{S}$ be a $d-$simplex, spanned by $\vec{V}_0, \vec{V}_1, \ldots, \vec{V}_d$. An edgewise subdivision is a function that, given an integer k, transforms every point $\vec{X} \in \mathcal{S}$ into a color scheme M, which is defined by a matrix as follows:

$$M = \begin{pmatrix} \chi_{1,0} & \chi_{1,1} & \cdots & \chi_{1,j} \\ \chi_{2,0} & \chi_{2,1} & \cdots & \chi_{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ \chi_{k,0} & \chi_{k,1} & \cdots & \chi_{k,j} \end{pmatrix}$$

where $j \leq d$. Each entry of the matrix is an integer from 0 through d, the columns are pairwise different, and the entries appear in non-decreasing order when read like English text:

$$\chi_{1,0} \leq \chi_{1,1} \leq \ldots \leq \chi_{1,j} \leq \chi_{2,0} \leq \ldots \leq \chi_{k,j}.$$

The color scheme defines $j+1$ independent vectors $\vec{V}_0^*, \vec{V}_1^*, \ldots, \vec{V}_j^*$, where $\vec{V}_l^* = \frac{1}{k} \sum_{i=1}^{k} \vec{V}_{\chi_{i,l}}$ which span a j-simplex. By applying the function to every point $\vec{X} \in \mathcal{S}$ we obtain a subdivision of $\mathcal{S}$ into subsimplices, some of them are $d-$simplices.

Recall that

$$\vec{\delta}_m[k] = \begin{cases} 1 & , m = k \\ 0 & , m \neq k \end{cases}$$

Lemma 4.2 *Let $\mathcal{S}$ be the $(v-1)-$simplex, spanned by $\vec{V}_0, \vec{V}_1, \ldots, \vec{V}_{v-1}$, where $\vec{V}_i = n \cdot \vec{\delta}_{i+1}$. Let $\vec{a}$ be a vertex of any $(v-1)-$simplex in the edgewise subdivision of $\mathcal{S}$ using $k = n$. Then for every $1 \leq i \leq v$, $\vec{a}[i]$ is an integer. Moreover, $\sum_{i \in V} \vec{a}[i] = n$.*

Lemma 4.2 implies that for every vertex $\vec{a}$ of a $(v-1)-$simplex of the edgewise subdivision of $\mathcal{S}$, $\vec{a} \in A$. We will color every vertex $\vec{a}$ of the subdivision with $\tilde{f}(\vec{a})$. Since it holds that for every $\vec{a} \in A$ such that $\tilde{f}(\vec{a}) = b$, $\#b(\vec{a}) \geq t+1 > 0$, then the coloring is a Sperner coloring, and according to Sperner's lemma there must exist in the subdivision a subsimplex $\mathcal{S}^*$ such that all its vertices' colors are different.

Lemma 4.3 *Let $\mathcal{S}$ be a $d-$simplex spanned by $\vec{V}_0, \vec{V}_1, \ldots, \vec{V}_d$, such that for every $0 \leq i \leq d$, $\vec{V}_i = n \cdot \vec{\delta}_i$. Let $\mathcal{S}^*$ be a $d-$simplex of the edgewise subdivision of $\mathcal{S}$ using the integer n, when $\mathcal{S}^*$ is spanned by $\vec{V}_0^*, \vec{V}_1^*, \ldots, \vec{V}_d^*$. Then $\vec{V}_0^* \to \vec{V}_1^* \ldots \to \vec{V}_d^* \to \vec{V}_0^*$ is a geodesic path with minimal changes.*

Now we can prove the main theorem:

Theorem 4.4 *For every* EV *system* D *with a symmetric function f, instability*$(\mathrm{D}) \geq vt+1$.

Proof. $\tilde{f}$ is defined over A, which subdivides $\mathcal{S}$ using the edgewise subdivision (Lemma 4.2). We will use $\tilde{f}$ to color the vertices in A. According to Sperner's Lemma, there is a $(v-1)-$simplex in the subdivision so that the colors of its vertices are pairwise different. Let $\vec{a}_0, \vec{a}_1, \ldots, \vec{a}_v \in A$ be the vertices spanning $\mathcal{S}$.

As shown above, for every i, b, it holds that $\#b(\vec{a}_i) \geq t$, and for every b there exists i_b such that $\#b(\vec{a}_{i_b}) \geq t+1$, where $i_{b_1} \neq i_{b_2}$ for every $b_1 \neq b_2$. Without loss of

generality, we will assume $\#0(\vec{a}_0) \geq t+1$. Then $\vec{a}_0$ corresponds to the input vector $0^t 1^t \dots (v-1)^t 0\vec{z}$.

According to Lemma 4.3, $\vec{a}_0 \to \vec{a}_1 \to \dots \to \vec{a}_v \to \vec{a}_0$ is a geodesic path with minimal changes. Let c_j the color that change between $\vec{a}_j$ and $\vec{a}_{j+1}$ for every $0 \leq j \leq v-1$, and let $c_v = v$. Now, for every step i we start with an input vector corresponding to $\vec{a}_{i \bmod v}$, and will switch an input value $c_{i \bmod v}$ to $(c_{i \bmod v} + 1) \bmod v$, thus arriving to an input vector corresponding to $\vec{a}_{(i+1) \bmod v}$. We can repeat these steps $vt+1$ times, for every input value $b \neq c_0$, the original input vector holds $\#b(\vec{a}_0) \geq t$, and also $\#0(\vec{a}_0) \geq t+1$. And since the colors of the vertices are pairwise different, then the values of $\tilde{f}$ over the vertices are pairwise different, and therefore the values of f over the input vectors corresponding to the vertices are pairwise different (between pairs corresponding to different vertices), then this path yields $vt+1$ changes to the consensus value. □

4.3 Instability upper bound, MLS *system*

We now describe a memoryless symmetric EV system MLS, and prove that its instability is a factor of two away from the lower bound presented in the previous section. An interesting open question is to close this gap. Since MLS is memoryless and symmetric, it is defined solely by its decision function f_{MLS}. We will define f_{MLS} as follows:

$$f_{MLS}(\vec{x}) = \min\{b \mid \#b(\vec{x}) \geq t+1\}$$

Namely, f_{MLS} decides upon the smallest value possible.

Lemma 4.5 *Instability*(MLS) $\geq 2(v-1)(t+1)$, *when* $n \geq 2(v-1)(t+1)$.

Lemma 4.6 *Instability*(MLS) $\leq 2(v-1)(t+1)$.

Proof. Let $P = \vec{x}_0 \to \vec{x}_1 \to \dots \to \vec{x}_n$ be a geodesic path. The number of decision value changes in P is equal to the sum of the times the decision changes to a higher value plus the times it changes to a lower value, i.e., $\# \uparrow + \# \downarrow$. We next show that each one of these quantities is at most $(v-1)(t+1)$, which proves the lemma.

To compute $\# \uparrow$ notice that if the decision d changes to a higher value then it must be that an input bit equal to d changes: in the current input $\vec{x}_i$ we must have $\#d(\vec{x}_i) = t+1$ while $\#d(\vec{x}_{i+1}) = t$. Now,

$$\# \uparrow = \sum_{d=0}^{v-2} \# \uparrow_d,$$

where $\# \uparrow_d$ denotes the number of changes of decision from d to a higher value. Consider the first time, $\vec{x}_j$, where the decision is d and the change in $\vec{x}_{j+1}$ is to a higher value. We have $\#d(\vec{x}_j) = t+1$ and therefore the total subsequent number of changes from d to a higher value is at most $t+1$ (since once this happens $t+1$ inputs equal to d have changed and thus fixed in a geodesic path).

To compute $\# \downarrow$ observe that if the decision d changes in $\vec{x}_i$ to a smaller value d' then it must be that a value changes to d': $\#d'(\vec{x}_i) = t$ and $\#d'(\vec{x}_{i+1}) = t+1$. This can happen at most $t+1$ times for each such d' in a geodesic path (once this happens there are $t+1$ entries with value d' fixed). There are $(v-1)$ such values d', so the total number of such changes is at most $(v-1)(t+1)$ □

Corollary 4.7 *Instability*(MLS) $= 2(v-1)(t+1)$.

Acknowledgments. It is a pleasure to thank Ronen Peretz, Nir Shavit and Yuri Vasilevski for their comments.

References

[1] H. Attiya and S. Rajsbaum, "The Combinatorial Structure of Wait-Free Solvable Tasks," *SIAM Journal of Computing,* 31(4), 2002, pp. 1286–1313.

[2] H. Attiya and J. Welch, *Distributed Computing: Fundamentals, Simulations and Advanced Topics,* McGraw–Hill, 1998.

[3] A. Bar-Noy, X. Deng, J. Garay, T. Kameda, "Optimal amortized distributed consensus," *Info. and Comp.,* 120(1):93-100, 1995. Prel. in WDAG'91.

[4] J.A. Bondy and U.S.R. Murty, *Graph theory with applications,* American Elsevier, 1976.

[5] A.P. Chandrakasan and R.W. Brodersen, *Low power digital CMOS design,* Kluwer Academic Publishers, 1995.

[6] L. Davidovitch, S. Dolev and S. Rajsbaum. "Consensus Continue? Stability of Multi-Valued Continuous Consensus!", Technical Report #2004-03, Department of Computer Science, Ben-Gurion University, May 2004.

[7] S. Dolev, *Self-Stabilization*, MIT Press, 2000.

[8] S. Dolev, and S. Rajsbaum, "Stability of Long-lived Consensus" *Proc. of the 19th Annual ACM Symp. on Principles of Distributed Computing*, (PODC 2000), pp. 309-318, 2000, and *Journal of Computer and System Sciences*, Vol. 67, Issue 1, pp. 26-45, August 2003.

[9] H. Edelsbrunner and D. R. Grayson, "Edgewise subdivision of a simplex", *Discrete Comput. Geom. 24*, 2000, 707-719

[10] M.J. Fischer, N.A. Lynch and M.S. Paterson, "Impossibility of Distributed Consensus with One Faulty Process," *Journal of the ACM* 32, 1985, pp. 374-382. Extended abstract in the *ACM Symposium on Principles of Database Systems*, 1983.

[11] J. Kahn, G. Kalai and N. Linial, "The Influence of Variables on Boolean Functions," *Proc. of the IEEE FOCS*, 1988.

[12] L. Lamport, "The part-time parliament," *ACM Trans. on Comp. Systems,* vol. 16 (2):133-169, May 1998.

[13] N.A. Lynch, *Distributed Algorithms,* Morgan Kaufmann Publishers, Inc. 1996.

[14] K. Marzullo, "Tolerating failures of continuous-valued sensors," *ACM Trans. on Comp. Systems,* 8(4):284-304, Nov. 1990.

[15] H. Kopetz and P. Veríssimo, "Real Time and Dependability Concepts," chapter 16, pp.411-446 in Sape Mullender (ed.), *Distributed Systems,* ACM Press, 1993.

[16] E. Musoll, T. Lang, J. Cortadella, "Exploiting the locality of memory references to reduce the address bus energy," *Proc. of the Int. Symp. on Low Power Electronics and Design,* Aug. 1997, pp. 202–207.

Appendix

A Edgewise subdivision

In [9] an edgewise subdivision of a $d-$simplex is suggested, which uses an abacus model of the simplex. In this appendix we will briefly describe the method described in [9], and its main results.

Let $\vec{V}_0, \vec{V}_1, \ldots, \vec{V}_d$ be independent vectors which define a $d-$simplex $\mathcal{S}$ in $\mathbb{R}^d$. Every point $\vec{X} \in \mathcal{S}$ can be described by $\lambda_0, \lambda_1, \ldots, \lambda_d$ non-negative real numbers which sum to 1, so that

$$\vec{X} = \sum_{i=0}^{d} \lambda_i \cdot \vec{V}_i$$

These are called *the barycentric coordinates* of $\vec{X}$. We may portray them graphically by drawing the unit interval as a rectangle with regions colored from 0 through d, making sure to arrange the colors from left to right, so that λ_i is the fraction of points with color i. Figure A.1 illustrates this for $d = 7$ and the point with barycentric coordinates $(0.26, 0.11, 0.07, 0.11, 0.19, 0.08, 0.04, 0.14)$.

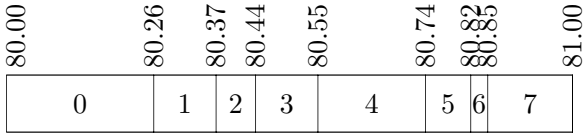

Fig. A.1. The rectangle represents the unit interval with points colored from 0 to 7.

The coordinates of the dividing lines are displayed above the rectangle. They are provided by the partial sums $0 = B_0, B_1, \ldots, B_d, B_{d+1} = 1$ with $B_i = \lambda_0 + \lambda_1 + \ldots + \lambda_{i-1}$. B_1 through B_d can be any non-decreasing sequence in the unit interval.

Suppose we chop the rectangle in Figure A.1 into k pieces of equal width, stack them on top of each other, and expand the scale by factor of k in the horizontal direction (see Figure A.2).

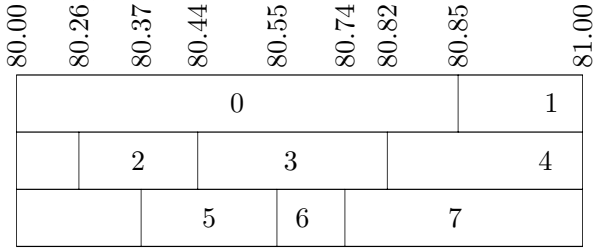

Fig. A.2. The rectangle in Figure A.1 is chopped into three pieces, which are stacked and expanded.

The coordinates of the dividing lines are obtained by multiplying the coordinates of the earlier dividing lines by k and discarding the integer part, keeping only the fractional part. We discard any duplications, letting $j+2$ be the number of distinct values remaining. Call these numbers $0 = C_0, C_1, \ldots, C_j, C_{j+1} = 1$, making sure

to sort them in increasing order. We extend the dividing lines vertically until they span the entire stack, and label each region by its color (see Figure A.3).

80.00	80.26	80.37	80.44	80.55	80.74	80.82	80.85	81.00
0	0	0	0	0	0	0	1	
1	2	2	3	3	3	4	4	
4	4	5	5	6	7	7	7	

Fig. A.3. Each rectangle is cut into equally many regions, and each region keeps the original color of its points.

The number of regions of each stack is $j+1$. Forgetting the positions of the vertical dividing lines we get a matrix:

$$M = \begin{pmatrix} \chi_{1,0} & \chi_{1,1} & \cdots & \chi_{1,j} \\ \chi_{2,0} & \chi_{2,1} & \cdots & \chi_{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ \chi_{k,0} & \chi_{k,1} & \cdots & \chi_{k,j} \end{pmatrix}$$

of $k(j+1)$ color entries. We call $M = M(\vec{X}, k)$ a *color scheme* because it determines the combinatorics but not the geometry of the coloring. The matrices that may occur are those whose entries are drawn from the set $\{0, 1, \ldots, d\}$, whose entries are in non-decreasing order when read like English text:

$$\chi_{1,0} \leq \chi_{1,1} \leq \cdots \leq \chi_{1,j} \leq \chi_{2,0} \leq \cdots \chi_{k,j}$$

and whose columns are pairwise different.

For every k not necessarily distinct colors χ_1 through χ_k we introduce the notation

$$\vec{V}_{\chi_1,\chi_2,\ldots,\chi_k} = \frac{1}{k}(\vec{V}_{\chi_1} + \vec{V}_{\chi_2} + \ldots \vec{V}_{\chi_k})$$

Now we can define for every $0 \leq l \leq j$, $\vec{V}_l^* = \vec{V}_{\chi_{1,l},\chi_{2,l},\ldots,\chi_{k,l}}$, each vector corresponds to a column in the color scheme $M = M(\vec{X}, k)$. In [9] it is proved that these vectors are independent, thus spanning a $j-$simplex. Let $\mathcal{S}_M$ denote the simplex spanned by the vectors corresponding to M. Then the *k-edgewise subdivision* of $\mathcal{S}$ consists of all the simplexes defined this way by points of $\mathcal{S}$:

$$\mathrm{Esd}_k(\mathcal{S}) = \left\{ \mathcal{S}_M \mid M = M(\vec{X}, k), \vec{X} \in \mathcal{S} \right\}.$$